

Haixun Wang Shijun Li
Satoshi Oyama Xiaohua Hu
Tieyun Qian (Eds.)

LNCS 6897

Web-Age Information Management

12th International Conference, WAIM 2011
Wuhan, China, September 2011
Proceedings

 Springer

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Alfred Kobsa

University of California, Irvine, CA, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

TU Dortmund University, Germany

Madhu Sudan

Microsoft Research, Cambridge, MA, USA

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max Planck Institute for Informatics, Saarbruecken, Germany

Haixun Wang Shijun Li Satoshi Oyama
Xiaohua Hu Tieyun Qian (Eds.)

Web-Age Information Management

12th International Conference, WAIM 2011
Wuhan, China, September 14-16, 2011
Proceedings

Volume Editors

Haixun Wang
Microsoft Research Asia, Beijing, 100190, China
E-mail: haixunw@microsoft.com

Shijun Li
Wuhan University, Hubei 430072, China
E-mail: shjli@whu.edu.cn

Satoshi Oyama
Hokkaido University, Sapporo, Hokkaido 060-0814, Japan
E-mail: oyama@ist.hokudai.ac.jp

Xiaohua Hu
Drexel University, Philadelphia, PA 19104, USA
E-mail: xh29@drexel.edu

Tieyun Qian
Wuhan University, Hubei, 430072, China
E-mail: qty@whu.edu.cn

ISSN 0302-9743
ISBN 978-3-642-23534-4
DOI 10.1007/978-3-642-23535-1
Springer Heidelberg Dordrecht London New York

e-ISSN 1611-3349
e-ISBN 978-3-642-23535-1

Library of Congress Control Number: 2011934878

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

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

© Springer-Verlag Berlin Heidelberg 2011

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

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

WAIM has been a leading international conference on research, development, and applications of Web technologies, database systems, information management and software engineering. WAIM is focused in the Asia-Pacific region, and previous WAIM conferences were held in Shanghai (2000), Xi'an (2001), Beijing (2002), Chengdu (2003), Dalian (2004), Hangzhou (2005), Hong Kong (2006), Huangshan (2007), Zhangjiajie (2008), Suzhou (2009), and Jiuzhaigou (2010). As the 12th event in the increasingly popular series, WAIM 2011 attracted outstanding researchers from all over the world to Wuhan, China. In particular, this year WAIM and Microsoft Research Asia jointly sponsored a database summer school, which was collocated with WAIM.

This high-quality program would not have been possible without the authors who chose WAIM as a venue for their publications. Out of 181 submitted papers from various countries and regions, we selected 53 full papers for publication. The acceptance rate for regular full papers is 29%. The contributed papers addressed a wide scope of issues in the fields of Web-age information management and advanced applications, including Web data mining, knowledge discovery from streaming data, query processing, multidimensional data analysis, data management support to advanced applications, etc.

A conference like this can only succeed as a team effort. We want to thank the Program Committee Chairs, Program Committee members and the reviewers for their invaluable efforts. Special thanks to the local Organizing Committee headed by Liang Hong and Ming Zhong. Many thanks also go to the Workshop Co-chairs (Chengfei Liu and Liwei Wang), Finance Chairs (Howard Leung and Xuhui Li), and Publicity Co-chairs (Weiyi Meng, Lei Chen and Guohui Li). We also thank the conference sponsors for their generous support.

We hope that you enjoy reading the proceedings of WAIM 2011.

September 2011

Haixun Wang
Shijun Li
Satoshi Oyama

Conference Organization

Honorary General Co-chairs

Yanxiang He
Changjie Tang

Wuhan University, China
Sichuan University, China

General Co-chairs

Katsumi Tanaka
Zhiyong Peng

Kyoto University, Japan
Wuhan University, China

Program Committee Co-chairs

Haixun Wang
Shijun Li
Satoshi Oyama

Microsoft Research Asia
Wuhan University, China
Hokkaido University, Japan

Workshop Co-chairs

Chengfei Liu
Liwei Wang

Swinburne University of Technology, Australia
Wuhan University, China

Industry Co-chairs

Mukesh Mohania
Xiaoxin Wu

IBM, India
Huawei, China

Publication Co-chairs

Xiaohua Hu
Tieyun Qian

Drexel University, USA
Wuhan University, China

Publicity Co-chairs

Weiyi Meng
Lei Chen

Binghamton University, USA
Hong Kong University of Science and
Technology, Hong Kong

Guohui Li

Huazhong University of Science and
Technology, China

Local Organization Co-chairs

Liang Hong	Wuhan University, China
Ming Zhong	Wuhan University, China

Finance Co-chairs

Howard Leung	City University of Hong Kong
Xuhui Li	Wuhan University, China

Steering Committee Liaison

Qing Li	City University of Hong Kong
---------	------------------------------

CCF DBS Liaison

Xiaofeng Meng	Renmin University, China
---------------	--------------------------

Program Co-chairs

Haixun Wang	Microsoft Research Asia
Shijun Li	Wuhan University, China
Satoshi Oyama	Hokkaido University, Japan

Area Chairs

Bin Cui	Peking University, China
Danushka Bollegala	University of Tokyo, Japan
Jianyong Wang	Tsinghua University, China
Kenny Zhu	Shanghai Jiao Tong University, China
Seung-won Hwang	Pohang University of Science and Technology, Korea
Wei Wang	University of New South Wales, Australia
Xiaochun Yang	Northeastern University, China
Alfredo Cuzzocrea	University of Calabria, Italy

Program Committee

Anthony Tung	National University of Singapore, Singapore
Aoying Zhou	East China Normal University, China
Baihua Zheng	Singapore Management University, Singapore
Bin Cui	Peking University, China
Bingsheng He	Chinese University of Hong Kong, Hong Kong
Chengkai Li	University of Texas at Arlington, USA

Danushka Bollegala	The University of Tokyo, Japan
David Cheung	The University of Hong Kong
Donghui Zhang	Microsoft Jim Gray Systems Lab, USA
Fei Xu	Microsoft
Feifei Li	Florida State University, USA
Ge Yu	Northeastern University, China
Guoren Wang	Northeastern University, China
Guozhu Dong	Wright State University, USA
Heng Tao Shen	University of Queensland, Australia
Hiroaki Ohshima	Kyoto University, Japan
Hong Chen	Chinese University of Hong Kong, Hong Kong
Hongzhi Wang	Harbin Industry University, China
Hua Wang	University of Southern Queensland, Australia
Huan Liu	Arizona State University, USA
Hwanjo Yu	Pohang University of Science and Technology, Korea
Jaewoo Kang	Korea University
Jeffrey Yu	Chinese University of Hong Kong
Jianliang Xu	Hong Kong Baptist University, Hong Kong
Jianyong Wang	Tsinghua University, China
Jie Tang	Tsinghua University, China
Jimmy Huang	York University, Canada
Jun Gao	Peking University, China
Ke Wang	Simon Fraser University, Canada
Kenny Zhu	Shanghai Jiao Tong University, China
Lei Chen	Hong Kong University of Science and Technology, Hong Kong
Lei Duan	Sichuan University, China
Lin Li	Wuhan University of Technology, China
Lei Zou	Peking University, China
Lipeow Lim	University of Hawaii at Manoa, USA
Min Wang	HP Lab China
Nick Koudas	University of Toronto, Canada
Ning Jing	National University of Defense Technology, China
Peiquan Jin	University of Science and Technology China, China
Peng Wang	Fudan University, China
Philip Yu	University of Illinois at Chicago, USA
Qiang Ma	Kyoto University, Japan
Qiang Zhu	University of Michigan at Dearborn, USA
Raymond Ng	University of British Columbia, Canada
Ruili Wang	Massey University, New Zealand
Ruoming Jin	Kent State University, USA

Seung-won Hwang	Pohang University of Science and Technology, Korea
Shinsuke Nakajima	Kyoto Sangyo University, Japan
Shuai Ma	University of Edinburgh, UK
Shuigeng Zhou	Fudan University, China
Shuming Shi	Microsoft Research Asia
Tao Li	Florida International University, USA
Tengjiao Wang	Peking University, China
Ting Yu	North Carolina State University, USA
Toshiyuki Amagasa	University of Tsukuba, Japan
Wei Wang	University of New South Wales, Australia
Weiyi Meng	State University of New York at Binghamton, USA
Weizhu Chen	Microsoft Research Asia
Xiaochun Yang	Northeastern University, China
Xiaofeng Meng	Renmin University of China, China
Xiaoyong Du	Renmin University of China, China
Xin (Luna)	Dong AT&T Research, USA
Xingquan Hill Zhu	University of Technology, Sydney
Xintao Wu	University of North Carolina at Charlotte, USA
Xu Yu	Teradata, USA
Xuanjing Huang	Fudan Universtiy, China
Xuemin Lin	University of New South Wales, Australia
Yan Jia	National University of Defense Technology, China
Yanghua Xiao	Fudan Universtiy, China
Yaokai Feng	Kyushu University, Japan
Yi Cai	City University of Hong Kong, Hong Kong
Yi Ke	Hong Kong University of Science and Technology, Hong Kong
Yingzi Jin	University of Tokyo, Japan
Yoshiharu Ishikawa	Nagoya University, Japan
Yuchen Fu	Soochow University, China
Yunjun Gao	Zhejiang University, China
Yuqing Wu	Indiana University at Bloomington, USA
Zhanhuai Li	Northwestern Polytechnical University, China
Zhongfei Zhang	State University of New York at Binghamton, USA
Zhongyuan Wang	Microsoft Research Asia

Table of Contents

Keynotes

Analytics for Info-plosion Including Information Diffusion Studies for the 3.11 Disaster	1
<i>Masaru Kitsuregawa and Masashi Toyoda</i>	
Using the Web for Collaborative Language Learning and Teaching	2
<i>Werner Winiwarter</i>	
Data-Driven Modeling and Analysis of Online Social Networks	3
<i>Divyakant Agrawal, Bassam Bamieh, Ceren Budak, Amr El Abbadi, Andrew Flanagin, and Stacy Patterson</i>	

Session 1A: Query Processing

Efficient Filter Algorithms for Reverse k-Nearest Neighbor Query	18
<i>Shengsheng Wang, Qiannan Lv, Dayou Liu, and Fangming Gu</i>	
Keyword Query Cleaning with Query Logs	31
<i>Lei Gao, Xiaohui Yu, and Yang Liu</i>	
A Self-adaptive Cross-Domain Query Approach on the Deep Web	43
<i>Yingjun Li, Derong Shen, Tiezheng Nie, Ge Yu, Jing Shan, and Kou Yue</i>	

Session 1B: Uncertain Data

SPARQL Query Answering with RDFS Reasoning on Correlated Probabilistic Data	56
<i>Chi-Cheong Szeto, Edward Hung, and Yu Deng</i>	
Probabilistic Threshold Join over Distributed Uncertain Data	68
<i>Lei Deng, Fei Wang, and Benxiong Huang</i>	
Bayesian Classifiers for Positive Unlabeled Learning	81
<i>Jiazhen He, Yang Zhang, Xue Li, and Yong Wang</i>	

Session 1C: Social Media (1)

Measuring Social Tag Confidence: Is It a Good or Bad Tag?	94
<i>Xiwu Gu, Xianbing Wang, Ruixuan Li, Kunmei Wen, Yufei Yang, and Weijun Xiao</i>	

A New Vector Space Model Exploiting Semantic Correlations of Social Annotations for Web Page Clustering 106
Xiwu Gu, Xianbing Wang, Ruixuan Li, Kunmei Wen, Yufei Yang, and Weijun Xiao

A Generalization Based Approach for Anonymizing Weighted Social Network Graphs 118
Xiangyu Liu and Xiaochun Yang

Session 2A: Semantics

Incremental Reasoning over Multiple Ontologies 131
Jing Lu, Xingzhi Sun, Linhao Xu, and Haofen Wang

General-Purpose Ontology Enrichment from the WWW 144
Mohammed Maree, Mohammed Belkhatir, and Saadat M. Alhashmi

QuerySem: Deriving Query Semantics Based on Multiple Ontologies 157
Mohammed Maree, Saadat M. Alhashmi, and Mohammed Belkhatir

Session 2B: Data Mining (1)

Getting Critical Categories of a Data Set 169
Cheqing Jin, Yizhen Zhang, and Aoying Zhou

MFCluster: Mining Maximal Fault-Tolerant Constant Row Biclusters in Microarray Dataset 181
Miao Wang, Xuequn Shang, Miao Miao, Zhanhuai Li, and Wenbin Liu

Expansion Finding for Given Acronyms Using Conditional Random Fields 191
Jie Liu, Jimeng Chen, Tianbi Liu, and Yalou Huang

Session 2C: Social Media (2)

Leveraging Communication Information among Readers for RFID Data Cleaning 201
Tao Jiang, Yingyuan Xiao, Xiaoye Wang, and Yukun Li

Web Article Quality Assessment in Multi-dimensional Space 214
Jingyu Han, Xiong Fu, Kejia Chen, and Chuandong Wang

DRScribe: An Improved Topic-Based Publish-Subscribe System with Dynamic Routing 226
Guohui Li and Sheng Gao

Session 3A: Cloud Data

An Efficient Quad-Tree Based Index Structure for Cloud Data Management	238
<i>Linlin Ding, Baiyou Qiao, Guoren Wang, and Chen Chen</i>	
Efficient Duplicate Detection on Cloud Using a New Signature Scheme	251
<i>Chuitian Rong, Wei Lu, Xiaoyong Du, and Xiao Zhang</i>	
A Secure and Efficient Role-Based Access Policy towards Cryptographic Cloud Storage	264
<i>Cheng Hong, Zhiquan lv, Min Zhang, and Dengguo Feng</i>	

Session 3B: Multimedia Data

Tagging Image by Exploring Weighted Correlation between Visual Features and Tags	277
<i>Xiaoming Zhang, Zhoujun Li, and Yun Long</i>	
Credibility-Oriented Ranking of Multimedia News Based on a Material-Opinion Model	290
<i>Ling Xu, Qiang Ma, and Masatoshi Yoshikawa</i>	
Actions in Still Web Images: Visualization, Detection and Retrieval	302
<i>Piji Li, Jun Ma, and Shuai Gao</i>	

Session 3C: User Models

Social Analytics for Personalization in Work Environments	314
<i>Qihua Wang and Hongxia Jin</i>	
Finding Appropriate Experts for Collaboration	327
<i>Zhenjiang Zhan, Lichun Yang, Shenghua Bao, Dingyi Han, Zhong Su, and Yong Yu</i>	
Location Privacy Protection in the Presence of Users' Preferences	340
<i>Weiwei Ni, Jinwang Zheng, Zhihong Chong, Shan Lu, and Lei Hu</i>	

Session 4A: Data Management

Layered Graph Data Model for Data Management of DataSpace Support Platform	353
<i>Dan Yang, Derong Shen, Tiezheng Nie, Ge Yu, and Yue Kou</i>	
A Study of RDB-Based RDF Data Management Techniques	366
<i>Vahid Jalali, Mo Zhou, and Yuqing Wu</i>	

Requirement-Based Query and Update Scheduling in Real-Time Data Warehouses 379
Fangling Leng, Yubin Bao, Ge Yu, Jingang Shi, and Xiaoyan Cai

Session 4B: Graph Data

Answering Subgraph Queries over Large Graphs 390
Weiguo Zheng, Lei Zou, and Dongyan Zhao

Finding Relevant Papers Based on Citation Relations..... 403
Yicong Liang, Qing Li, and Tieyun Qian

ASAP: Towards Accurate, Stable and Accelerative Penetrating-Rank Estimation on Large Graphs..... 415
Xuefei Li, Weiren Yu, Bo Yang, and Jiajin Le

Session 4C: Name Disambiguation

Efficient Name Disambiguation in Digital Libraries 430
Jia Zhu, Gabriel Fung, and Liwei Wang

A Classification Framework for Disambiguating Web People Search Result Using Feedback..... 442
Ou Jin, Shenghua Bao, Zhong Su, and Yong Yu

Incorporating User Feedback into Name Disambiguation of Scientific Cooperation Network..... 454
Yuhua Li, Aiming Wen, Quan Lin, Ruixuan Li, and Zhengding Lu

Session 5A: Performance

Multi-core vs. I/O Wall: The Approaches to Conquer and Cooperate ... 467
Yansong Zhang, Min Jiao, Zhanwei Wang, Shan Wang, and Xuan Zhou

W-Order Scan: Minimizing Cache Pollution by Application Software Level Cache Management for MMDB..... 480
Yansong Zhang, Min Jiao, Zhanwei Wang, Shan Wang, and Xuan Zhou

Index Structure for Cross-Class Query in Object Deputy Database 493
Yuwei Peng, Hefei Ge, Mao Ding, Zeqian Huang, and Chuanjian Wang

Session 5B: Data Mining (2)

Ensemble Pruning via Base-Classifer Replacement	505
<i>Huaping Guo and Ming Fan</i>	
Efficient Approximate Similarity Search Using Random Projection Learning	517
<i>Peisen Yuan, Chaofeng Sha, Xiaoling Wang, Bin Yang, and Aoying Zhou</i>	
Informed Prediction with Incremental Core-Based Friend Cycle Discovering	530
<i>Yue Wang, Weijing Huang, Wei Chen, Tengjiao Wang, and Dongqing Yang</i>	

Session 5C: Temporal Data

Early Prediction of Temporal Sequences Based on Information Transfer	542
<i>Ning Yang, Jian Peng, Yu Chen, and Changjie Tang</i>	
Mining Event Temporal Boundaries from News Corpora through Evolution Phase Discovery	554
<i>Liang Kong, Rui Yan, Han Jiang, Yan Zhang, Yan Gao, and Li Fu</i>	
Event Detection over Live and Archived Streams	566
<i>Shanglian Peng, Zhanhuai Li, Qiang Li, Qun Chen, Wei Pan, Hailong Liu, and Yanming Nie</i>	

Session 6A: XML

Renda-RX: A Benchmark for Evaluating XML-Relational Database System	578
<i>Xiao Zhang, Kuicheng Liu, Li Zou, Xiaoyong Du, and Shan Wang</i>	
Energy-Conserving Fragment Methods for Skewed XML Data Access in Push-Based Broadcast	590
<i>Jingjing Wu, Peng Liu, Lu Gan, Yongrui Qin, and Weiwei Sun</i>	

Session 6B: Spatial Data

Evaluating Probabilistic Spatial-Range Closest Pairs Queries over Uncertain Objects	602
<i>Mo Chen, Zixi Jia, Yu Gu, Ge Yu, and Chuanwen Li</i>	
Synthesizing Routes for Low Sampling Trajectories with Absorbing Markov Chains	614
<i>Chengxuan Liao, Jiaheng Lu, and Hong Chen</i>	

Approximate Continuous K -Nearest Neighbor Queries for Uncertain
Objects in Road Networks 627
Guohui Li, Ping Fan, and Ling Yuan

Session 6C: Event Detection

Parallel Detection of Temporal Events from Streaming Data 639
Hao Wang, Ling Feng, and Wenwei Xue

Towards Effective Event Detection, Tracking and Summarization on
Microblog Data 652
Rui Long, Haofen Wang, Yuqiang Chen, Ou Jin, and Yong Yu

Author Index 665

Analytics for Info-plosion Including Information Diffusion Studies for the 3.11 Disaster

Masaru Kitsuregawa and Masashi Toyoda

Institute of Industrial Science, The University of Tokyo,
4-6-1 Komaba Meguro-ku, 1538505 Tokyo, Japan
{kitsure, toyoda}@tkl.iis.u-tokyo.ac.jp

Abstract. Information explosion(Info-plosion) is one of the most substantial phenomena in 21st century. Not only the mentions in blogs and twitter, but also the data from various kinds of sensors is becoming to explode considerably. In Japan, we conducted national project, Info-plosion by MEXT(Ministry of Education, Sports, Culture, Science and Technology), Grand Information Voyage by METI(Ministry of Economy, Trade and Industry). Recently we started FIRST project. In this talk, we would like to show how info-plosion analytics, especially sensor analytics, created disruptive services. In addition, we will show the diffusion pattern analysis of twitter and blogs for the 3.11 disaster. On 9.11, we did not have real time media such as microblogging, while such media quite effectively worked on 3.11.

Explosive amount of information is becoming available, especially since around the beginning of 21st century. This causes so called information overload. Although there are such negative problems, we could think that this is a first experience for human being to see such a vast amount of information. Taking advantage of such a totally new opportunity, we believe various disruptive services could be introduced which we have never imagined a decade ago. We launched info-plosion project in 2005. CPS started to get funding since 2009, while its idea was discussed in 2006. IOT, smarter planet, M2M, and Big Data etc. are targetting not necessarily exactly the same objective, but the goals are quite similar. We would like to introduce some of our experiments. We collected Japanese blogs and tweets in March 11th earthquake and tsunami disaster. Although mobile phone service stopped, the Internet was not damaged. Lots of interesting societal movement happened immediately after the earthquake such as refuge place notification, power saving etc. We examined its diffusion pattern. In addition, we also could clarify the difference of role among the media. People are using blog and microblog differently. We will report the role of IT media and its importance.

Using the Web for Collaborative Language Learning and Teaching

Werner Winiwarter

University of Vienna
Research Group Data Analytics and Computing
Universitätsstraße 5, Vienna, A-1010 Austria
`werner.winiwarter@univie.ac.at`

Abstract. For the past few years, the research focus in Computer-Assisted Language Learning has shifted towards learner-centered approaches, in particular through Web-based source material and linguistic resources. Recent findings emphasize the importance of personalizing the learning experience, integrating it seamlessly into everyday activities, and making use of collaborative and social features.

We addressed these issues by developing *COLLIE*, a *Collaborative Language Learning and Instruction Environment*. We have implemented a first prototype for Japanese, which lets the students work with Japanese Web documents and offers appealing visualizations of linguistic information at the character, word, and sentence level. For each user, we store all the relevant information about personal preferences to customize the display of the Web content and about the interaction with *COLLIE* as valuable input for the instructor.

A key component of *COLLIE* is the machine translation system *JETCAT*, which has the big advantage that it learns translation rules automatically from bilingual corpora. The acquisition process is performed by parsing source and target sentences, retrieving word translations of Japanese content words from lexical resources, mapping the content words to corresponding English words through an iterative contextual disambiguation process, and, finally, structurally aligning the two trees based on the word alignment data.

The translation rules used in our formalism can be easily interpreted and understood by language students. Therefore, we created an animated view of the source and target elements of the individual rules in the Japanese and English parse trees to convey a better understanding of the translation process.

The *JETCAT* rule base can be incrementally updated by simply correcting translation results. We provide this feature for the language students so that they can create their own personal translation rule bases and customize *COLLIE* according to their preferences. In addition, we offer the possibility to display a dynamically generated list of the most popular translations produced with the rule bases of the other *COLLIE* users.

We finally report on new directions and activities towards terminology discovery, semantic annotation and retrieval of multimedia documents, community building, mobile language learning, and language learning in collaborative 3D virtual worlds.

Data-Driven Modeling and Analysis of Online Social Networks

Divyakant Agrawal¹, Bassam Bamieh², Ceren Budak¹, Amr El Abbadi¹,
Andrew Flanagin³, and Stacy Patterson²

¹ Department of Computer Science
University of California at Santa Barbara
Santa Barbara, CA 93106, USA

² Department of Mechanical Engineering
University of California at Santa Barbara
Santa Barbara, CA 93106, USA

³ Department of Communication
University of California at Santa Barbara
Santa Barbara, CA 93106, USA

Abstract. With hundreds of millions of users worldwide, social networks provide incredible opportunities for social connection, learning, political and social change, and individual entertainment and enhancement in a wide variety of forms. In light of these notable outcomes, understanding information diffusion over online social networks is a critical research goal. Because many social interactions currently take place in online networks, we now have access to unprecedented amounts of information about social interaction. Prior to the advent of such online networks, investigations about social behavior required resource-intensive activities such as random trials, surveys, and manual data collection to gather even small data sets. Now, massive amounts of information about social networks and social interactions are recorded. This wealth of data can allow us to study social interactions on a scale and at a level of detail that has never before been possible.

We present an integrated approach to information diffusion in online social networks focusing on three key problems: (1) Querying and analysis of online social network datasets; (2) Modeling and analysis of social networks; and (3) Analysis of social media and social interactions in the contemporary media environment. The overarching goals are to generate a greater understanding of social interactions in online networks through data analysis, to develop reliable and scalable models that can predict outcomes of these social processes, and ultimately to create applications that can shape the outcome of these processes. We start by developing and refining models of information diffusion based on real-world data sets. We next address the problem of finding influential users in this data-driven framework. It is equally important to identify techniques that can slow or prevent the spread of misinformation, and hence algorithms are explored to address this question. A third interest is the process by which a social group forms opinions about an idea or product, and we therefore describe preliminary approaches to create models that

accurately capture the opinion formation process in online social networks. While questions relating to the propagation of a single news item or idea are important, these *information campaigns* do not exist in isolation. Therefore, our proposed approach also addresses the interplay of the many information diffusion processes that take place simultaneously in a network and the relative importance of different topics or *trends* over multiple spatial and temporal resolutions.

Keywords: Information propagation, Social Networks, Data Analysis, Sub-modular optimization.

1 Introduction

Internet technologies and online social networks are changing the nature of social interactions. There exist incredible opportunities for learning, social connection, and individual entertainment and enhancement in a wide variety of forms. People now have ready access to almost inconceivably vast information repositories that are increasingly portable, accessible, and interactive in both delivery and formation. Basic human activities have changed as a result, and new possibilities have emerged. For instance, the process by which people locate, organize, and coordinate groups of individuals with shared interests, the number and nature of information and news sources available, and the ability to solicit and share opinions and ideas across myriad topics have all undergone dramatic change as a result of interconnected digital media. Indeed, recent evidence indicates that 45% of users in the U.S. say that the Internet played a crucial or important role in at least one major decision in their lives in the last two years, such as attaining additional career training, helping themselves or someone else with a major illness or medical condition, or making a major investment or financial decision [HR06]. The significant role of online social networks in human interactions motivates our goals of generating a greater understanding of social interactions in online networks through data analysis, the development of reliable models that can predict outcomes of social processes, and ultimately the creation of applications that can shape the outcome of these processes.

This work centers on social processes related to the diffusion of information and opinions in online social networks. We are interested in questions relating to how a single news item or idea propagates through a social network. We also note that such “information campaigns” do not exist in isolation. So while it is important to understand the dynamics of individual campaigns, we also study the interplay of the many information diffusion processes that take place simultaneously in a network and the relative importance of different information topics or *trends* over multiple geographical and temporal resolutions.

Diffusion of Information and Opinions. Given the notable outcomes and the potential applications of information diffusion over online social networks, it is an important research goal to increase our understanding of information diffusion processes. It is also desirable to understand how these processes can be

modified to achieve desired objectives. We describe and refine models of information diffusion for online social networks based on analysis of real-world data sets. The scalable, flexible identification of influential users or opinion leaders for specific topics is crucial to ensuring that information reliably reaches a large audience, and so we propose to develop algorithms for finding influential users in this data-driven framework. It is equally important to identify techniques that can slow or prevent the spread of misinformation, and we create models and algorithms to address this question. We are also interested in the processes by which a social group forms opinions about an idea or product. We create models that accurately capture the opinion formation process in online social networks and develop scalable algorithms and techniques for external intervention that can alter those opinions.

Information Trend Analysis. In light of an online environment where there is an increasing number of sources of information, understanding information trends is critical. For example, data showing that there is sustained and substantial interest in a particular political issue locally, but not regionally, nationally, or internationally, can signal opportunities for political organization around that topic, political party proclivities in a geographic area, or even where best to focus fund-raising efforts. The relative trendiness of a message or topic, such as when interest in it suddenly spikes, can be more important than the overall popularity of a topic. Finally, underlying trends that illustrate relationships among entities can be used to highlight information that is otherwise obscure. For example, strangers independently confirming the importance of a certain piece of information (such as a user's feedback rating on eBay) are better indicators of its veracity than are a tightly-connected cluster of information sources reporting the same information in common. Trending tools can thus help to discern these and other distinctions. We are interested in developing tools to discover many types of trends at multiple spatio-temporal resolutions.

In the following sections, we detail our research questions, propose solution techniques, and highlight open problems for further investigation.

2 Information Diffusion

Social networks have already emerged as a significant medium for the widespread distribution of news and instructions in mass convergence events such as the 2008 U.S. Presidential Election [HP09], the 2009 Presidential election in Iran [Gro09], and emergencies like the landfall of Hurricanes Ike and Gustav in the fall of 2008 [HP09]. Due to the open nature of social networks, it is not only possible to spread factual information, but also to create misinformation campaigns that can discourage or disrupt correct response in these situations. For instance, Twitter has served as forum for spreading both useful information and false rumors about the H1N1 pandemic which altered users' attitudes about the safety and efficacy of the flu vaccine [Far09].

Given the notable outcomes and the potential applications of information diffusion over online social networks, it is an important research goal to increase

our understanding of information diffusion processes and to study how these processes can be modified to achieve desired objectives. We have developed preliminary models of information and opinion diffusion for online social networks. First, we address the question of how to formalize a loosely specified model of social diffusion based on data analysis in order to adapt this model to social interactions in the blogosphere. We then consider the problem of limiting the reach of an information campaign and highlight preliminary results and proposed extensions.

2.1 Influence Maximization: The Law of the Few

The identification of *influential users* or *opinion leaders* in a social network is a problem that has received a significant amount of attention in recent research [KKT03, CYY09, LKG⁺07, KSNM09, WCSX10]. Informally, we define an *information campaign* as the process by which a news item or idea spreads from the initiators of the campaign, *i.e.* the users who first learn the news, throughout the social network. This initial set of users is denoted by the set A , and the set of users who eventually receive this news is the *influence set* of A , denoted $IF(A)$. In the *influence maximization problem*, given a model of how information diffuses in a social network, the objective is to select a set of users A of size k who are to be the initial recipients of the information (through some offline means), so that the size of $IF(A)$ is maximized [DR01, RD02].

With an efficient, robust solution to this problem, it would be possible to ensure the widespread dissemination of important information in a social network. Early works relied on heuristics such as node degree and distance centrality [WF94] to select the set A . More recently, several investigators have considered probabilistic models of information diffusion such as the *Independent Cascade* [GLM01] and *Linear Threshold* [Gra78]. The problem of finding the optimal initiator set in this model is NP-hard, but there is a polynomial-time greedy algorithm that yields a result that is within $1 - 1/e$ of optimal [KKT03]. Work has been done on improving the performance of greedy algorithms for influence maximization [CYY09, LKG⁺07, KSNM09, WCSX10], but the scalability of these algorithms, in the context of social networks spanning millions of users, remains a significant challenge.

We summarize our recent work [BAEA10] on the influential users problem using a different model of diffusion based on the theories introduced in the popular book “The Tipping Point” by Malcolm Gladwell [Gla02]. The main idea of “The Tipping Point” is the crucial roles of three types of “fascinating” people that the author calls *mavens*, *connectors* and *salesmen* on the effectiveness of an information campaign. These people are claimed to “play a critical role in the word-of-mouth epidemics that dictate our tastes, trends and fashions”. We study those three types of people or *actors* and a fourth type of interesting actor that we call a *translator*.

The first type of actor introduced by Gladwell is the connector. In terms of a social network graph, we define a connector to be a node that has high degree centrality. The second type of important actor in information propagation

is the maven. The word “maven” comes from Yiddish and means one who accumulates knowledge. Gladwell lists three important characteristics for mavens: 1) they seek new knowledge, 2) they share the knowledge they acquire with others and 3) an individual hearing something from a maven is very likely to believe the correctness and importance of this piece of information. Translating those features into graph theory, we define mavens to be nodes that start a large number of cascades (they are the original source of new information) and have high influence on their neighbors. The third kind of actor that Gladwell introduces is the salesman, a person with high charisma who can sell ideas to almost anyone since he never gives up. We define a salesman to be a node that has a large number of trials to activate its neighbors for cascades that the node itself is a part of. We also study another class of actors referred to as *translators*. These actors act as bridges or translators among different communities and therefore have the power of changing the context in which to present an idea. In order to identify the translators in the blogosphere, first we need to detect the communities. Different from many community detection research [Cla05, PSL90, New03, New06, GL08, BGMI05, ZWZ07, Gre07], we define communities based on the existence of flow of influence between nodes rather than relying solely on the structure of the graph. Using this definition, we detect overlapping communities using the algorithm presented in [BGMI05] and define translators as the nodes that belong to the most number of communities.

Weblogs have become a predominant way of sharing data online. The blogosphere has considerable influence on how people make decisions in areas such as politics or technology [AG05]. We used the August-October 2008 Memetracker data that contains timestamped phrase and link information for news media articles and blog posts from different blogs and news websites. The data set consists of 53,744,349 posts and 2,115,449 sources of information (blogs, news media and sources that reside outside the blogosphere) and 744,189 cascades. Using the methods formalized above, we identify the mavens, salesmen, connectors and translators of the blogosphere and study their effect on the success of cascades. Our initial results are quite promising in that they indicate that algorithms for finding the best method of reaching out to certain actors, rather than the entire network, can be a good heuristic to impact influence in social networks. The types of actors identified can also be used to augment the current models of diffusion to capture real world behavior. As part of future work, we also plan to augment our analysis on the intermediaries to investigate if there exists an optimal timing to reach out to a connector, maven, salesman or translator. Are these actors more useful if they adopt and advocate a cascade early or later on? We analyzed the blogosphere data to investigate the validity of the heuristics introduced but the same heuristics can indeed be evaluated on other social networks. We believe that different social networks provide different types of interactions, which means that certain actors, while not so significant in some networks, can be highly influential in others.

2.2 Limiting Diffusion of Misinformation

While a substantial amount of research has been done in the context of influence maximization, a problem that has not received much attention is that of limiting the influence of a malicious or incorrect information campaign. One strategy to deal with a misinformation campaign is to limit the number of users who are willing to accept and spread this misinformation. In this section, we present preliminary work on techniques to limit the number of users who participate in an information campaign. We call this problem the *influence limitation problem*.

In this context, we consider the social network as an edge-weighted graph. The nodes represent users of the network and edges represent a relationship between these users. Edges can be undirected, indicating that the relationship is symmetric (users are friends with each other), or they can be directed, indicating one-way communication such as a publisher/subscriber relationship (follower relationship in Twitter). We use edge weights to quantify the “influence” that one node has upon another node; the weight of edge $e_{i,j}$ is an indicator of the likelihood that node j will agree with and pass on information it receives from node i . Nodes that have adopted the idea or information are called *active* and those that have not are *inactive*.

We consider two different information diffusion models, both similar to the Independent Cascade Model (ICM). In ICM, the information diffusion process takes place in discrete rounds. When node i is first activated in round r , it tries to activate each inactive neighbor j ; it succeeds with probability $p(i,j)$. Whether or not it succeeds, node i does not try to activate anyone in subsequent rounds. In order to capture simultaneous spread of two cascades (the initial misinformation campaign and the limiting campaign), we introduced two extensions to ICM called the Multi-Campaign Independent Cascade Model (MCICM) and Campaign-Oblivious Independent Cascade Model (COICM) [BAEA11a]. We omit the details of these models due to space limitations but note that they are similar to a number of other models in literature [BKS07, DGM06, CNWvZ07, KOW08].

Our objective is to minimize the number of people affected by the misinformation campaign by “vaccinating” users through a *limiting campaign*. Let the campaign that is to be minimized be campaign C and the initial set of nodes activated in campaign C be A_C . The limiting campaign is called campaign L and the initial activation set is A_L . For simplicity, we assume that a user can be active in only one campaign, *i.e.*, once the user has decided on a side, he will not change his mind. Given a network and a diffusion model, suppose that a campaign that is spreading bad information is detected r rounds after it began. Given a budget l , select l individuals for initial activation with the competing information such that the expected number of nodes eventually activated in campaign C is minimized. Let $IF(A_C)$ denote the influence set of campaign C without the presence of campaign L , *i.e.* the set of nodes that would accept campaign C if there were no limiting campaign. We define the function $\pi(A_L)$ to be the size of the subset of $IF(A_C)$ that campaign L prevents from adopting

campaign C . Then, the influence limitation problem is equivalent to selecting A_L such that the expectation of $\pi(A_L)$ is maximized.

- 1: Initialize A_L to \emptyset
- 2: **for** $i = 1$ to l **do**
- 3: Choose node i that maximizes $\pi(A_L \cup \{i\}) - \pi(A_L)$;
- 4: Set $A_L \leftarrow A_L \cup \{i\}$;

Fig. 1. Greedy algorithm to select the set for initial activation in the limiting campaign

a falsehood). We refer to this notion as *high-effectiveness property*. In a recent study [BAEA11a], we have shown that $\pi(A_L)$ is a monotone, sub-modular function for this setting. We have also proved the same result in the context of Campaign-Oblivious Independent Cascade Model, even without the *high-effectiveness property*. Therefore, for both diffusion models, the greedy algorithm given in Figure 1 yields an A_L for which $\pi(A_L)$ is within $1 - 1/e$ of optimal [BAEA11a].

We now outline a potential solution to a simplified version of this problem. We assume that there is only a single source of information for campaign C , meaning $|A_C| = 1$ and that information diffusion follows the Multi-Campaign Independent Cascade Model. Finally, we assume that L is accepted by users with probability 1 (it may be much easier to convince a user of the truth than

3 Opinion Dynamics

Another process that is of interest in social networks research is the process by which a group of individuals in a social network form an opinion about a piece of information or an idea and how interpersonal influence affects the opinion formation process. This process is known as *opinion dynamics*. While this process shares some similarities with the diffusion of an information campaign, it differs in the main respect that the individual opinions evolve over time, continuously changing in response to interactions with other individuals and possibly external inputs.

Research on opinion formation in social groups predates the advent of online social networks by decades, and several formal mathematical models of the opinion formation process have been proposed [Fre56, DeG74, Leh75, EJ99, HK02]. These early works are based on the assumption that all individuals interact with all friends simultaneously in a synchronized fashion. In online social networks, however, individuals are spread out across space and time and they interact with different communities and friends at different times and with different frequencies. We investigate how these previously proposed models of opinion dynamics can be augmented to incorporate the asynchronous nature of social interactions that arise in online social networks.

We first briefly review the general opinion dynamics model. The social network is modeled as a graph $G = (V, E)$ where the vertices represent users or *agents*, with $|V| = n$, and the edges represent relationships between users, with

$|E| = m$. The graph may be directed or undirected. A directed graph is used to model networks where relationships are not symmetric, for example, the follower relationship in Twitter. An undirected graph models a network with symmetric relationships such as the friend relationship in Facebook. We say that agent i is a *neighbor* of agent j if $(i, j) \in E$. Each individual i has an initial opinion $x_i(0)$. The opinion is assumed to be a real number, for example a numerical representation of the individual's support for an issue. The agreement process takes place in discrete rounds. In each round, each individual updates his opinion based on information exchanged along edges in the network graph, taking a weighted average of his own opinion and the opinions of his neighbors. Let w_{ij} be the weight that agent i places on the opinion of agent j with the normalization requirement that $\sum_{j=1}^n w_{ij} = 1$. In each round, individual i updates his opinion as follows:

$$x_i(t+1) = w_{i1}x_1(t) + w_{i2}x_2(t) + \dots + w_{in}x_n(t),$$

where $w_{ij} > 0$ only if $(i, j) \in E$. We note that this formulation admits the possibility that $w_{ij} = 0$ even if $(i, j) \in E$, meaning i does not place any weight on the opinion of j even though they are neighbors in the network. The evolution of all of the opinions, called the *opinion profile*, is captured by the following recursion:

$$x(t+1) = W(t, x(t))x(t).$$

$x(t)$ is the n -vector of opinions at round t , and $W(t, x(t))$ is an $n \times n$ matrix that gives the edge weights, the interpersonal influence, for round t . This general model allows for the possibility that these edge weights may change over time and may depend on the current opinion profile.

In the following sections, we restrict our discussion to the *classical model* of opinion dynamics that was proposed by De Groot [DeG74] and Lehrer [Leh75], where the edge weights are assumed to remain constant throughout the opinion formation process. The process is given by the recursion

$$x(t+1) = Wx(t) \tag{1}$$

where W is an $n \times n$ matrix.

Due to the simple form of this model, it is possible to analyze properties of the opinion formation process and predict the outcome by examining the W matrix. In particular, it can be shown that (1) Individuals reach agreement if and only if $|\lambda_2(W)| < 1$, where $\lambda_2(W)$ is the second largest eigenvalue of W by magnitude; (2) If $|\lambda_2(W)| < 1$ and W is symmetric, the consensus value is the average of the initial opinions. If W is not symmetric, the consensus value is a weighted average of the initial opinions; and (3) If agreement will occur, the number of rounds required for individuals to be within ϵ of the consensus value is $\log \epsilon / \log(\lambda_2(W))$. We next describe several extensions to the classical opinion dynamics that capture the types of social interactions exhibited in online networks.

It has been observed that users interact frequently with only a small subset of their neighbors and that communication is infrequent along many edges

[WBS+09]. It is reasonable to expect that the frequency of interaction will have a large impact on the evolution of opinions in online social networks. To capture the notion of interaction frequency, we associate a (unique) probability of communication p_{ij} with each edge $(i, j) \in E$. The value p_{ij} is the probability that agents i and j will communicate in each round. The evolution of the opinion profile with probabilistic interactions is given by the following recursion,

$$x(t+1) = \left(I - \sum_{(i,j) \in E} \delta_{ij}(t) w_{ij} L_{ij} \right) x(t) \quad (2)$$

where $\delta_{ij}(t)$ are independent Bernoulli random variables with

$$\delta_{ij}(t) := \begin{cases} 1 & \text{with probability } p_{ij} \\ 0 & \text{with probability } 1 - p_{ij}. \end{cases}$$

This model has been adapted from the model for multi-agent consensus in stochastic networks [PBE10]. Each L_{ij} is the weighted Laplacian matrix of the graph $G_{ij} = (V, \{(i, j)\})$, the graph that contains the same n vertices as the original graph G and the single edge (i, j) . When $\delta_{ij} = 1$, agents i and j exchange information just as they did in the classical model. When $\delta_{ij} = 0$, agents i and j do not communicate.

In our recent work [PB10, PBE10], we have derived a matrix-valued, Lyapunov-like operator $\mathcal{W}(\cdot)$ that describes the evolution of the covariance of the opinion profile in this stochastic model of opinion dynamics and we have provided analysis similar to the well-known results for classical opinion dynamics.

4 Information Trend Analysis

Social networks provide a large-scale information infrastructure for people to discuss and exchange ideas about variety of topics. Detecting trends of such topics is of significant interest for many reasons. For one, it can be used to detect emergent behavior in the network, for instance a sudden increase in the number of people talking about explosives or biological warfare. Information trends can also be viewed as a reflection of societal concerns or even as a consensus of collective decision making. Understanding how a community *decides* that a topic is trendy can help us better understand how ad-hoc communities are formed and how decisions are made in such communities. In general, constructing “useful” trend definitions and providing scalable solutions that detect such trends will contribute towards a better understanding of human interactions in the context of social media.

Before we study the problem of finding trendy topics in a social network, we first need to develop a clear definition of “trendiness”. Assume users of a network can choose to (or not to) broadcast their opinions about various topics at any point in time. Assume further that we can abstract away what the topic is from what a user broadcasts. A simple definition of trendy topics can be the

frequent items throughout the entire history of user broadcasts. The problem, defined this way, is simply to find the frequent items in a stream of data, also referred to as *heavy hitters*. The *frequent elements problem* is well studied and several scalable, online solutions have been proposed [CCFC02, CM05, MAE06, MM02, DLOM02]. While the heavy hitters view trend definition is compelling because of the existence of scalable algorithms, this simple definition overlooks various important aspects such as the spatio-temporal dimensions of human interaction and the network structure and its effect on the emergence of trends. In the following, we explore structural trend analysis as an example that depend on structural connections between the users who are broadcasting. Information trends at the granularity of spatio-temporal dimensions remains future work.

Assuming that information diffusion on a social network is a substantial part of the process that creates the information trends, properties that are defined in the context of the network structure are of significant interest. For example, consider a group of friends in a large social network like Facebook discussing an attack. Detecting this new interest of this specific group on “attacks” can be of great importance. Note that especially for cyber attacks, those people do not necessarily need to be in the same geographical region, and in some instances, this geographic information is not even available. In essence, a structural trend is a topic that is identified as “hot” within structural subgroups of the network. The challenges are to formally define the notions of a structural subgroup and to develop techniques to detect *structural trends*.

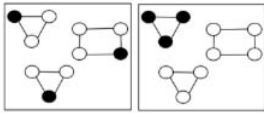


Fig. 2. Black nodes represent nodes talking about topic x , whereas white nodes represent the nodes that are not

nodes are people who are not talking about it. Even though both graphs have the same number of people talking about this topic, in the graph on the right, the people talking about the issue are a part of a more clustered subgraph, giving the topic a higher structural significance. The number of pairs of users talking about the topic in the graph on the left is 0 whereas the pair count is 3 for the graph on the right. The detection of structural trends is harder to solve than the traditional definition of trends since in the traditional setting a counter can be updated on each new data item without dependence on any future or past items. This is not the case for the correlated trends. Value addition of a tuple $\langle n_i, T_j \rangle$ to topic T_j , where n_i represents a node in the graph and T_j is an arbitrary topic depends on other tuples involving T_j and neighbors of n_i . The detection of structural trends calls for new, graph-oriented solutions.

As a starting point, we consider the problem of identifying the number of connected pairs of users in a social network that are discussing a specific topic. We refer to this as detecting *correlated trends*. Consider the two graphs in Figure 2. The black nodes correspond to people that are talking about a specific topic and white

The exact solution for *correlated trends* that requires keeping track of all topics for all nodes is not scalable for large networks so we need to explore approximation algorithms. Here we describe possible solutions: (i) Use the activity level and degree of a node as a heuristic to limit the number of nodes monitored per topic; (ii) Only monitor those topics that are frequent in the traditional sense and for such topics find the order of their importance w.r.t. correlated trendiness; (iii) As demonstrated by Mislove et al. [MMG⁺07], there is a small subset of nodes in social networks without which the network would be a collection of many small disconnected subgraphs. This property can be exploited to partition the graph into smaller sub-graphs and apply the counting algorithms in parallel. Query processing requires periodically merging the counts from multiple sub-partitions with the counts from highly connected nodes. This approach is highly scalable in the MapReduce paradigm [DG08].

Another solution we propose to evaluate involves a semi-streaming approximation algorithm. We essentially reduce the problem of evaluating the importance of each topic with respect to the correlated trendiness notion to a problem of counting local triangles in a graph, *i.e.* counting the number of triangles incident to every node $n \in N$ in the graph. Consider a social network graph $G = (N, E)$, a set of all topics T and stream of node-topic tuples S , where each tuple is in the form: $\langle n_i, T_j \rangle$ s.t. $n_i \in N$ and $T_j \in T$. Let us create a graph $G' = (N', E')$ s.t. $N' = N \cup T$ and $E' = \{(u, v) | (u, v) \in E \wedge (u, v) \in S\}$. The number of connected pairs of users in the social network G that are discussing a specific topic T_j is simply the number of triangles incident to node T_j in G' . Using approximation algorithms based on the idea of min-wise independent permutations similar to [BBCG08], we are able to provide a solution using $O(|N'|)$ space in main memory and performing $O(\log|N'|)$ sequential passes over E' . Note that the naive solution would need $O(|N'| \cdot |T|)$ space in main memory.

Alternatively, we can define a structural trend be the other extreme, where we are interested in the number of *unrelated* people interested in a specific topic and in trends that results from these unrelated people. We call these *uncorrelated trends*. Going back to our example of two graphs in Figure 2, for the graph on the left this count will be 3, whereas it will be only 1 for the graph on the right. This definition of trendiness can be used capture the notion of the *trustworthiness* of a trend. In this case trendiness of a topic will not be biased by a discussion amongst a small clustered group. We note that we have only considered two extremes of structural trends. Identifying alternative definitions of structural trends that will span the entire spectrum remains future work [BAEA11b].

5 Concluding Remarks

Internet technologies and online social networks are changing the nature of social interactions and user behaviors. Recent evidence indicates that 45% of users in the U.S. state that the Internet played a crucial or important role in at least one major decision in their lives in the last two years, such as attaining additional career training, helping themselves or someone else with a major illness or medical

condition, or making a major investment or financial decision [HR06]. The significant role of online social networks in human interactions motivates our goals of generating a greater understanding of social interactions in online networks through data analysis, the development of reliable models that can predict outcomes of social processes, and ultimately the creation of applications that can shape the outcome of these processes. In this paper, we have presented a preliminary formulation of modeling and analyzing *information diffusion*, *opinion dynamics*, and *information trends* in online social networks.

Acknowledgments. This work was partially supported by the NSF under grant IIS-1135389.

References

- [AG05] Adamic, L.A., Glance, N.: The political blogosphere and the 2004 u.s. election: divided they blog. In: LinkKDD 2005: Proceedings of the 3rd International Workshop on Link Discovery, pp. 36–43 (2005)
- [BAEA10] Budak, C., Agrawal, D., El Abbadi, A.: Where the blogs tip: connectors, mavens, salesmen and translators of the blogosphere. In: SIGKDD Workshop on Social Media Analytics (2010)
- [BAEA11a] Budak, C., Agrawal, D., El Abbadi, A.: Limiting the spread of misinformation in social networks. In: Proceedings of the 20th International Conference on World Wide Web, WWW 2011, pp. 665–674 (2011)
- [BAEA11b] Budak, C., Agrawal, D., El Abbadi, A.: Structural trend analysis for online social networks. In: VLDB 2011 (2011)
- [BBCG08] Becchetti, L., Boldi, P., Castillo, C., Gionis, A.: Efficient semi-streaming algorithms for local triangle counting in massive graphs. In: KDD 2008: Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 16–24. ACM, New York (2008)
- [BGMI05] Baumes, J., Goldberg, M., Magdon-Ismael, M.: Efficient identification of overlapping communities. In: IEEE International Conference on Intelligence and Security Informatics (ISI), pp. 27–36 (2005)
- [BKS07] Bharathi, S., Kempe, D., Salek, M.: Competitive influence maximization in social networks. In: Deng, X., Graham, F.C. (eds.) WINE 2007. LNCS, vol. 4858, pp. 306–311. Springer, Heidelberg (2007)
- [CCFC02] Charikar, M., Chen, K., Farach-Colton, M.: Finding frequent items in data streams. In: Widmayer, P., Triguero, F., Morales, R., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 693–703. Springer, Heidelberg (2002)
- [Cla05] Clauset, A.: Finding local community structure in networks. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)* 72(2), 026132 (2005)
- [CM05] Cormode, G., Muthukrishnan, S.: What’s Hot and What’s Not: Tracking Most Frequent Items Dynamically. *ACM Trans. Database Syst.* 30(1), 249–278 (2005)

- [CNWvZ07] Carnes, T., Nagarajan, C., Wild, S.M., van Zuylen, A.: Maximizing influence in a competitive social network: a follower's perspective. In: ICEC 2007: Proceedings of the Ninth International Conference on Electronic Commerce, pp. 351–360 (2007)
- [CYY09] Chen, W., Wang, Y., Yang, S.: Efficient influence maximization in social networks. In: Proceedings of the 15th ACM International Conference on Knowledge Discovery and Data Mining, pp. 199–208 (2009)
- [DeG74] DeGroot, M.H.: Reaching a consensus. *Journal of the American Statistical Association* 69, 118–121 (1974)
- [DG08] Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. *ACM Commun.* 51(1), 107–113 (2008)
- [DGM06] Dubey, P., Garg, R., De Meyer, B.: Competing for customers in a social network. Cowles Foundation Discussion Papers 1591, Cowles Foundation, Yale University (November 2006)
- [DLOM02] Demaine, E.D., López-Ortiz, A., Munro, J.I.: Frequency estimation of internet packet streams with limited space. In: Möhring, R.H., Raman, R. (eds.) ESA 2002. LNCS, vol. 2461, pp. 348–360. Springer, Heidelberg (2002)
- [DR01] Domingos, P., Richardson, M.: Mining the network value of customers. In: Proceedings of the 7th ACM International Conference on Knowledge Discovery and Data Mining, pp. 57–66 (2001)
- [Far09] Farrell, M.B.: Schwarzenegger tweets about swine flu. so does everyone else. *Christian Science Monitor* (April 2009)
- [FJ99] Friedkin, N.E., Johnsen, E.C.: Social influence networks and opinion change. *Advances in Group Processes* 16, 1–29 (1999)
- [Fre56] French, J.R.P.: A formal theory of social power. *Psychological Review* 63, 181–194 (1956)
- [GL08] Ghosh, R., Lerman, K.: Community Detection Using a Measure of Global Influence. In: Giles, L., Smith, M., Yen, J., Zhang, H. (eds.) SNAKDD 2008. LNCS, vol. 5498, pp. 20–35. Springer, Heidelberg (2010)
- [Gla02] Gladwell, M.: *The Tipping Point: How Little Things Can Make a Big Difference*. Back Bay Books (January 2002)
- [GLM01] Goldenberg, J., Libai, B., Muller, E.: Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing Letters* 12(3), 209–221 (2001)
- [Gra78] Granovetter, M.: Threshold models of collective behavior. *American Journal of Sociology* 83(6), 1420–1443 (1978)
- [Gre07] Gregory, S.: An algorithm to find overlapping community structure in networks. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenić, D., Skowron, A. (eds.) PKDD 2007. LNCS (LNAI), vol. 4702, pp. 91–102. Springer, Heidelberg (2007)
- [Gro09] Grossman, L.: Iran protests: Twitter, the medium of the movement. *Time* (online) (June 2009)
- [HK02] Hegselmann, R., Krause, U.: Opinion dynamics and bounded confidence models, analysis, and simulation. *Journal of Artificial Societies and Social Simulation* 5(3) (2002)
- [HP09] Hughes, A.L., Palen, L.: Twitter adoption and use in mass convergence and emergency events. In: Proceedings of the 6th International Information Systems for Crisis Response and Management Conference (2009)

- [HR06] Horrigan, J., Rainie, L.: When facing a tough decision, 60 million americans now seek the internet's help: The internet's growing role in life's major moments (2006), <http://pewresearch.org/obdeck/?0bDeckID=19> (retrieved October 13, 2006)
- [KKT03] Kempe, D., Kleinberg, J.M., Tardos, É.: Maximizing the spread of influence through a social network. In: Proceedings of the Ninth ACM International Conference on Knowledge Discovery and Data Mining, pp. 137–146 (2003)
- [KOW08] Kostka, J., Oswald, Y.A., Wattenhofer, R.: Word of Mouth: Rumor Dissemination in Social Networks. In: 15th International Colloquium on Structural Information and Communication Complexity (SIROCCO) (June 2008)
- [KSNM09] Kimura, M., Saito, K., Nakano, R., Motoda, H.: Finding Influential Nodes in a Social Network from Information Diffusion Data. In: Social Computing and Behavioral Modeling. Springer, US (2009)
- [Leh75] Lehrer, K.: Social consensus and rational agnoiology. *Synthese* 31(1), 141–160 (1975)
- [LKG⁺07] Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., Glance, N.: Cost-effective outbreak detection in networks. In: Proceedings of the 13th ACM International Conference on Knowledge Discovery and Data Mining, pp. 420–429 (2007)
- [MAE06] Metwally, A., Agrawal, D., El Abbadi, A.: An integrated efficient solution for computing frequent and top-k elements in data streams. *ACM Trans. Database Syst.* 31(3), 1095–1133 (2006)
- [MM02] Manku, G.S., Motwani, R.: Approximate frequency counts over data streams. In: Proc. 28th Int. Conf. on Very Large Data Bases, pp. 346–357 (2002)
- [MMG⁺07] Mislove, A., Marcon, M., Gummadi, K.P., Druschel, P., Bhattacharjee, B.: Measurement and analysis of online social networks. In: Proc. 7th ACM SIGCOMM Conf. on Internet Measurement, pp. 29–42 (2007)
- [New03] Newman, M.E.J.: Fast algorithm for detecting community structure in networks. *Physical Review E* 69 (September 2003)
- [New06] Newman, M.E.J.: Modularity and community structure in networks. *Proceedings of the National Academy of Sciences* 103(23), 8577–8582 (2006)
- [PB10] Patterson, S., Bamieh, B.: Interaction-driven opinion dynamics in online social networks. In: SIGKDD Workshop on Social Media Analytics (2010)
- [PBE10] Patterson, S., Bamieh, B., El Abbadi, A.: Convergence rates of distributed average consensus with stochastic link failures. *IEEE Transactions on Automatic Control* 55(4), 880–892 (2010)
- [PSL90] Pothen, A., Simon, H.D., Liou, K.-P.: Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. Matrix Anal. Appl.* 11(3), 430–452 (1990)
- [RD02] Richardson, M., Domingos, P.: Mining knowledge-sharing sites for viral marketing. In: Proceedings of the 8th ACM International Conference on Knowledge Discovery and Data Mining, pp. 61–70 (2002)
- [WBS⁺09] Wilson, C., Boe, B., Sala, A., Puttaswamy, K.P.N., Zhao, B.Y.: User interactions in social networks and their implications. In: Proc. 4th ACM European Conference on Computer Systems, pp. 205–218 (2009)

- [WCSX10] Wang, Y., Cong, G., Song, G., Xie, K.: Community-based greedy algorithm for mining top-k influential nodes in mobile social networks. In: Proceedings of the 16th ACM International Conference on Knowledge Discovery and Data Mining (2010)
- [WF94] Wasserman, S., Faust, K.: Social Network Analysis. Cambridge University Press, Cambridge (1994)
- [ZWZ07] Zhang, S., Wang, R.S., Zhang, X.S.: Identification of overlapping community structure in complex networks using fuzzy c-means clustering. *Physica A: Statistical Mechanics and its Applications* 374(1), 483–490 (2007)

Efficient Filter Algorithms for Reverse k -Nearest Neighbor Query

Shengsheng Wang, Qiannan Lv, Dayou Liu, and Fangming Gu

College of Computer Science and Technology, Jilin University
130012 Changchun, China
lvqiannan@163.com, {wss, liudy}@jlu.edu.cn,
fangminggu77@gmail.com

Abstract. Reverse k -Nearest Neighbor ($RkNN$) Queries have got considerable attentions over the recent years. Most state of the art methods use the two-step (filter-refinement) $RkNN$ processing. However, for a large k , the amount of calculation becomes very heavy, especially in the filter step. This is not acceptable for most mobile devices. A new filter strategy called BRC is proposed to deal with the filter step for $RkNN$ queries. There are two pruning heuristics in BRC. The experiments show that the processing time of BRC is still acceptable for most mobile devices when k is large. And we extend the BRC to the continuous $RkNN$ queries.

Keywords: Reverse k -Nearest Neighbor, Continuous Reverse k -Nearest Neighbor, Spatial Database.

1 Introduction

Reverse k Nearest Neighbor ($RkNN$) processing has got considerable attentions over the recent years. $RkNN$ plays an important role in decision support, resource allocation, data mining, profile-based marketing and other important areas. One object may request some services from its nearest neighbor. It is important for an object to know how many objects it is supposed to serve. The objects which request services could be soldiers in a battle field, tourists in dangerous environments, etc. The services providing objects could be aid workers, etc.

Example 1: In a multi-player computer game, players often shoot their nearest neighbors. In order to dodge the fires from others, players need $RkNN$ queries.

Example 2: Many road accidents occur in the city. Aid workers or related law-enforcement staff should use $RkNN$ queries to handle the accidents in the most efficient way.

Given a multidimensional object set P and a query point $q \notin P$, a $RkNN$ query retrieves all the points $p \in P$ which have q as one of their k nearest neighbors (we assume Euclidean distance). $p \in kNN(q)$ does not necessarily imply $p \in RkNN(q)$ and vice versa. Fig.1 shows object set $P = \{p_1, p_2, p_3, p_4, p_5\}$ and the query point q .

The red circle is the vicinity circle of q (centered at q with radius equal to the distance between q and $NN(q)$). The black circles are the vicinity circles of p_1 - p_5 . Because q lies in the vicinity circles of p_4 and p_5 , $RNN(q) = \{p_4, p_5\}$.

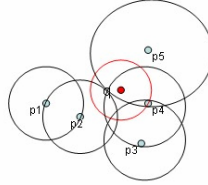


Fig. 1. NN and RNN examples

Early $RkNN$ works fail to solve at least one of the following problems: they (i) do not support arbitrary values of k , (ii) cannot deal efficiently with database updates, (iii) are applicable only to 2D dataset but not to higher dimensionality and (iv) retrieve only approximate results. Tao et al. [1] proposed the TPL method which successfully solves these problems. However, when the value of k is large, the amount of calculation becomes very heavy. The TPL method is not suitable for mobile devices such as mobile phone. It is necessary to provide a new method to reduce the calculation. Motivated by the above reasons, we propose a new solution to $RkNN$ queries. Our experiments will show that the proposed method is superior to TPL. Then we extend our methodology to continuous reverse k nearest neighbor (C- $RkNN$) [2] queries.

The rest of the paper is organized as follows. Section 2 surveys related work on kNN and $RkNN$ search. Section 3 presents filter step for $RkNN$ with static query point. Section 4 discusses filter step for continuous $RkNN$ processing. Section 5 contains an experimental evaluation that demonstrates the superiority of the proposed methods over the previous algorithms. Section 6 concludes the paper.

2 Related Works

2000, Korn et al. [3] introduced RNN queries and gave a processing method which relied on pre-processing. Yang et al. [4] and Maheshwari et al. [5] respectively improved the method in [3]. But the pre-computing is necessary in the above methods [3-5]. And their updates can not be performed efficiently, because each insertion or deletion may affect the vicinity circles of several objects. Stanoi et al. [6] presented the six-regions pruning method. This method does not need any pre-computation. This method is only applicable to 2D-space. It was extended to arbitrary values of k by Tao et al [1]. Singh et al.[7] used a intuitive idea that the NNs of objects are likely to be RNNs. The major shortcoming of the method is that it may incur false misses. Tao et al. [1] used the property of perpendicular bisectors to answer $RkNN$ queries, called TPL. Wu et al. [8] proposed an algorithm FINCH for $RkNN$ queries in 2D-space. Instead of using the bisectors to prune the objects, they use a convex polygon obtained from the intersection of the bisectors.

Benetis et al. [2] presented the first continuous RNN algorithm. Their method is based on the extension of literature [6]. In each sub-space, the set of candidates is continuously maintained, and each candidate's k NNs are also continuously maintained to do continuous refinement. TPL-pruning and Six-regions pruning can also be used in C-RkNN queries [9,10].

3 The Filter Step for RkNN Query with Static Query Point

The queries based on R-tree indexing generally adopt the two-step (filter-refinement) processing. The filter step eliminates the false objects to obtain a set of candidate results. The refinement step verifies these candidate results to acquire a set of exact results. And our work focuses on the filter step. Like other RkNN works, the pruning heuristics are used to eliminate the false objects in the filter step. We use two pruning heuristics here. We present a method called BRC in the filter step. This method needs pre-computing the cover-value [11] for each node when the R-tree is built. The cover-value of a node is the number of the objects contained in the sub-tree rooted by this node. The R-tree is built by inserting the objects one by one. When an object is inserted to a leaf, the cover-value of each node on the path from the root to the leaf should be increased by one.

For the R-tree based RkNN queries, traverse the nodes of the R-tree is necessary. To increase the speed of a query, reducing the number of the visited nodes and lessening the I/O cost are very important. If it can be vindicated by dealing with the current node that the sub-tree rooted by the current node does not contain the real results, the current node can be pruned and its child nodes should not be visited. The following content mainly discusses how the nodes of a R-tree are pruned. In other words, how is it vindicated that the objects contained in the nodes of the R-tree can not be included in the final results.

The essence of RkNN queries: a multidimensional object set P and points $q \notin P, p \in P$, if it exists a set $P' \subset R$ ($|P'| \geq k$), for each object $p_i \in P'$ exists $dist(p_i, p) < dist(q, p)$. Although these objects are not k NN of p , it is sure that q isn't one of k NN(p). That is, p isn't one of Rk NN(q).

Pruning Heuristic 1

The motivation of the pruning heuristic 1: If a region R on data space contains n ($n > k$) objects and objects in this region R mutually close (that is, for each object p contained in R , it at least exists other k objects contained in R are closer p than q), there are not Rk NN(q) contained in R .

Lemma 1: For a node N , if $cover(N) \geq k + 1$ and $mindist(N, q) > diag(N)$, the node N can be pruned. Here, $mindist(N, q)$ is the minimal distance between the MBR of the node and q , $diag(N)$ is the diagonal of the MBR of the node (see Fig. 2).

Proof. The object p and p' contained in N , $dist(p, p') \leq diag(N)$ and $mindist(N, q) \leq dist(p, q)$. If $diag(N) < mindist(N, q)$, then $dist(p, p') < dist(p, q)$. There exists at least k objects p' satisfied $dist(p', p) < dist(q, p)$ in N , so p is not in Rk NN(q). ■

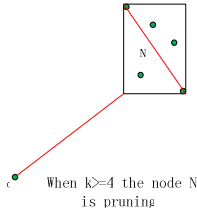


Fig. 2. Pruning Heuristic 1

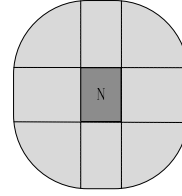


Fig. 3. $cover(N) > k$, the case of N not pruned

In the following cases, a node N will not be pruned by the pruning heuristic 1.

- (a). $cover(N) \leq k$
- (b). $cover(N) > k$ and q lies on the light region (see Fig.3) where the length of the dotted line equals to $diag(N)$.

The experiment will show most of objects can be pruned by pruning heuristic 1 and the amount of calculation is rather minimal.

Pruning Heuristic 2

The motivation of the pruning heuristic 2: If a region R on data space contains n ($n > k$) objects, a object $p \notin R$ satisfies that $maxdist(p,R) < mindist(p,q)$ (i.e. there exists at least other k objects contained in R which are closer to p than q), p is not in $RkNN(q)$.

Lemma 2: For a node N , if $cover(N) \geq k$ and it is showed the relative position of q and N in Fig.4 (q lies on the bottom left corner of green lines), the node N can be pruned.

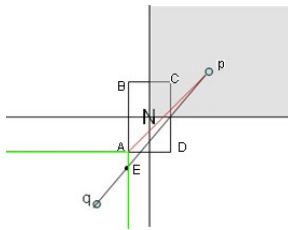


Fig. 4. Pruning Heuristic 2

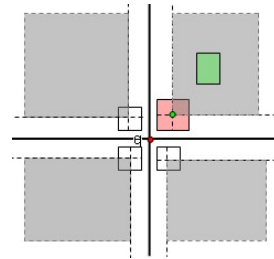


Fig. 5. Using pruning 2 heuristic in 2D-space

Proof. The farthest distance between object p existed in shaded area and the node N is $dist(A, p)$ (the red line in Fig.4). Segments qp and BA intersect in point E . Obviously, $dist(q, p) > dist(E, p)$, $dist(E, p) \geq dist(A, p)$, then $dist(q, p) > dist(A, p)$. In the meantime, $dist(A, p)$ is the farthest distance between the node N and p . The distance between objects contained in N and p is lesser than $dist(q, p)$. Because

of $cover(N) \geq k$, there exists at least k objects which satisfy $dist(p', p) < dist(q, p)$. So the nodes or objects existed in the shaded area can't be contained in $RkNN(q)$. ■

Taking 2D-space for instance, to increase the effectivity of the pruning heuristic 2, the whole object space is divided as Fig. 5, where q is the center.

In 2D-space, whole object space is divided in 4 sections. There is a rectangle RN for each section and $cover(RN) = k$. Then the pruning heuristic 2 is used in every sections, each section forms a shaded area. And objects contained in the shaded area are not $RkNN(q)$. For increasing the pruning affectivity and forming greater shaded area, k nearest objects are selected in each section, and using these k objects form the rectangle RN . Because the BRC algorithm uses a priority queue based on the min-distances, the front k objects belonging to each section are the ones.

In Fig.5, when the nodes of the R-tree (the green rectangle) entirely belong to one section, the pruning heuristic 2 can be used. Through the observation, it is clear that when the relative position of a node N and the central point of RN identifies with the relative position of RN and q , the nodes or objects can be pruned by the pruning heuristic 2. For example, there are 4 relative positions of a node and a point in 2D-space (the relative positions of 4 rectangles and the red point in Fig.5).

The partition about multidimensional object space is similar to 2D-space. Assuming the dimension is n then the multidimensional object space is divided into 2^n sections. Similarly, there are 2^n relative positions.

BRC Algorithm

The BRC algorithm uses a priority queue based on the min-distances to preserve the nodes of the R-tree and objects which haven't been visited for now but will be visited. At first, BRC inserts the root of the R-tree into a heap H sorted in ascending order of its entries' minimal distance from q . Then, BRC de-heaps an entry e and visits e . If it can not be pruned, its child nodes are inserted into H . The filter step terminates when $H = \emptyset$. Next, it is specifically described how an entry can be pruned.

The case of $cover(e) > k$: First BRC verdicts whether e can be pruned by pruning heuristic1. If e can't be pruned by the pruning heuristic1, and e entirely belongs to one section, then the pruning heuristic 2 is used to prune. If e can't be pruned by the above pruning heuristics, then the child nodes are inserted to H .

The case of $cover(e) \leq k$: (1) An entry e entirely belongs to a section and e is an object. If this section have not formed the pruning rectangle, then the object is inserted to the set of rectangle which is used to form pruning rectangle. If this section have formed the pruning rectangle and the pruning heuristic 2 is successful, then the object is inserted to the set S_{trim} , else consider whether the maximal distance between this object and the pruning rectangle is lesser than the distance between this object and q . If the answer is yes, then the object is inserted to the set S_{trim} . Otherwise then the object is inserted to the set S_{scnd} . (2) An entry e entirely belongs to a section and e is a node of the R-tree. If this section have formed the pruning rectangle and the pruning heuristic 2 is successful, then the node is inserted to the set S_{trim} . (3) If an entry e does not entirely belong to one section and e is an object, then the object is inserted to the set S_{scnd} . (4) Otherwise, the child nodes are inserted to H .

BRC algorithm:

```

1  initialize  $H = \emptyset$ ;
2  insert the root of the R-tree to H;
3  bool  flag[ $2^{\text{dim}}$ ]; // Whether pruning rectangles have
                        formed for every sections
4  pointSet  site[ $2^{\text{dim}}$ ]; //The pruning rects for every
                        sections
5  WHILE   $H \neq \emptyset$  DO
6     $e \leftarrow$  the first entry of H
7    IF cover( $e$ )  $> k$ 
8    THEN
9      (IF trim1( $e$ ) THEN ( $S_{\text{trim}}$ .insert( $e$ ). continue.)
10     site=judge_Mbr_Point( $e$ .MBR,  $q$ , dim).
11     IF site  $\neq -1$  THEN
12       (IF trim2( $e$ )
13        THEN ( $S_{\text{trim}}$ .insert( $e$ ).
14              continue.))
15     ELSE
16       site=judge_Mbr_Point( $e$ .MBR,  $q$ , dim).
17       IF (site  $\neq -1$ )
18       THEN
19         (IF  $e$  is an object
20          THEN (IF !flag[site-1]
21                THEN (insert it to current pruning rect.
22                      continue.)
23                 ELSE ( IF trim2( $e$ )
24                        THEN ( $S_{\text{trim}}$   $\rightarrow$  insert( $e$ ).
25                               continue.)
26                       ELSE ( IF trim3( $e$ )
27                              THEN ( $S_{\text{trim}}$ .insert.
28                                    continue.)
29                              ELSE
30                                 $S_{\text{scnd}}$ .insert( $e$ )))
31          ELSE (IF flag[site-1] AND trim2( $e$ )
32                THEN ( $S_{\text{trim}}$ .insert( $e$ ).continue.))
33          ELSE (IF  $e$  is objects THEN
34                ( $S_{\text{scnd}}$ .insert( $e$ ).continue.))
35     insert its child nodes to H.

```

At the last, we will briefly discuss the refinement step of $RkNN$ processing. The output of the filter step is used as the input of the refinement step. There are usually two processing methods for the refinement step of $RkNN$: one is using kNN queries. kNN queries are performed for each object. If the distance between a candidate object and the k -nearest neighbor is not lesser than the distance between the candidate object and q , then this candidate object is a result. The other one uses $range - k(q, p)$. If the result is *true*, then this candidate object is a result. But the two methods all need to traverse R-tree again. So we employ the refinement method of the TPL. After the filter step, we have a set S_{scnd} of candidate objects and a set of nodes and objects pruned. In the refinement step, S_{scnd} and S_{trim} can be used as input. These candidate objects are vindicated by using the nodes of the R-tree and objects pruned. So it prevents revisiting the same nodes from happening.

4 The Filter Step for Continuous RkNN Processing

The continuous query point is represented by a segment which begins at q_A ends at q_B . Given a segment q_Aq_B , a C-RkNN query aims at reporting the RkNNs for every point on the segment. The goal is to find a set of split points that partition q_Aq_B into disjoint sub-segments, such that all points in the same sub-segment have identical RkNNs.

As with RkNN queries, C-RkNN queries also have a filter and a refinement step for retrieving and verifying candidates, respectively. However, unlike conventional RkNN search, C-RkNN includes a third step(the splitting step) for obtaining the split points.

Since C-BRC is similar to BRC, our discussion focuses on clarifying the differences between the two algorithms.

C-RkNN Pruning Heuristic 1

Lemma 3: For a node N , if $cover(N) \geq k + 1$ and $mindist(N, q_Aq_B) > diag(N)$, the node N can be pruned. Here, $mindist(N, q_Aq_B)$ is the minimal distance between the MBR of the node and q_Aq_B , $diag(N)$ is the diagonal of the MBR of the node (see Fig. 6).

Proof. Given objects p and p' contained in N and a point $q \in q_Aq_B$, $dist(p, p') \leq diag(N)$ and $mindist(N, q_Aq_B) \leq mindist(N, q) \leq dist(p, q)$. If $diag(N) < mindist(N, q_Aq_B)$, then $dist(p, p') < dist(p, q)$. That is, it exists at least k objects satisfied $dist(p, p') < dist(p, q)$ in N , so p isn't one of $RkNN(q_Aq_B)$. ■

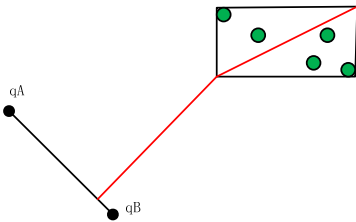


Fig. 6. Pruning Heuristic 1 for CRkNN

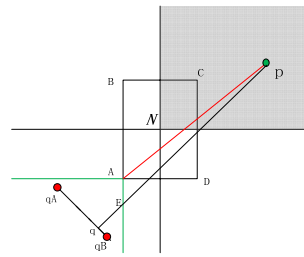


Fig. 7. Pruning Heuristic 2 for C-RkNN

C-RkNN Pruning Heuristic 2

Lemma 4: For a node N , if $cover(N) \geq k$ and it is showed the relative position of q_Aq_B and N in Fig.7 (q_Aq_B lies on the bottom left corner of green lines), the node can be pruned.

Proof. The farthest distance between object p existed in shaded area and the node N is $dist(A, p)$ (the red line in Fig.7) and the point $q \in q_Aq_B$. Segments qp and BA

intersect in a point E . Obviously, $\text{mindist}(q_A q_B, p) \geq \text{dist}(q, p) > \text{dist}(E, p)$, $\text{dist}(E, p) \geq \text{dist}(A, p)$, then $\text{mindist}(q_A q_B, p) > \text{dist}(A, p)$. In the meantime, $\text{dist}(A, p)$ is the farthest distance between the node N and p . The distance between objects contained in N and p is lesser than $\text{dist}(q, p)$. Because of $\text{cover}(N) \geq k$, it exists at least k objects which satisfy $\text{dist}(p', p) < \text{mindist}(q_A q_B, p)$. So the nodes or objects existed in shaded area can't contain $RkNN(q_A q_B)$. ■

For increasing the effectivity of the C-RkNN pruning heuristic 2, the whole object space is divided like RkNN with the static query point. The point q is replaced by the rectangle whose diagonal is $q_A q_B$. Fig. 8 takes the 2D-space for instance. Similarly, the 4 green rectangles are formed in the same way and objects contained in the shaded area are not $RkNN(q_A q_B)$. So when the nodes of the R-tree entirely belong to a shaded area, the pruning heuristic 2 can be used. Similarly, when the relative position of a node N and the central point of RN identifies with the relative position of RN and $q_A q_B$, the nodes or objects can be pruned by the pruning heuristic 2. The relative position of a rectangle R and a segment $q_A q_B$ is defined as the relative position of R and q_A/q_B , when the relative position of R and q_A identifies with the relative position of R and q_B . There are 4 relative positions with a rectangle and a segment in 2D-space. The partition about multidimensional object space is similar to 2D-space.

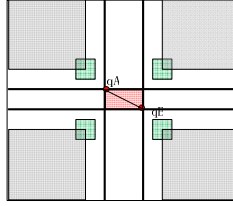


Fig. 8. Using pruning heuristic 2 in 2D-space for C-RkNN

C-BRC Algorithm

The C-BRC algorithm for filter step is similar to BRC. In this paper, we employ the refinement and the splitting step of TPL. The criterion of the refinement step is that a object p is a final result if and only if no other objects exists in the circle centered at p with a radius $\text{mindist}(q_A q_B, p)$. In the splitting step, every candidate obtains the true NN -dist regarding to the entire dataset. The split points on $q_A q_B$ are the intersections between the circle of every candidate and $q_A q_B$.

5 Experimental Evaluation

RkNN with Static Query Point

In this section we study the performance of our proposed algorithm, namely the BRC algorithm for RkNN queries with static query point. We compare its performance with the TPL algorithm, which is the state-of-the-art RkNN algorithm. Experiments

are run on a Windows XP desktop machine with a Core 2 Duo 2.8GHZ CPU and 1G memory. We deploy four real datasets (LB,hypsogr,wave and color) whose statistics are summarized in Table 1. We also employ SpatialDataGenerator to create synthetic data following the uniform and Zipf distributions. The random coordinates of each point in a uniform dataset are specified in [0,10000], the coordinates of each point in a Zipf dataset follow Zipf distribution (with a skew coefficient 0.8). Each point's coordinates in the above datasets on each dimension are mutually independent. Each dataset is indexed by an R*-tree, and the size of a node is fixed to 1k bytes.

Table 1. Statistics of the real datasets used

	LB	hypsogr	wave	color
Dimensionality	2	2	3	4
Cardinality	53145	76999	65536	65536

The experiments focus on investigating the influence of these factors: data distribution, dataset cardinality, dimensionality, value of k . For one and the same query, we will execute 400 queries (the query points for the 400 queries belong to the object space), all of the reported value in the following diagrams each reported value is the average of all the 400 queries.

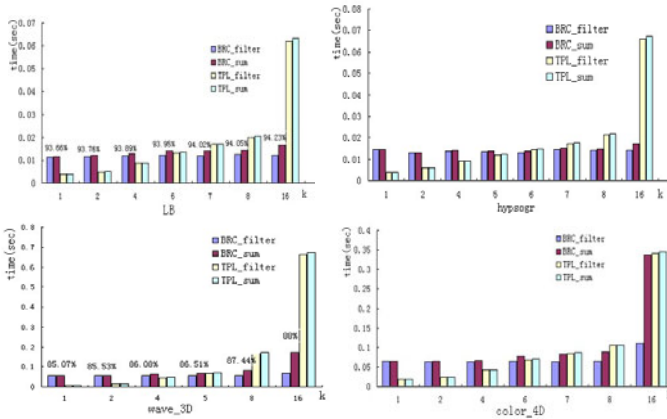


Fig. 9. Time taken by BRC and TPL while different values of k

Fig.9 shows the query time (in seconds) of the BRC and TPL when k takes different values with the real datasets. The query time is divided in two components (the filter step and the refinement step). The numbers in Fig.9 indicate the percentages of objects pruned by the pruning heuristic 1 in the total dataset cardinality. It is showed that BRC costs lesser than TPL for all datasets when k takes larger value. And the pruning heuristic 1 of BRC is independent of k , so the time taken by the filter step of BRC varies within a small range. The percentages in Fig. 9 confirm that most of objects can be pruned by the pruning heuristic 1. Besides the amount of calculation of the pruning heuristic 1 is very small. These two causes lead to the lesser query time taken by BRC.

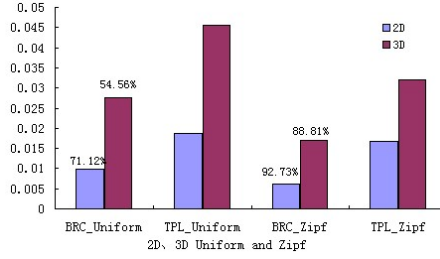


Fig. 10. $k=8$, time taken by 2D、3D *Uniform* and *Zipf* datasets

Next, the experiments inspect the impact of the dimensionality. Here, the value of k takes 8, the 2D and 3D datasets (*Uniform* and *Zipf*) contain 10000 objects. Fig. 10 shows that the performance of BRC and TPL degrades with the dimensionality growing, because the R*-tree become less efficient with the dimensionality growing. Moreover, the percentages in Fig.10 show that the pruning heuristic 1 of BRC is more efficient on *Zipf* datasets than *Uniform* datasets.

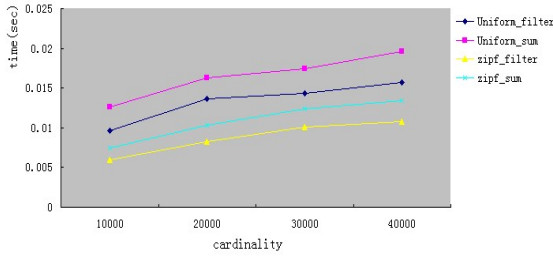


Fig. 11. Time taken by different size of *Uniform* and *Zipf* datasets

To study the effect of the dataset cardinality, we use the same data distribution, same dimensionality and different cardinalities datasets. In these experiments, k takes 8, the cardinalities of 2D datasets (*Uniform* and *Zipf*) range from 10000 to 40000. Fig.11 shows the query time increases with the cardinalities growing, because the height of the R*-tree increases with the cardinalities growing.

RkNN with Moving Query Point

Above having demonstrated the efficiency of BRC for RkNN queries with static query point, we proceed to evaluate C-BRC for C-RkNN. We compare C-BRC with C-TPL. In addition to the above factors, the query performance is also affected by the length l of the query segment $q_A q_B$.

Fig.12 shows the query time of the C-BRC and C-TPL when k takes different values with the real datasets (here $l=100$). Its comparison with C-TPL is similar to that between BRC and TPL in Fig. 9.

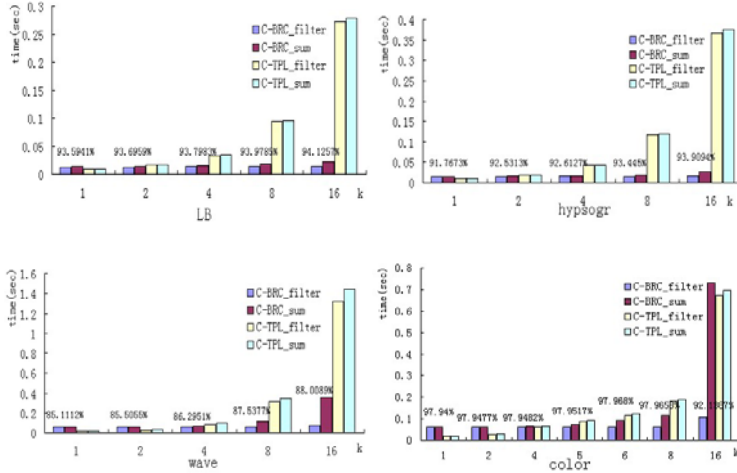


Fig. 12. Time taken by C-BRC and C-TPL while different values of k

Next, similar studies about the effects of dimensionality and cardinality on the performance of C-BRC are illustrated in Fig.13 and Fig.14 (here, $k=8$ and $l=100$).

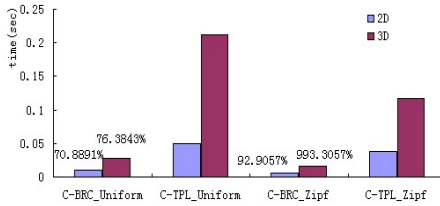


Fig. 13. $k=8$, time taken by 2D, 3D *Uniform* and *Zipf* datasets for C-RkNN

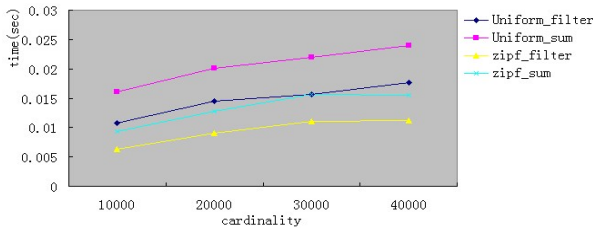


Fig. 14. Time taken by different size of *Uniform* and *Zipf* datasets for C-RkNN

Finally, Fig. 15 examines the performance of C-BRC and C-TPL by varying l from 10 to 200. And we can see that C-BRC still outperforms C-TPL significantly.

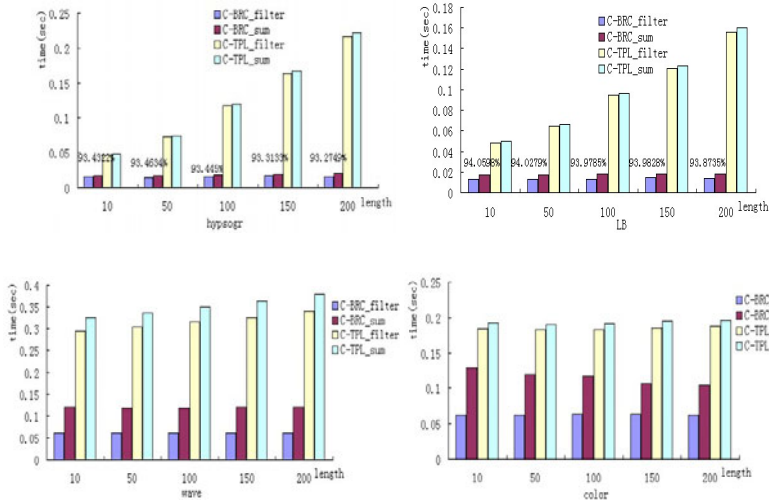


Fig. 15. Time taken by C-BRC and C-TPL while different values of l

6 Conclusion

This paper proposes a new filter processing for $RkNN$ with static query point and continuous $RkNN$. There are two pruning heuristics proposed for the filter step. The amount of calculation of the two pruning heuristics is lesser. The processing speed of BRC is also acceptable when the value of k is great. And we extend the BRC method to answer the continuous $RkNN$ queries. Future work will be focused on the continuous $RkNN$ for moving objects trajectories.

Acknowledgment. This work is supported by Jilin University Research Project under Grant No. 200903178, Seed Foundation of Jilin University and Open Project of Key laboratory of symbolic computing and knowledge engineering of ministry of education in China.

References

1. Tao, Y., Papadias, D., et al.: Multidimensional reverse kNN search. The International Journal on Very Large Data Bases 16(3), 293–316 (2007)
2. Benetis, R., Jensen, C.S., et al.: Nearest and Reverse Nearest Neighbor Queries for Moving Objects. The International Journal on Very Large Data Bases 15(3), 229–249 (2006)
3. Korn, F., Muthukrishnan, S.: Influence sets based on reverse nearest neighbor queries. In: ACM SIGMOD International Conference on Management of Data, pp. 201–212 (2000)
4. Yang, C., Lin, K.-I.: An index structure for efficient reverse nearest neighbor queries. In: ICDE, pp. 485–492 (2001)
5. Maheshwari, A., et al.: On reverse nearest neighbor queries. In: CCCG, pp. 128–132 (2002)

6. Stanoi, I., Divyakant, A., et al.: Reverse nearest neighbor queries for dynamic databases. In: ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, pp. 744–755 (2000)
7. Singh, A., Ferhatosmanoglu, H., Tosun, A.S.: High dimensional reverse nearest neighbor queries. In: The 20th International Conference on Information and Knowledge Management, pp. 91–98 (2003)
8. Wu, W., Yang, F., et al.: Finch: Evaluating reverse k-nearest-neighbor queries on location data. *The International Journal on Very Large Data Bases* 1(1), 1056–1067 (2008)
9. Cheema, M.A., Lin, X., et al.: Lazy Updates: An Efficient Technique to Continuously Monitoring Reverse kNN*. *The International Journal on Very Large Data Bases* 2(1), 1138–1149 (2009)
10. Xia, T., Zhang, D.-h.: Continuous Reverse Nearest Neighbor Monitoring. In: Proceedings of the 22th International Conference on Data Engineering, pp. 77–86 (2006)
11. Güting, R.H., Behr, T., Xu, J.: Efficient k-nearest neighbor search on moving object trajectories. *The International Journal on Very Large Data Bases* 19(5), 687–714 (2010)

Keyword Query Cleaning with Query Logs

Lei Gao¹, Xiaohui Yu^{1,2,*}, and Yang Liu¹

¹ School of Computer Science & Technology, Shandong University, Jinan, China

² School of Information Technology, York University, Toronto, Canada
gaolei_sdu@yahoo.com.cn, xhyu@yorku.ca, yliu@sdu.edu.cn

Abstract. Keyword queries over databases are often dirty with some irrelevant or incorrect words, which has a negative impact on the efficiency and accuracy of keyword query processing. In addition, the keywords in a given query often form natural segments. For example, the query “Tom Hanks Green Mile” can be considered as consisting of two segments, “Tom Hanks” and “Green Mile”. The goal of keyword query cleaning is to identify the optimal segmentation of the query, with semantic linkage and spelling corrections also considered. Query cleaning not only helps obtaining queries of higher quality, but also improves the efficiency of query processing by reducing the search space. The seminal work along this direction by Pu and Yu does not consider the role of query logs in performing query cleaning. Query logs contain user-issued queries together with the segmentations chosen by the user, and thus convey important information that reflects user preferences. In this paper, we explore the use of query logs to improve the quality of keyword query cleaning. We propose two methods to adapt the scoring functions of segmentations to account for information gathered from the logs. The effectiveness of our approach are verified with extensive experiments conducted on real data sets.

Keywords: query cleaning, keyword search, query log.

1 Introduction

Popularized by Web search engines, keyword search has become the de-facto standard way for people to access unstructured information. In recent years, a great deal of research has been conducted in the database community to also support keyword search as a way for users to access structured data repositories such as XML documents and relational databases. Enabling users to access databases using simple keywords can relieve them from the trouble of mastering a structured query language and understanding complex and possibly fast evolving database schemas. However, for structured databases, such as relational databases, it is a very challenging task to develop high performance and robust keyword search algorithms. One such challenge is that for structured data, the expected query result is no longer a list of relevant documents and the corresponding search space is no longer a collection of documents as in unstructured

* Corresponding author.

information retrieval. Rather, the expected result is usually a joined network of tuples [34] or a database view [7]; the search space is also increased dramatically as it is necessary to assemble keyword-matching tuples from different tables into the final results. In general, the search space is exponential in the number of keywords in the query.

Besides the exponential explosion of the search space, the dirtiness of keyword queries also influence the accuracy and effectiveness of keyword search, making keyword search over the structured database more difficult. Keywords query is called dirty when it contains words which are not intended as part of query or dirty words. The so-called dirty words refer to words with spelling errors or those that do not appear in the database but are semantically equivalent to some words in the database. In addition, the keywords in a given query are not isolated and may have connection with each other, which means that nearby query words may be grouped into segments that could match some tuples in the database. For example, the query “Tom Hanks Green Mile” over a movie database may be considered as consisting of two segments, “Tom Hanks” and “Green Mile”. Performing query segmentation before conducting keyword processing can often reduce the number of logical units to be considered during the search and thus significantly reduce the search space.

Motivated by the above observations, Pu and Yu [8] introduce a pre-processing phase to keyword query processing, called keyword query cleaning, which involves two main tasks: keyword rewriting and query segmentation. Keyword rewriting rewrites the keywords in the query to compensate for spelling errors, synonyms of text words in the database, while query segmentation is mainly responsible for grouping nearby keywords in the query into logical segments.

In this paper, we propose two approaches to enhance the original scoring function. Both approaches make use of the information embedded in the query log to extend the original scoring function. The first enhanced approach combines the segment score on the database with the segment score on the query log by normalization and weighting, while the second approach boosts the segment score using a predefined boost function. The extensive experiments we conducted prove the effectiveness of our two approaches.

The rest of the paper is organized as follows. Section 2 introduces related work. Section 3 defines some related basic concepts. Section 4 presents two enhanced scoring functions based on query logs. Section 5 reports the experimental results. Section 6 concludes the paper.

2 Related Work

Keyword search over the structured database has been studied extensively in recent years. Early works [13,4] in this area mainly focus on finding the best join networks to assemble matching tuples from different tables in the database. The optimal join network problem, or alternatively, the Steiner tree problem, has been proved to be NP-complete with respect to the number of relevant tables. Therefore, some heuristic algorithms for finding the top-k candidate join

networks have been proposed [4]. Some recent works mainly focus on more reasonable ranking functions to improve the effectiveness of keyword search [5,6]. [3] first introduce the classical information retrieval (IR) technology to the ranking function, which has been shown to be very helpful. [5] makes further improvement based on the previous work by introducing the size normalization factor. In addition to works on keyword search over traditional relational databases, there are many existing works on other types of structured data. [15] discusses the problem of performing works on relational data streams. [7] addresses the issue of keyword search on relational databases with star-schemas commonly found in OLAP applications, where the focus is finding the relevant data subspaces rather than joining networks. [16] introduces keyword search over the spatial database. Keyword search over XML data has also attracted a great deal of attention [12,14]. [10,11,13] address the issue of top-k database selection across multiple databases.

Tan and Peng [9] have addressed the issue of query cleaning in the Web context. However, approaches proposed there cannot be easily applied to query cleaning in the context of relational databases because of the different characteristics of structured data and unstructured data. Pu and Yu [8] first introduce the problem of keyword query cleaning over the databases. It is shown that the added query cleaning phase not only improves the quality of the search result but also significantly reduces the search space for the subsequent search algorithm. However, the scoring function in [8] does not consider user preferences. Such preferences can be mined from query logs, which contain past queries issued by the user as well as the correct results chosen. In this paper, we enhance the work by Pu and Yu by designing scoring functions used in query cleaning that could make use of query logs to adapt to user preferences.

3 Preliminaries

We first introduce some basic concepts involved in query cleaning. We define tokens as strings that are considered as indivisible units, and terms as sequences of tokens. Let D be a database (relational or XML). A database token is a token which appears in somewhere in the database at least once, and the set of all database tokens of D is denoted by $token^D$. Similarly, a database term is a term which appears in the database at least once, and the set of all database terms of D is denoted by $term^D$.

Let L be a query log. We assume that an entry in the query log L has at least the following four fields: user ID, the query posed by user, the cleaned query selected by user, and the timestamp that queries are posed. A log token is a token which appears in the cleaned query field at least once, and the set of all log tokens of L is denoted by $token^L$. Similarly, a log term is a term which appears in cleaned query field at least once, and the set of all log terms of L is denoted by $term^L$.

Segment and *Segmentation* are two key concepts we often refer to in our paper. A *segment* is defined as a sequence of tokens in the query. A segment S is

considered valid if this sequence appears in the database at least once, that is, $S \in \text{term}^D$. A *segmentation* is defined as a sequence of non-overlapping segments that completely cover all tokens in the query. A segmentation \mathbb{S} with respect to a query is valid if and only if each segment of \mathbb{S} is valid. Note that a query may potentially have many valid segmentations. For example, the query “XML document structure” can have four valid segmentations: (XML, document, structure), (XML document, structure), (XML, document structure), and (XML document structure). For a given query, the objective is to find the optimal segmentation with respect to a certain scoring function. The main focus of our work is to identify the scoring functions that take into consideration user preferences.

4 Adapting to User Preferences

Keyword queries can have many different valid segmentations. It is therefore necessary to rank those segmentations according to some criteria. Ideally, such criteria should take user preferences into account, as even for the same query, different users may prefer different segmentations based on their information need.

4.1 The Original Approach

The scoring function for a single segment S in [8] is defined as follows:

$$\text{score}_{DB}(S) = \text{score}_{IR}(T_S) \cdot \text{normalize}(\delta_Q(S), \delta_{exp}(S)) \cdot \text{boost}(|T_S|) \quad (1)$$

where $\text{score}_{IR}(T_S) = \max\{tfidf(T_S, T) : T \in \text{term}^D\}$.

In Equation (1), $\delta_Q(S)$ is the maximum distances (in the original query) between each adjacent pair of tokens in the segment. $\delta_{exp}(S)$ is the total expansion distance (i.e., the sum of distances between the original tokens and the “corrected” tokens). $\text{score}_{IR}(S)$ is the information retrieval score of the segment induced terms. *normalize* is an anti-monotonic normalization function and *boost* is a monotonic function with a lower bound no less than 1.

The rationale behind this scoring function is as follows [8]:

1. We prefer that tokens adjacent to each other in the user-supplied query are grouped into a single segment.
2. We prefer to minimize the changes made to the original query tokens.
3. We prefer longer segments.

In order to adapt to user preference, this scoring function has to be enhanced. The basic idea is to make use of information contained in query logs that record past user behaviour. In what follows, two enhanced approaches are proposed to improve the cleaning quality on basis of the original scoring function by making use of the query logs.

4.2 The First Enhanced Approach: Normalization and Weighting

In the first approach we propose, two additional terms based on the log information are added to the original scoring function. The first term added to the scoring functions is as follows:

$$score_{Log1}(S) = score_{IR2}(T_S) \cdot normalize(\delta_Q(S), \delta_{exp}(S)) \cdot boost(|T_S|) \quad (2)$$

where $score_{IR2}(T_S) = \max\{tfidf(T_S) : T \in term^L\}$

Where $term^L$ is the terms in the cleaned query field of the query log. This formula is similar to Equation (II), but it uses log information instead of the database in computing the scores. It also shares a similar rationale. That is, we favour the segment which is similar to the terms appearing in the log.

We further use the rank of a segment to measure the user preference of different segments. This rank refers to the rank of the access frequency of its corresponding query terms in the query log, and can be obtained as follows. The query terms are sorted based on their frequency. Similar frequencies, where the frequency difference between the corresponding terms is below the predefined threshold, are treated as having the same rank. It is believed that segments with higher ranks are often preferred by users because they are chosen more often by users. Therefore, we consider the score of a segment to be inversely proportional to its rank. Hence, the second term added to the ranking function is as follows:

$$score_{Log2}(S) = \frac{1}{1 + \log(r)} \quad (3)$$

Here r is the rank of a segment. Note that the logarithm of the rank is used here instead of the original rank value in order to dampen the effect of the rank.

Before combining the two terms mention above with the original scoring function, normalization is necessary because of their different scales. We need to normalize all three terms into the unit interval $[0, 1]$. We use a sigmoid function to achieve this effect, which takes the form of:

$$sigmoid(x) = \frac{1}{1 + e^{-\Delta x}}$$

where the weight parameter Δ controls the linearity of the curve.

The sigmoid function can be made more linear by scaling the parameter Δ . For example, the sigmoid function is nearly same as the linear function $y = kx + 0.5$ within range $[-6, 6]$ when $\Delta = 0.1$. The sigmoid function can also be made linear in a larger range by adjusting the Δ value. The range of the original sigmoid function in the interval $[0, +\infty]$ is $[0.5, 1]$. Here we scale its range to interval $[0, 1]$ with the transformation $2 \cdot (sigmoid(x) - 0.5)$. So the normalization function used here is as follows:

$$NRM(x) = 2 \cdot (sigmoid(x) - 0.5) \quad (4)$$

The above function can be made approximately linear within a larger range by setting a reasonable Δ value. In such cases, the NRM function has the property of preserving the original order of different segmentations (as dictated by

Equation (II). The following example illustrates this property. Consider two possible segmentation of a given query Q , $\mathbb{S}_1 = (S_1, S_2, S_3)$, $\mathbb{S}_2 = (S_4, S_5)$. The relationship between their scores is $score(\mathbb{S}_1) \geq score(\mathbb{S}_2)$ according to Equation (II). After normalizing the score of every segment, the relationship between their scores remains same, that is, $NRM(score(\mathbb{S}_1)) \geq NRM(score(\mathbb{S}_2))$.

We favour the segment which is similar to the segment in the query log and also have a high rank. Thus, three terms can be combined after normalization as follows:

$$score_{Final}(S) = \lambda \cdot NRM(score_{DB}(S)) + (1 - \lambda) \cdot (score_{Log}(S)) \quad (5)$$

$$score_{Log}(S) = \frac{NRM(score_{Log1}(S)) + score_{Log2}(S)}{2} \quad (6)$$

where λ is the proportion of the score from Equation (II) in the final score. Note that Equation (III) does not require normalization because its range is in $[0, 1]$.

There is a relative importance between the segment similarity and segment rank, which means which factor of two is more important. Here we use the average value of the Equation (II) and (III) as the final segment score based on the query log, which means that each part contributes half of their score to the final log score and have the same weight. Of course, other weighting schemes may also be feasible. Note that this enhanced score function is the same as the original scoring function when $\lambda = 1$.

Let's take a simple example to illustrate the above equation. Suppose that a query $Q = t_1 t_2 t_3$ needs to be segmented, where $t_1 t_2$ and $t_2 t_3$ can be grouped into a valid segment, thus generating two possible segmentations $\mathbb{S}_1 = (t_1 t_2, t_3)$, $\mathbb{S}_2 = (t_1, t_2 t_3)$. Suppose that $t_2 t_3$ are used in the query log while $t_1 t_2$ are not. So the segmentation \mathbb{S}_2 may be the segmentation that the user wanted. Without taking the query log into account, that is, only following the original approach, the relationship between two segmentations $\mathbb{S}_1, \mathbb{S}_2$ is that $score_{DB}(\mathbb{S}_1) \geq score_{DB}(\mathbb{S}_2)$. When the query log is taken into account, the relationship between two segmentations $\mathbb{S}_1, \mathbb{S}_2$ may be reversed, that is, $score_{final}(\mathbb{S}_1) < score_{final}(\mathbb{S}_2)$, because $score_{Log}(\mathbb{S}_1) < score_{Log}(\mathbb{S}_2)$. By combining the score based on query log, the segmentation is adapt to user preference.

4.3 Second Enhanced Approach: Boosting the Original Score

The second enhanced approach is to boost the original segment score when the segment is found in the query log. The following is the enhanced scoring function:

$$score_{Final}(S) = score_{DB}(S) * boost_{log}(S) \quad (7)$$

where the $boost_{log}(S)$ is a boost function for segment S .

The rationale behind Equation (VII) is that the original score of a segment should be boosted if a segment appears in the query log, and its score should not change from the original score if it does not exist in the query log. By

boosting the score, segments appearing more frequently in the query log are more preferred.

There may be many boosting functions to adopt. The boosting function we adopt here is as follows:

$$boost_{log}(S) = e^{\frac{\log(count(S))}{\log(maxCount)}}$$

where $count(S)$ is the times that segment S appears in the query log and $maxCount$ is the maximum count for all segments appearing in the query log. The logarithm of count are used to dampen the effect of counts. Here we boost the score of a segment based on the count of the segment relative to the most popular segment. That is, the more popular a segment is, the more its score should be boosted.

5 Experiments

5.1 Query Log Generation

Due to the lack of real query log data, we generate synthetic query logs instead. Here *zipf distribution* is used to generate the query logs needed. Zipf distribution is one of the important laws in bibliometrics. It is first applied to the statistics of natural language. It shows that only few words in English are frequently used while most of the others are not used so often. The relationship between the frequency of the word and its rank in the frequency table can be expressed in the form of mathematical expression as follows:

$$Pr = \frac{C}{r^\alpha} \quad (8)$$

where C, α are constant for specific application.

It has been shown that many other languages not just English possess this characteristic. The same relationship occurs in many other rankings, unrelated to language, such as the population ranks of cities in various countries, corporation sizes, income rankings, etc. Similarly, the only few terms are often used and most of others are not used so often. Here we assume that the relationship between the frequency of the query term and its rank in the frequency table also conforms to Zipf distribution. Consequently, it is reasonable to simulate user behaviours using Zipf distribution.

In our experiments, we sample 1000 items from the data set (described below), treating them as the items of the database frequently accessed by users, and then generate a synthetic query log with 10,000 query records following Zipf distribution.

5.2 Experimental Setting and Evaluation

Our experiments are mainly conducted on two real datasets, the IMDB dataset and the DBLP dataset. We process the two dataset in the exactly same way as

what is done in [6,8]. The IMDB data contains information on movies, actors, directors, and so on, with 9,839,026 tuples. The raw text files in the IMDB dataset were converted to relational data as described in [6], and then its text attributes are indexed. The XML file in the DBLP dataset is parsed and also converted to the relational data in the way described in [6]. The obtained DBLP data with 881,867 tuples contains information on publications, authors, titles, and publishers, and then its selected text attributes are extracted and indexed.

Our implementation was done purely in Java. The full-text index on the text attributes in the datasets are built using Apache Lucene, an open-source full-text search engine. For each data set, two kinds of indexes were built. The first one is term index, which treats each term as unit for indexing, while the second one is token index, which are obtained by tokenize the terms using whitespace as delimiters.

All experiments are conducted on a Lenovo Desktop running Ubuntu 10.04 operating system, with a 2.33GHz Intel Dual Core processor, 2GB of memory, and 250GB of hard drive.

In order to systematically evaluate the performance of our approaches and eliminate the effect of the subjectivity to our experimental evaluation as possible as we can, test queries need to be generated randomly. The test queries are generated in the same way as what is done in [8,17]. Specifically, for each query to be generated, d terms are sampled from the data set, and for each sampled term, only c contiguous tokens are kept. Spelling errors are introduced to each character in the sampled token with some probability. The terms taken from the d different terms are concatenated to form the query.

The segmentation accuracy is introduced here to evaluate the performance on our approaches. For a given query, \hat{S} denote the sets of segments generated from the query cleaning algorithm, and S denote the true sets of segments. The segmentation accuracy is defined as follows:

$$Accuracy = \frac{|S \cap \hat{S}|}{|\hat{S}|}$$

5.3 Experiments on the First Enhanced Approach

Figure 1(a) and Figure 1(b) demonstrate the effect of parameter λ on the segment accuracy on IMDB and DBLP respectively. Here the experiments are carried out on three different classes of queries, which are short queries, medium queries and long queries respectively. They are generated in the way described in [8]. The short queries are generated with $d = 2$ and $c = 3$, with 6 tokens at most. Medium queries are generated with $d = 5$ and $c = 3$, which have 15 or a bit less tokens. Long queries are generated with $d = 10$ and $c = 3$, with about 30 tokens. In Figure 1(a), it was shown that the segmentation accuracy on arbitrary length queries on IMDB is higher than 80%, while in Figure 1(b) the segmentation accuracy on arbitrary length queries on DBLP is above 70% in most cases.

Query ambiguity is the main reason for causing segmentation errors. When a token in the query can form the valid terms with both the preceding tokens

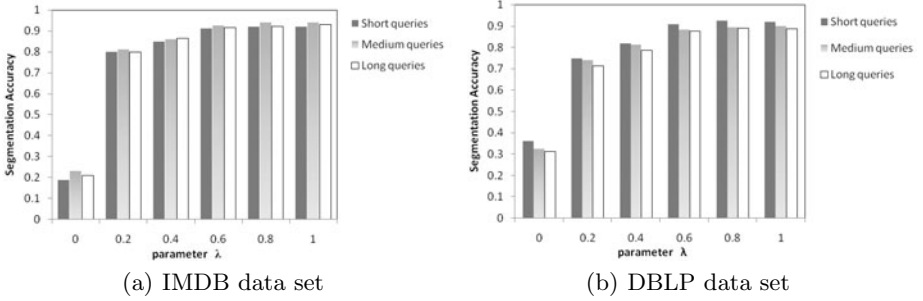


Fig. 1. The effect of λ on segmentation accuracy on First Enhanced Approach

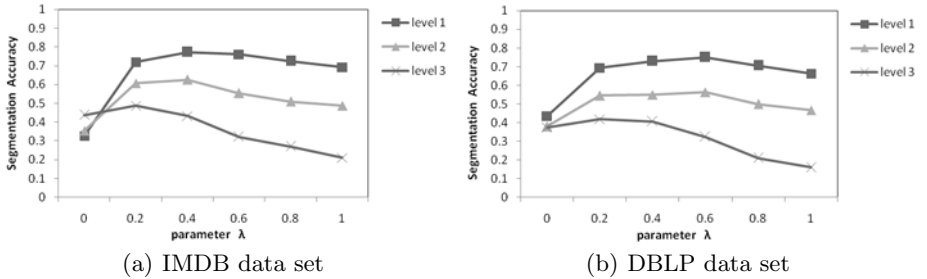


Fig. 2. The effect of λ on segmentation accuracy on First Enhanced Approach at different ambiguity levels

and the following tokens, the query is ambiguous. This situation between two corresponding terms is called an ambiguous connection [17]. The number of ambiguous connections present in a query is used to measure the ambiguity level of a query. We conduct extensive experiments to evaluate how our enhanced approaches perform at different levels of ambiguity present in the queries, compared with the Original Approach.

In our experiment, we generate 1000 medium-sized queries with different ambiguity levels on IMDB and DBLP respectively to evaluate the performance of our approaches under different ambiguity levels. The effect of λ on the segmentation accuracy under different ambiguity levels on IMDB and DBLP are shown in Figure 2(a) and Figure 2(b) respectively. Here we conduct these experiments using medium-sized queries with different ambiguity levels.

Note that the segment accuracy on $\lambda = 1$ is equal to the segment accuracy of the Original Approach. From Figure 2(a) and Figure 2(b), we observe that the segmentation accuracy on $\lambda = 0.4, 0.6, 0.8$ is higher than that on $\lambda = 1$, which proves that the quality of query cleaning is improved by our first enhanced approach. For IMDB dataset, $\lambda = 0.4$ is optimal for all ambiguity levels. For DBLP dataset, choosing $\lambda = 0.4$ is more reasonable for applications because it

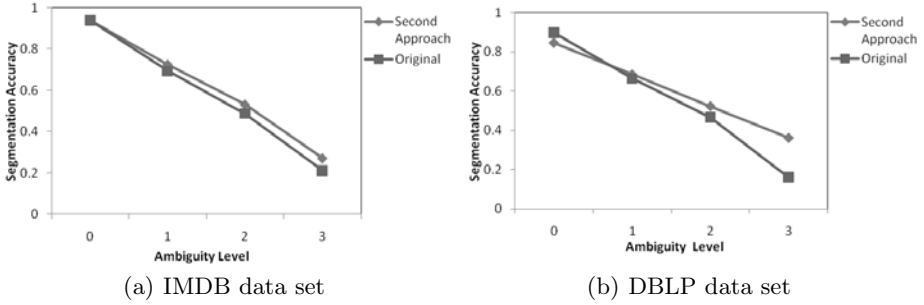


Fig. 3. The Comparison between Original Approach and Second Enhanced Approach

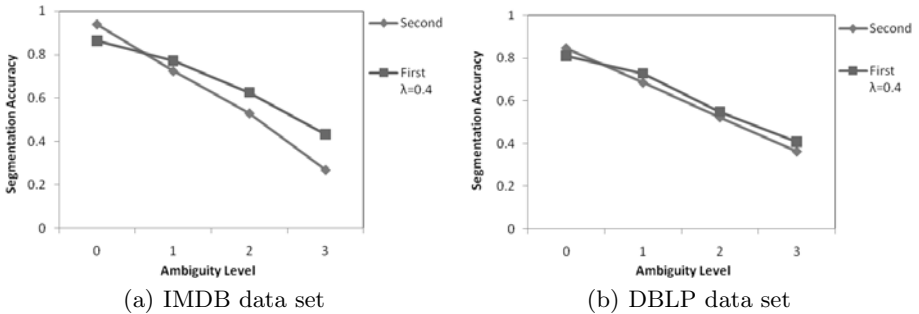


Fig. 4. The Comparison between First Enhanced Approach and Second Enhanced Approach

is optimal for level1 and level2 and there usually not exists so many ambiguities in one query.

5.4 Experiments on the Second Enhanced Approach

Similar experiments are conducted on the Original Approach and Second Enhanced Approach to compare their performance. The segmentation accuracy at different ambiguity levels on the Original Approach and Second Approach on IMDB and DBLP is shown at Figure 3(a) and Figure 3(b) respectively. The experimental results show that the segmentation accuracy gets improved to some extent in the second enhanced approach.

5.5 Comparison between Two Enhanced Approaches

Two approaches are proposed in this paper to enhance the scoring function for improving the cleaning quality. We conduct some experiments on First Enhanced Approach and Second Enhanced Approach to compare their performance. Figure

4(a) and Figure 4(b) show the segmentation accuracy at different ambiguity levels on the First Enhanced Approach and Second Enhanced Approach on IMDB and DBLP respectively, where in the Second Enhanced Approach the parameter $\lambda = 0.4$. It is shown that the First Enhanced Approach performs better than Second Enhanced Approach in most cases when there exists query ambiguities.

6 Conclusion

Keyword cleaning can significantly reduce the search space of keyword search over relational database and improve the efficiency of subsequent keyword search. In this paper, two enhanced approaches are proposed to utilize query logs to adapt to user preferences and improve answer quality. The effectiveness of both approaches have been verified by extensive experiments we conducted.

Acknowledgement. This work was supported in part by National Natural Science Foundation of China Grants (No. 61070018, No. 60903108, 61003051), the Program for New Century Excellent Talents in University (NCET-10-0532), an NSERC Discovery Grant, the Independent Innovation Foundation of Shandong University (2009TB016), and the SAICT Experts Program.

References

1. Agrawal, S., Chaudhuri, S., Das, G.: DBXplorer: enabling keyword search over relational databases. In: SIGMOD (2002)
2. Ding, B., Yu, J.X., Wang, S., Qin, L., Zhang, X., Lin, X.: Finding top-k min-cost connected trees in databases. In: ICDE (2007)
3. Hristidis, V., Gravano, L., Papakonstantinou, Y.: Efficient IR-style keyword search over relational databases. In: VLDB (2003)
4. Hristidis, V., Papakonstantinou, Y.: Discover: keyword search in relational databases. In: VLDB (2002)
5. Liu, F., Yu, C., Meng, W., Chowdhury, A.: Effective keyword search in relational databases. In: SIGMOD (2006)
6. Luo, Y., Lin, X., Wang, W., Zhou, X.: SPARK: top- keyword query in relational databases. In: SIGMOD (2007)
7. Wu, P., Sismanis, Y., Reinwald, B.: Towards keyword-driven analytical processing. In: SIGMOD (2007)
8. Pu, K.Q., Yu, X.: Keyword query cleaning. In: Proc. VLDB Endow. (2008)
9. Tan, B., Peng, F.: Unsupervised query segmentation using generative language models and wikipedia. In: WWW (2008)
10. Sayyadian, M., LeKhac, H., Doan, A., Gravano, L.: Efficient keyword search across heterogeneous relational databases. In: ICDE (2007)
11. Vu, Q.H., Ooi, B.C., Papadias, D., Tung, A.K.H.: A graph method for keyword-based selection of the top-k databases. In: SIGMOD (2008)
12. Guo, L., Shao, F., Botev, C., Shanmugasundaram, J.: XRANK: ranked keyword search over xml documents. In: SIGMOD (2003)
13. Yu, B., Li, G., Sollins, K., Tung, A.K.H.: Effective keyword-based selection of relational databases. In: SIGMOD (2007)

14. Hristidis, V., Koudas, N., Papakonstantinou, Y., Srivastava, D.: Keyword proximity search in xml trees. TKDE (2006)
15. Markowetz, A., Yang, Y., Papadias, D.: Keyword search on relational data streams. In: SIGMOD (2007)
16. Zhang, D., Chee, Y.M., Mondal, A., Tung, A.K.H., Kitsuregawa, M.: Keyword Search in Spatial Databases: Towards Searching by Document. In: ICDE (2009)
17. Yu, X., Shi, H.: Query Segmentation Using Conditional Random Fields. In: KEYS (2009)

A Self-adaptive Cross-Domain Query Approach on the Deep Web

Yingjun Li, Derong Shen, Tiezhen Nie, Ge Yu, Jing Shan, and Kou Yue

College of Information Science and Engineering, Northeastern University,
110004 Shenyang, China
liyijun_neu_ise@163.com,
{shenderong, nietiezheng, yuge, kouyue}@ise.neu.edu.cn

Abstract. As integration systems of data sources in the same domain become more and more, another application comes up with the tide of them. Because of correlation of pairs of domains, when we do some queries involving multiple domains, such as “find a post named software development engineer on job web and look for apartments for rental around the company having been chosen”, we note that general-purpose search engines and general integration frameworks fail to answer cross-domain queries. This paper presents SCDQ, an approach providing fully automated support for cross-domain queries. More specifically, for SCDQ, (i) find which domains are correlated based on data sources having been clustered according to domain. (ii) Recommend different cross-domain paths to meet user’s all possible intentions when query arrives.

Keywords: cross-domain, top-k, Deep Web.

1 Introduction and Motivation

An increasing number of search engines and query interfaces characterize current evolution of the Web, ranging from generic search engines to domain-specific query interfaces. Meanwhile, integration technology is evolving so as to enable it possible to extract content from data-intensive Web sites and visit them as Web Services.

While an increasing amount of data integration systems in a specific domain become available, they still work in isolation; inability to support complex queries ranging over multiple domains is their intrinsic faults. Answering the query reported in our abstract requires combining different domains, for instance, finding an IT training institution and try to buy some books for this training which involves job hunting services and online bookstore services. However, how to identify which pair of domains is correlated in order to pass on the query from the former domain to the latter domain and how to recommend top-k query paths to infer the user all possible intentions are both difficult tasks. In order to solve these problems, this paper describes a novel method of Self-adaptive Cross-Domain Query on deep web (SCDQ) which supports queries over multiple domains assuming that all data sources have been clustered according to domain and each cluster owns a global interface corresponding with a domain.

To discover the correlation between a pair of domains, existing researches mostly resolve it according to experience. Take a specific example: we usually notice that flight, hotel and car rental ranging over three domains appear in the same web services. Though they are integrated in the same service, the queries over them are not connected, which means if we want to do some queries over them, we have to give query conditions in three interfaces respectively. In our method, firstly, we analyze attribute names of query interfaces which reflect their functional information in algorithm 1(ANS). Secondly, data sources' contents behind query interfaces provide us with another way to dig out the correlation between domains in algorithm 2(CCD). Finally, we build a DCM (domain correlation model) and construct a domain correlation graph.

Recommending top-k query paths means we select k query paths for each cross-domain application. For each query, we will build a query tree which describes the correlation between any pair of domains based on DCM. For the domain involved in direct query by the user, search services return a list of available data sources which return query results; for the related domains with it, search services return top-k query paths which are ranked according to the QPEM (Query Path Evaluating Model based on cross-domain query).

The outline of this paper is as follows: Section 2 presents related work and Section 3 presents the algorithms of discovering the correlation between any pair of domains and recommending top-k query paths. Section 4 presents experimental results. Conclusion is presented in Section 5.

2 Related Work

Recent work has led to the development of many platforms which provide query in single domain composed of multiple data sources. In contrast, our work provides cross-domain query based on query interfaces of data sources. While Mashups integrate a set of complementary Web services and data sources [1] and COMPASS [2] assists users in their navigation through MashAPPs, Mashup which uses and combines data, presentation or functionality from two or more sources to create new services implies frequently using of open APIs [3]. The work closest to ours is the NGS framework [4], fully automated supporting queries over multiple, specialized search engines, makes use of service-enabled and XML-related technologies and ontological knowledge in the context of data mapping. Our goal is to discover correlation of any pair of domains based on the interfaces and contents of the data sources and recommend top-k query paths crossing multiple domains based on a synthesized evaluating model proposed by ourselves.

A large amount of work [5-7] has addressed top-k selection query, but they rank query results in or between relational databases. However, in our approach, our top-k selection means selecting top-k query connection paths across correlated domains. In this research, we have to take more factors into account, ranging from data source intrinsic quality, the relativity between data sources, and the selectivity which reflects the number of data sources related with the data source currently being queried.

3 Cross-Domain Query

Our cross-domain query approach can be described as follows: discovering correlation of any pair of domains and ranking source-to-source routes in each related pair of domains to recommend top-k cross-domain paths to answer users' all possible intentions according to users' input. For example, a user enters java development engineer in a job web interface to search a job, then maybe some books about java in book domain should be recommended to the user which suggests he or she read them in order to get that job. Furthermore, once this user chooses a company, information about house renting around this company is provided to the user to help him or her to decide whether this job suits him or her considering the cost for living. Before we move to the details of cross-domain query we present an assumption for doing our research more conveniently.

Assumption 1. All data sources have been clustered according to domains. Each class represents one domain. Each data source has a corresponding query interface.

3.1 Identifying the Correlation among Different Domains

Based on above assumption, if there are n domains, we use $D_1, D_2, D_3, \dots, D_n$ to represent these domains. The data source j in domain i is denoted as S_i^j . Through abundant observation, we learn that a data source is relevant to another source mostly because they share the same attributes or attribute values. Hence, in order to discover the correlation of different domains we not only investigate query interface schema which reflects data attributes but also take data source contents into account.

In terms of attribute names we build an associated undirected graph model based on data sources from different domains to describe them theoretically, in which a node represents a data source, the edge between node S_p^i in domain D_p and S_q^j in domain D_q is represented as $Edge < S_p^i, S_q^j >$ and the correlation level between S_p^i and S_q^j is represented as the weight of this edge. In order to characterize the correlation we propose two definitions.

Definition 1. Relativity: the weight of the edge $Edge < S_p^i, S_q^j >$.

Definition 2. Selectivity of S_p^i : the number of the edges from current data source S_p^i in D_p to data sources in D_q if D_p and D_q are correlated.

Though interface attribute schema of data sources from different domains is different, most share some common attribute names. For example, in Figure 1, we search a job in a job web and maybe we also want to check house renting information around the company we have chosen in the job web to help us to decide whether to choose this company. Hence, job hunting domain is connected with house renting domain. The interfaces from job hunting domain share attribute names "location" and "company" with the interfaces from house renting domain. The query results from the job search

interface can serve as the input of the interface from house renting based on the common attribute names. Hence, common attribute names will contribute to predicating the correlation of domains.

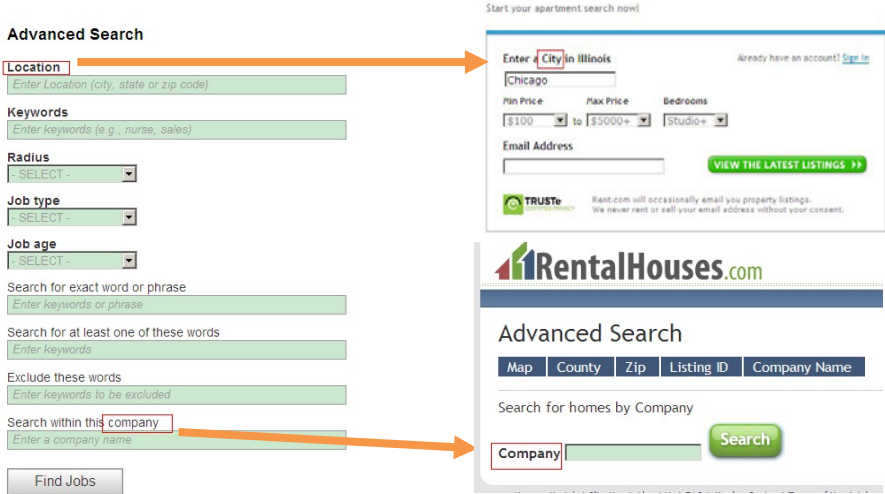


Fig. 1. Interfaces from job hunting domain and house renting domain

Thus, we analyze attribute names of an interface just through common words, because data source content is the key point to predicate correlation of a pair of data sources from different domains. We present attribute name similarity method in algorithm 1.

Algorithm 1. Attribute Names Similarity of a pair of data sources from different domains (ANS)

Input: two interfaces I_p^i, I_q^j from two data sources

Output: similarity $C < I_p^i, I_q^j >$ of two data sources

- 1 for each interface I_p^i, I_q^j do
- 2 extract attribute names
- 3 add words which are noun to set W_p^i, W_q^j

4 $C < I_p^i, I_q^j > = \frac{|W_p^i \cap W_q^j|}{|W_p^i \cup W_q^j|}$ is the number of the common words in W_p^i, W_q^j

Surfacing Deep Web Content policy (SDWC) has been addressed in Wei Liu et al.[8]. We adopt surfacing contents of each data source in the same domain and collect records as samples of its contents. However, some attribute values have common sense which can't characterize the domain of data sources. We call them non-characterized attribute values. The attribute values which can characterize the domain of data sources well are called characterized attribute values.

Algorithm 2. Correlation calculation between domains (CCD)

Input: threshold value τ_1 , parameter σ, ϕ , all domains $D_1, D_2, D_3, \dots, D_n$

Output: the relativity of all correlated pairs of data sources and the correlation of any pair of domains

```

1 for each domain  $D_j (1 \leq j \leq n)$ 
2   for each data source  $S_k^j$ 
3     extract content sample set  $SS_k^j$  by the approach SDWC
4 for each pair sample sets  $SS_p^i, SS_q^j$  corresponding with each pair correlated data
   sources  $S_p^i, S_q^j$  do
5   Remove non-characterized attribute values
6   Set the size of  $SS_p^i$  is n and the size of  $SS_q^j$  is m
7   Suppose  $R_p$  and  $R_q$  are null set
8   for(x=0; x<n; x++) do
9     enter characterized attribute values of the xth record in the query interface
       of  $S_q^j$  and the result is  $Q_x$ 
10     $R_q += Q_x \cap SS_q^j$ 
11   for(y=0; y<m; y++) do
12     enter characterized attribute values of the yth record in the query interface
       of  $S_p^i$  and the result is  $Q_y$ 
13     $R_p += Q_y \cap SS_p^i$ 
14   The attribute value similarity of  $S_p^i, S_q^j$  is  $C < VS_p^i, VS_q^j > = \frac{|R_x| + |R_y|}{|SS_p^i| + |SS_q^j|}$ 
15 for each data source  $S_k^j$  in domain  $D_j$ 
16   for each data source  $S_t^m$  in domain  $D_m ((1 \leq m \leq n) \wedge (m \neq j))$ 
17     if ( $C < I_k^j, I_t^m > \geq \tau$ )
18        $w(\text{Edge} < S_k^j, S_t^m >) = \phi(C < I_k^j, I_t^m >) / \tau + \sigma C < VS_k^j, VS_t^m >$ 
19     else  $w(\text{Edge} < S_k^j, S_t^m >) = \sigma C < VS_k^j, VS_t^m >$ 
20     if ( $w(\text{Edge} < S_k^j, S_t^m >) = 0$ )
21       the data sources  $S_k^j, S_t^m$  are not correlated and there is no edge between
       them in the correlation graph
22     if(there is no edge between any pair of data sources in two different
       domains)
23        $D_j$  and  $D_m$  are not correlated
     else  $D_j$  and  $D_m$  are correlated

```

Threshold τ means that attribute name contributes to justifying domain correlation only when it reaches a certain value. Parameter σ and ϕ are coordination factors to express the impact level of attribute name and value to the correlation between data sources.

According to algorithm 2, we build a domain correlation model as shown in Figure 2 and construct a domain correlation graph as shown in Figure 3.

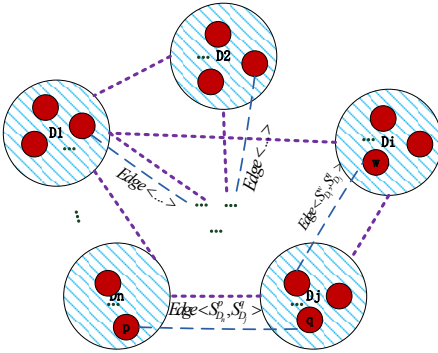


Fig. 2. The domain correlation model

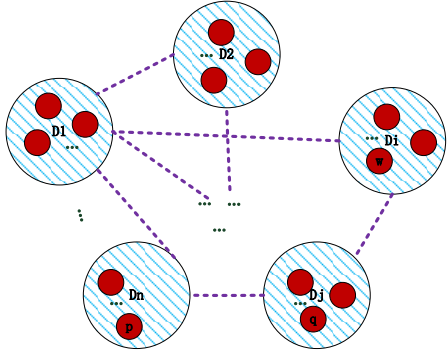


Fig. 3. The domain correlation graph

3.2 Recommending Top-k Cross-Domain Paths

After digging out the correlation of any pair of domains we can construct a query tree based on the correlation as Figure 4. The domain in which the user currently queries is considered as the first level and $depth=1$. We next move to the complete description of the algorithm for answering queries in a query tree. We use the example “query for a post of java development engineer in job hunting domain”.

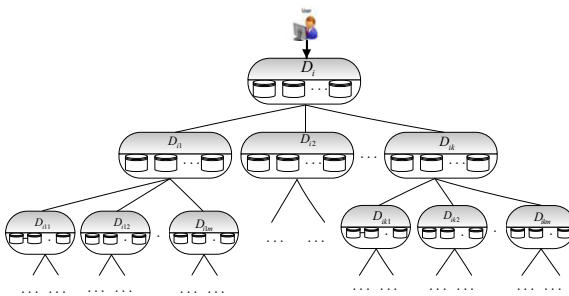


Fig. 4. Query tree based on the correlation graph

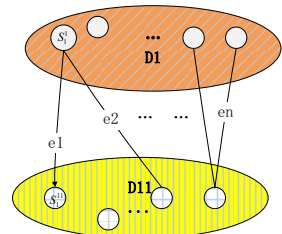


Fig. 5. The father-son model

Step 1. Input query conditions. We first input query conditions in the global query interface corresponding to the domain. For example, “java development engineer” “London” are values of keywords and location respectively in the global query interface of job hunting domain.

Step 2. According to the correlation of any pair of domains, once a query arrives, a query tree is built and the directed edges between data sources point from the data source in the current query domain to the ones in the correlated domains. In our example, we find that book shopping, house renting and training are relevant to the job hunting domain. Thus we recommend global query interfaces of these three domains according to the query tree. Then we recommend data sources which are of high quality query paths in each domain. However, generally, according to our observation and statistics, $depth \leq 2$ is in line with user query intentions in the real world. Hence, we take $depth \leq 2$ in our research.

Step 3. Heterogeneous result ranking: with potentially large numbers of results crossing multiple domains which reflect the user's multiple possible intentions, the results must be ranked based on all possible intentions before returning them to the user. The challenge of ranking, however, is how to evaluate diverse source-to-source paths in the same pair of domains. To solve this problem, we present a father-son model and propose a synthesized evaluating model aiming at this problem.

For each correlated pair of domains just as Figure 5 shows, D_1 is the father of D_{11} in the query tree. To recommend top-k paths from $e_1 \sim e_n$, we need to rank them based on a score function related with data source intrinsic quality, the selectivity and the relativity we have proposed in definition 1 and definition 2.

We design an evaluating matrix for each correlated pair of domains. The columns are the factors influencing the scores of the paths; the rows are all the paths. In the pair of domains in Figure 5, $e_1 \sim e_n$ are all rows; the columns include the quality of the father (QF) and son data sources(QC), the selectivity from father node to son node (out-degree) and from son to father (in-degree) and the relativity between father and son nodes.

Next, we present a theorem to prove that calculation is not so complicated for this matrix that we can deal in three steps referring to [9].

Theorem 1. The size of matrix is $O(N^2)$ if the size of nodes in a pair of domains is N and if the increment of nodes is Δn , then the increment of the size of matrix is $O((N + 2\Delta n)^2)$.

Proof

- (1) Suppose the number of nodes in domain D_1 is n and the number in domain D_{11} is m respectively, then maximum size of the edges between domain D_1 and D_{11} is $m \times n$. The number of the columns in this matrix is 5, the size of the rows is $m \times n$. So the size of the matrix is $5 \times m \times n$ and $(5 \times N \times n - n^2) < N^2$, hence, the size of the matrix is $O(N^2)$.
- (2) Set the increment of the nodes in D_1 is x , the increment of the size of the edges is $\Delta e = (n + x) \times (m + \Delta n - x) - m \times n = -x^2 + (m - n + \Delta n) \times x + \Delta n \times n$

When $x = (m - n + \Delta n) / 2$, Δe reaches its maximum $\frac{4 \times \Delta n \times n + (m - n + \Delta n)^2}{4} < (N + 2\Delta n)^2$.

Processing steps:

- (1) Standardization of evaluation factors. Since these factors vary in unit, dimensionality and order of magnitude which will influence the evaluation results. Standardizing these factors is necessary for unifying standards. In this paper we adopt Range Exchange to deal with it.
- (2) Quantification of vague factors. The data source quality is usually described as “good” or “bad” which we should assign values to quantify. Here, we assign the best 10, the worst 0 according to the method in [10].
- (3) Weight assignment of all the factors. The more important the factor is, the higher its weight is. The weights are normalized to be between 0 and 1 and the sum of all the weights equals 1.

Finally, we propose a synthesis evaluating model.

Suppose the matrix after standardizing is denoted as $R_{m \times n}$, the ideal values of all evaluation factors is:

$$F = (F_1, F_2 \dots F_n) = (1, 1, \dots, 1)^T. \tag{1}$$

The non-ideal value of all evaluation factors is:

$$B = (B_1, B_2 \dots B_n) = (0, 0, \dots, 0). \tag{2}$$

The weight vector is:

$$W = (W_1, W_2 \dots W_n)^T \sum_{i=1}^n W_i = 1. \tag{3}$$

If u_i^+ denotes the subsection rapport of path i to the ideal then the subsection rapport of path i to the non-ideal path is $u_i^- = 1 - u_i^+$.

The distance between the path i and the ideal path can be denoted as follows:

$$S_i^+ = u_i^+ \sqrt{\sum_{j=1}^n [W_j (1 - R_{ij})]}. \tag{4}$$

The distance between the path i and the non-ideal path can be denoted as follows:

$$S_i^- = (1 - u_i^+) \sqrt{\sum_{j=1}^n (W_j R_{ij})^2}. \tag{5}$$

To get the optimum value of the subsection rapport of path i to the ideal we give the following target function:

$$\begin{aligned}
\min = \{Z(u_i)\} &= (s_i^+)^2 + (s_i^-)^2 \\
&= (u_i^+)^2 \sum_{j=1}^n [W_j(1-R_{ij})]^2 + (1-u_i^+)^2 \sum_{j=1}^n (W_j R_{ij})^2.
\end{aligned} \tag{6}$$

Get the derivation of (6) and set it 0, then

$$u_i^+ = \frac{1}{1 + \sum_{j=1}^n [W_j(1-R_{ij})]^2 / \sum_{j=1}^n (W_j R_{ij})^2}. \tag{7}$$

Formula (7) is the synthesis evaluating function. Higher u_i^+ represents better path. Especially, $n=5$ in our model, we present formalization description in algorithm 3 based on formula (7).

Algorithm 3. Recommending top-k cross-domain paths

Input: query $Q(q_1, q_2, \dots, q_m)$, correlation between any pair of domains
 $// q_i$: query condition i

Output: top-k recommending paths in each related pair domains

- 1 Put query $Q(q_1, q_2, \dots, q_m)$ into query interfaces of the corresponding domain D_q
- 2 Pass on $Q(q_1, q_2, \dots, q_m)$ and the query results to the query interfaces of data sources in correlated domains $D_{q_1}, D_{q_2}, \dots, D_{q_x}$
- 3 **for** correlated domain D_{q_j} in $D_{q_1}, D_{q_2}, \dots, D_{q_x}$
- 4 rank the query paths from data sources in D_q to data sources in D_{q_j} in descending order according to formula (7);
- 5 recommend top-k query paths.

4 Experiments

We adopt the dataset composed of forms in the TEL-8 dataset [6] and Completeplanet and Dmoz. As Table 1 illustrates, the collected dataset consists of 305 sources covering 7 domains.

Table 1. Data sources in 7 domains

Domain	Books	Job	House Rental	Airfares	Hotel	Training	Car Rentals
number of sources	65	49	36	47	39	44	25

According to the investigation about the size of each domain, we know $N + \Delta n$ is less than 200, so we can deal with the matrix easily based on the theorem 1. Due to limit of space, we select less than 5 data sources for each domain and present the URL of the 13 data sources in table 2. We continue our example in section 3.2. “java

development engineer” and “London” are values of keywords and location in the global query interface corresponding to the job domain respectively. Figure 6 shows the query tree and the table 3 lists the relativity of each related pair of data sources according to algorithm 2. In this experiment, we set $k = 3$.

Table 2. Data sources example

domain	Abbreviation of site	web site url
Job	reed	http://www.reed.co.uk/Job/AdvancedSearch.aspx
	mon	http://jobsearch.monster.ca/StandardAdvancedSearch.aspx?cy=ca&lid=224&xms0=1
	tele	http://jobs.telegraph.co.uk/advanced-search.aspx/
book	amaz	http://www.amazon.ca/Books-Search/b/ref=sv_b_0?ie=UTF8&node=126072011
	buy	http://www.buy.com/retail/advSearch.asp?loc=106
	albk	http://www.allbookstores.com/search_advanced
training	autr	http://www.training.com.au/Pages/menuitem61dea3cd2b6b3588a392e51017a62dbc.aspx
	qa	http://www.qa.com/advanced-course-search
	new	http://www.newhorizons.com/content/courseCatalog.aspx
House rental	find	http://www.findaproperty.com/
	prim	http://www.primelocation.com/uk-property-to-rent/
	zooH	http://www.zoopla.co.uk/to-rent/
	exc	http://www.exclusiverental.ca

Table 3. The relativity of each related pair of data sources

JOB	BOOK			TRAINING			HOUSE RENTAL			
	amaz	buy	albk	autr	qa	new	find	prim	zooH	exc
reed	8.734	7.771	7.689	7.201	4.354	3.910	8.621	4.215	5.004	-
mon	7.950	6.996	8.064	7.481	4.065	-	7.241	7.540	6.958	1.548
tele	8.006	7.084	8.510	6.951	-	4.217	9.020	5.211	6.874	0.658

“-” denotes no edge between the pair of data sources.

For domain JOB-TRAINING, we present original decision matrix in table 4.

Table 4. The original decision matrix

Factors paths	QF	QC	Relativity	out-degree	in-degree
P1(reed-autr)	7.7360	8.2180	7.201	3	3
P2(reed-qa)	7.7360	7.9175	4.354	3	2
P3(reed-new)	7.7360	4.9375	3.910	3	2
P4(mon-autr)	7.4040	8.2180	7.481	2	3
P5(mon-qa)	7.4040	7.9175	4.065	2	2
P6(tele-autr)	8.0745	8.2180	6.951	2	3
P7(tele-new)	8.0745	4.9375	4.217	2	2

After standardizing the matrix by the Range Exchange, we get matrix R_i and put the matrix R_i and W into formula (7) to get u_i^+ . JOB-HOUSE RENTAL and JOB-BOOK may be deduced by analogy to get R_h, R_b, u_h^+ and u_b^+ as follows. The weight of each factor is assigned according to our experience. In this query tree, $W = (0.20, 0.20, 0.25, 0.20, 0.15)$.

$$R_v = \begin{bmatrix} 0.4952 & 1.0000 & 1.0000 & 1.0000 & 1.0000 \\ 0.4952 & 0.8760 & 0.4459 & 1.0000 & 1.0000 \\ 0.4952 & 0 & 0.3987 & 1.0000 & 1.0000 \\ 0 & 1.0000 & 0.5489 & 1.0000 & 1.0000 \\ 0 & 0.8760 & 0 & 1.0000 & 1.0000 \\ 0 & 0 & 0.6145 & 1.0000 & 1.0000 \\ 1.0000 & 1.0000 & 0.5811 & 1.0000 & 1.0000 \\ 1.0000 & 0.8760 & 0.0506 & 1.0000 & 1.0000 \\ 1.0000 & 0 & 0.8711 & 1.0000 & 1.0000 \end{bmatrix} R_i = \begin{bmatrix} 0.4952 & 1.0000 & 0.9216 & 1.0000 & 1.0000 \\ 0.4952 & 0.9084 & 0.1243 & 1.0000 & 0 \\ 0.4952 & 0 & 0 & 1.0000 & 0 \\ 0 & 1.0000 & 1.0000 & 0 & 1.0000 \\ 0 & 0.9084 & 0.0434 & 0 & 0 \\ 1.0000 & 1.0000 & 0.8516 & 0 & 1.0000 \\ 1.0000 & 0 & 0.0860 & 0 & 0 \end{bmatrix} R_h = \begin{bmatrix} 0.4952 & 0.0415 & 0.9523 & 0 & 1.0000 \\ 0.4952 & 0 & 0.4254 & 0 & 1.0000 \\ 0.4952 & 1.0000 & 0.5197 & 0 & 1.0000 \\ 0 & 0.0415 & 0.7873 & 1.0000 & 1.0000 \\ 0 & 0 & 0.8230 & 1.0000 & 1.0000 \\ 0 & 1.0000 & 0.7534 & 1.0000 & 1.0000 \\ 0 & 0.0276 & 0.1064 & 1.0000 & 0 \\ 1.0000 & 0.0415 & 1.0000 & 1.0000 & 1.0000 \\ 1.0000 & 0 & 0.5445 & 1.0000 & 1.0000 \\ 1.0000 & 1.0000 & 0.7434 & 1.0000 & 1.0000 \end{bmatrix}$$

$$u_i^+ = (0.7590 \ 0.1752 \ 0.0711 \ 0.2381 \ 0.0407 \ 0.4184 \ 0.0506)$$

$$u_h^+ = (0.1698 \ 0.0756 \ 0.2242 \ 0.2041 \ 0.2045 \ 0.3907 \ 0.0523 \ 0.4732 \ 0.3246 \ 0.9149 \ 0.1196)$$

$$u_b^+ = (0.7742 \ 0.4688 \ 0.1942 \ 0.3260 \ 0.1706 \ 0.1645 \ 0.7886 \ 0.3686 \ 0.4234)$$

According to u^+ , top-3 query paths for three domain pairs are shown in table 5.

Table 5. Top-3 query paths by the Range Exchange

Domain pair	JOB-BOOK	JOB-TRAINING	JOB-HOUSE RENTAL
Top-3	(tele-amaz)>(reed-amaz)>(reed-buy)	(reed-autr)>(tele-autr)>(mon-autr)	(tele-zooh)>(tele-find)>(mon-zooh)

Through conducting experiments on the rest data sources and domains by the same way, we do surveys and interviews in a random sample of 100 users from all walks of life and the results show 87% are satisfied with the ranked paths.

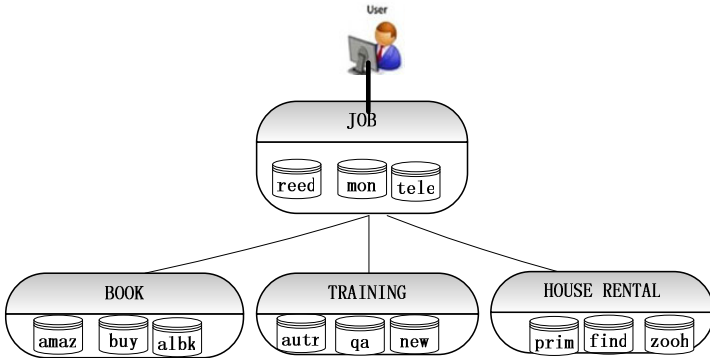


Fig. 6. The query tree of the example

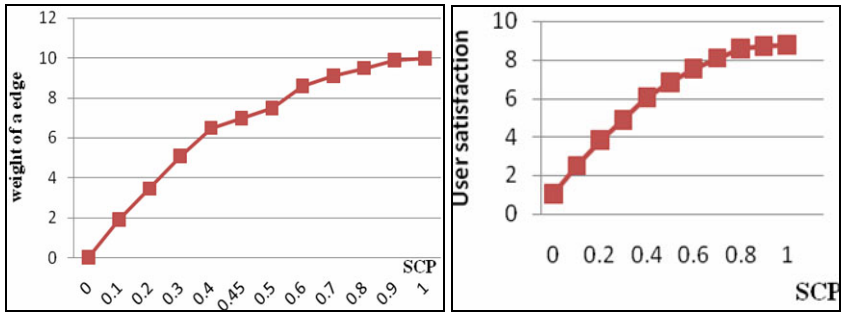


Fig. 7. The effect of SCP on weight of edges and user satisfaction

Furthermore, we construct experiment about the effect of Sample Coverage Percentage (SCP) of data source contents on user satisfaction. If a pair of data sources do not contain common contents, there is no effect. However, if they indeed share common contents, Figure 7 shows the effect of SCP on weight of edges and user satisfaction whose values are set between 0 and 10. It reflects that sampling method of data source contents also has influence on the experiment result of our method.

5 Conclusion

In this paper, we have proposed a strategy for answering cross-domain queries including discovering the correlation of any pair of domains and recommending top-k paths to users. Discovering the correlation is done based on the interface attribute names and attribute values. As to recommending top-k paths which is different from what we talk before, the paths mean the routes from one data source of one domain to one of another domain. Then recommend top-k paths from each correlated pair of domains. We propose a novel synthesized evaluating model to rank the result paths based on the quality of data sources, the relativity of a pair of data sources and the selectivity of the current starting source.

Our experiments show the effectiveness of our method of discovering the correlation. The experiment results show synthesized evaluating model is effective.

Acknowledgments. This work is supported by the National Science Foundation (60973021, 61003060), the National High-Tech Development Program (2008AA01Z146).

References

1. Deutch, D., Greenshpan, O., Milo, T.: Navigating in Complex Mashed-Up Applications. VLDB Endowment 03, 320–329 (2010)
2. Deutch, D., Greenshpan, O., Milo, T.: Navigating through mashed-up applications with compass. In: ICDE, pp. 1117–1120. IEEE Press, California (2010)
3. Mashup information, [http://en.wikipedia.org/wiki/Mashup_\(web_application_hybrid\)](http://en.wikipedia.org/wiki/Mashup_(web_application_hybrid))

4. Braga, D., Calvanese, D., Campi, A., Ceri, S., Daniel, F., et al.: NGS: a Framework for Multi-Domain Query Answering. In: ICDE Workshop, pp. 254–261. IEEE Press, Cancún (2008)
5. Ilyas, I.F., Aref, W.G., Elmagarmid, A.: Supporting top-k join queries in relational databases. In: VLDB, pp. 754–765. VLDB Endowment press, Berlin (2003)
6. Bruno, N., Gravano, L., Marian, A.: Evaluating top-k queries over web-accessible databases. In: ICDE, p. 369. IEEE Press, San Jose (2002)
7. Chang, K.C.-C., Hwang, S.w.: Minimal probing: supporting expensive predicates for top-k queries. In: SIGMOD, pp. 346–357. ACM Press, New York (2002)
8. Liu, W., Meng, X., Ling, Y.: A Web database sampling method based on graph model. *Journal of Software* 19, 179–193 (2008)
9. Xiao, G., Liu, J.: The multi-factor decision-making fuzzy evaluation model. *Statistics and Decision* 09, 12–14 (2007)
10. Bao, X., Fang, W., Zhao, P., Cui, Z., Hu, P.: A quality estimation model of Deep Web sources. *Microelectronics & Computer* 25, 47–50 (2008)

SPARQL Query Answering with RDFS Reasoning on Correlated Probabilistic Data

Chi-Cheong Szeto¹, Edward Hung¹, and Yu Deng²

- ¹ Department of Computing, The Hong Kong Polytechnic University, Hong Kong
{cscszeto, csehung}@comp.polyu.edu.hk
- ² IBM T.J. Watson Research Center, P.O. Box 704,
Yorktown Heights, NY 10598 USA
dengy@us.ibm.com

Abstract. In recent years, probabilistic models for Resource Description Framework (RDF) and its extension RDF Schema (RDFS) have been proposed to encode probabilistic knowledge. The probabilistic knowledge encoded by these models ranges from statistical relationships over possible objects of an RDF triple to relationships among correlated triples. The types of queries executed on these models vary from single triple patterns to complex graph patterns written in SPARQL, a W3C query language for RDF. Some query answerings only include reasoning of transitive properties and others do not have any reasoning. In this paper, we propose answering SPARQL queries with RDFS reasoning on probabilistic models that encode statistical relationships among correlated triples. One result to note is that although uncertainties of explicitly declared triples are specified using point probabilities, the evaluation of answers involving derived triples results in interval probabilities. Moreover, we experimentally examine how the execution time of the proposed query answering scales with the data size and the percentage of probabilistic triples in the data.

1 Introduction

Resource Description Framework (RDF) [7] is a World Wide Web Consortium (W3C) Recommendation. It is a data model that uses Uniform Resource Identifier (URI) references to identify things and uses RDF triples of the form (s, p, o) to make statements, where s , p , o are the subject, property and object respectively. RDF data is a set of RDF triples. It is also called an RDF graph because s and o of each triple represent nodes of a graph and (s, o) with label p represents a directed arc of a graph. RDF Schema (RDFS) is an extension of RDF. It provides a vocabulary to describe application-specific classes and properties, class and property hierarchies, and which classes and properties are used together. RDFS reasoning leverages the vocabulary to derive additional triples from triples explicitly declared in data. Triples in RDF data can be divided into schema and instance triples. Schema triples, whose subjects are either classes or properties, are more permanent and definitional in nature. Instance triples, whose subjects are instances, are more dynamic.

In recent years, probabilistic models for RDF(S) have been proposed to model triples that are known to be true with a certain degree of belief. Probabilistic RDF (pRDF) [11] models the statistical relationship over possible objects of a triple. Huang et al.’s probabilistic RDF database (pRDFDB) [3] models uncertainties of triples that are statistically independent of each other. Probabilistic RDFS (pRDFS) [9] assumes schema triples are always true and models statistical relationships among correlated instance triples.

The query pattern of the form (s, p, o, λ) where at most one of the members of the quadruple can be a variable is proposed [11] for pRDF. The query pattern matches a probabilistic triple if the pattern after substituting for the variable equals the triple and the probability of the triple is above λ . The triple can be explicitly declared or derived by reasoning of transitive properties. The SPARQL (a W3C query language for RDF) query [8] that has the “select” form and one basic graph pattern is proposed [3] for pRDFDB. A basic graph pattern is a set of triple patterns of the form (s, p, o) where any member of the triple can be a variable. The graph pattern matches a graph if the pattern after substituting for the variables equals the graph. The triple in the graph can be explicitly declared or derived by reasoning of transitive properties. SPARQL queries of the same type are proposed [9] for pRDFS. To match negated triples, which are unique to probabilistic models, triple patterns are extended to have the form of (s, p, o, t) where any of s , p and o can be a variable and t is a truth value. SPARQL queries having complex graph patterns are later proposed [10] for pRDFS. However, the triples in the matched graphs can only be explicitly declared.

In this paper, we propose answering SPARQL queries with RDFS reasoning on pRDFS to match against both declared and derived triples. Query answerings for pRDF [11] and pRDFDB [3] only include reasoning of transitive properties and query answerings for pRDFS [9,10] do not have any reasoning. We also propose an algorithm to compute the probability interval of a solution. The probability of a solution involving derived triples is not exact in general because we only know that a derived triple is true when any set of triples that derives it is true. A derived triple could be true or false if none of the sets of triples that derive it is true. We do not make the same assumption as pRDFDB that a derived triple is false if the sets of triples that derive it are all false.

The remainder of this paper is organized as follows. Section 2 reviews the probabilistic model pRDFS [9] and defines the entailment relationship between a pRDFS theory and a probabilistic sentence that is formed by a set of triples using negation, conjunction and disjunction. Section 3 first reviews SPARQL query evaluation on RDF data [8] and pRDFS theories without reasoning [9,10]. Then, it describes our proposed query evaluation on pRDFS theories with RDFS reasoning. Section 4 evaluates experimentally the execution time performance of our proposed query answering on top of a pRDFS theory. Finally, Section 5 concludes this paper.

2 Probabilistic Model pRDFS

2.1 Syntax

A pRDFS theory is a triple of the form (H, R, θ) . H is a set of RDF schema triples, which are assumed to be always true. R is a set of RDF instance triples, which are known to be true with a certain degree of belief. θ is a mapping from R to a set of probability distribution functions of triples in R . Note that the sets of classes, properties and individuals are assumed to be mutually disjoint. Let $\{R'_1, R'_2, \dots, R'_n\}$ be a partition of R such that any two different elements of the partition are statistically independent. τ is a truth value assignment function for the triples in R . It is from R to $\{\text{true}, \text{false}\}$, abbreviated as $\{T, F\}$. $\tau|_{R'_i}$ is the restriction of τ to R'_i . It is from R'_i to $\{T, F\}$. $P_{R'_i}$ is the joint probability distribution function of R'_i that maps $\tau|_{R'_i}$ to a probability value. θ is from R to $\{P_{R'_1}, P_{R'_2}, \dots, P_{R'_n}\}$ and maps t to $P_{R'_i}$ where $t \in R'_i$. Because of the independence assumption of R'_i , the joint probability distribution function of R , $P_R(\tau)$ can be written as $P_{R'_1}(\tau|_{R'_1}) \times P_{R'_2}(\tau|_{R'_2}) \times \dots \times P_{R'_n}(\tau|_{R'_n})$.

A pRDFS complex sentence is a pair of the form $(CS, [p_l, p_u])$. CS is a complex sentence in propositional logic where atomic sentences are RDF triples, symbol **true** and symbol **false**. It is constructed from simpler sentences using logical connectives not (\neg), and (\wedge) and or (\vee). p_l and p_u are the lower and upper bounds of the probability of CS being true respectively, where $0 \leq p_l \leq p_u \leq 1$.

2.2 Semantics

A possible world (or an RDFS interpretation) W is a set of triples that follows the semantics of RDFS except the semantics related to blank nodes, containers and datatypes. W satisfies an RDF triple r iff $r \in W$. W satisfies a set of RDF triples R iff W satisfies every $r \in R$. A set of triples is consistent if it has a satisfying interpretation. A set of triples R rdfs-entails (\models_{rdfs}) a triple r iff every satisfying interpretation of R is a satisfying interpretation of r . A set of triples $R \models_{rdfs}$ another set of triples R' iff every satisfying interpretation of R is a satisfying interpretation of R' .

Let Ω be the set of all possible worlds. A pRDFS interpretation is a mapping $I : \Omega \rightarrow [0, 1]$, such that $\sum_{W \in \Omega} I(W) = 1$. I satisfies a pRDFS theory (H, R, θ) iff $P_R(\tau) = \sum_{W \in \Omega \mid H \subseteq W \wedge \forall t \in R(\tau(t)=T \implies t \in W \wedge \tau(t)=F \implies t \notin W)} I(W)$ for any truth value assignment τ . A pRDFS theory is consistent if it has a satisfying interpretation.

An RDF triple t is true in a world W if $t \in W$ and is false if $t \notin W$. Symbol **true** is true in all worlds whereas symbol **false** is false in all worlds. The truth value of a complex sentence CS can be determined recursively from the truth values of simpler sentences and the truth tables of the logical connectives. A pRDFS interpretation I satisfies a pRDFS complex sentence $(CS, [p_l, p_u])$ iff $p_l \leq \sum_{W \in \Omega \mid CS \text{ is true in } W} I(W) \leq p_u$. A pRDFS theory $(H, R, \theta) \models_{prdfs}$ a pRDFS complex sentence $(CS, [p_l, p_u])$ iff every satisfying interpretation of (H, R, θ) is a satisfying interpretation of $(CS, [p_l, p_u])$.

3 pRDFS Query Evaluation

In this section, we first review the evaluation of a SPARQL query on RDF data [8] in Section 3.1 and on a pRDFS theory without reasoning [9,10] in Section 3.2. Then, we describe our proposed query answering with RDFS reasoning on a pRDFS theory in Section 3.3.

3.1 Query Evaluation on RDF Data

In the discussion of SPARQL queries, we focus on the “select” query form. In SPARQL syntax, the query pattern is described in the “where” clause and complex graph patterns can be formed using keywords “optional”, “union” and “filter”. In [8], a procedure is defined for converting a query pattern in SPARQL syntax into a SPARQL algebra expression, that uses operators “BGP”, “Filter”, “Join”, “LeftJoin” and “Union”. The following discusses these operators. Basic graph patterns are basic building blocks of an algebra expression. Let V be the set of query variables. A triple pattern is a member of the set $(\mathbb{U} \cup V) \times (\mathbb{U} \cup V) \times (\mathbb{U} \cup \mathbb{L} \cup V)$, where \mathbb{U} is a set of URI references and \mathbb{L} is a set of literals. A basic graph pattern G is a set of triple patterns. A solution μ is a partial function from V to $(\mathbb{U} \cup \mathbb{L})$. μ is a solution for G from data D if $\mu(G)$ is a subgraph of D , where $\mu(G)$ denotes a set of triples obtained by replacing every variable v in G with $\mu(v)$. The operator $\text{BGP}(G)$ returns a set of all solutions for the basic graph pattern G , that is, $\text{BGP}(G) = \{\mu \mid \mu(G) \subseteq D\}$. The operator $\text{Filter}(\text{expr}, \Omega)$ removes any solution μ where the filter expression $\text{expr}(\mu)$ evaluates to false from a multiset of solutions Ω , that is, $\text{Filter}(\text{expr}, \Omega) = \{\mu \in \Omega \mid \text{expr}(\mu) = \text{true}\}$. Operators $\text{Join}(\Omega_1, \Omega_2)$, $\text{LeftJoin}(\Omega_1, \Omega_2, \text{expr})$ and $\text{Union}(\Omega_1, \Omega_2)$ combine two multisets of solutions Ω_1 and Ω_2 . Two solutions μ_1 and μ_2 are compatible if $\mu_1(v) = \mu_2(v)$ for every variable v in both the domain of μ_1 and the domain of μ_2 . $\text{Join}(\Omega_1, \Omega_2) = \{\mu_1 \cup \mu_2 \mid \mu_1 \in \Omega_1 \text{ and } \mu_2 \in \Omega_2 \text{ are compatible}\}$. $\text{LeftJoin}(\Omega_1, \Omega_2, \text{expr}) = \{\mu_1 \cup \mu_2 \mid \mu_1 \in \Omega_1 \text{ and } \mu_2 \in \Omega_2 \text{ are compatible} \wedge \text{expr}(\mu_1 \cup \mu_2) = \text{true}\} \cup \{\mu_1 \mid \mu_1 \in \Omega_1 \text{ and } \mu_2 \in \Omega_2 \text{ are compatible} \wedge \text{expr}(\mu_1 \cup \mu_2) = \text{false}\} \cup \{\mu_1 \mid \mu_1 \in \Omega_1 \text{ and } \mu_2 \in \Omega_2 \text{ are not compatible}\}$. Finally, $\text{Union}(\Omega_1, \Omega_2) = \{\mu \mid \mu \in \Omega_1 \vee \mu \in \Omega_2\}$.

A multiset of solutions for a query pattern will be converted into a sequence of solutions. The sequence contains the same solutions as the multiset and the solutions are arranged in any order. In SPARQL algebra, the operator $\text{ToList}(\Omega)$ converts a multiset of solutions Ω into a sequence. In SPARQL syntax, the sequence of solutions can be modified using keywords “order by”, “select”, “distinct”, “limit” and “offset”. The corresponding operators in SPARQL algebra are “OrderBy”, “Project”, “Distinct” and “Slice”. These operators are applied in the order listed. The operator $\text{OrderBy}(\Psi, \text{ordering condition})$ changes the order of a sequence of solutions Ψ to satisfy the ordering condition. The operator $\text{Project}(\Psi, PV)$ changes every solution μ in a sequence of solutions Ψ to be the restriction of μ to PV , where PV is the set of variables mentioned in the “select” clause. The operator $\text{Distinct}(\Psi)$ eliminates duplicate solutions in a sequence of solutions Ψ . The operator $\text{Slice}(\Psi, s, l)$ returns a sub-sequence of

a sequence of solutions Ψ . The sub-sequence contains the s th to $(s + l - 1)$ th solutions of Ψ . The result of a query is a sequence of solutions obtained after the above operators are applied.

3.2 Query Evaluation on a pRDFS Theory without Reasoning

The result of a query on a pRDFS theory (H, R, θ) is a sequence of pairs. The first element of a pair is a solution μ , which represents that the query pattern after substituting for variables matches alternative subgraphs Γ_i of the possible worlds of the theory. The second element of the pair is the probability of the solution $P(\mu)$, which refers to the probability of the presence of any of the subgraphs Γ_i .

The query evaluation is as follows. We first obtain a sequence of intermediate solutions by matching the query pattern against $H \cup R$. To find the subgraph Γ that is matched for each intermediate solution μ , we number all basic graph patterns G_i in the query. Let the number of basic graph patterns in the query be N . We create a set of N variables $\{v_1, v_2, \dots, v_N\}$ that is disjoint from the original set of query variables V . Then, we replace $\text{BGP}(G_i)$ in the algebra expression of the query pattern with $\text{Join}(\text{BGP}(G_i), \{(v_i, 1)\})$ for all i so that if the solution μ is obtained from G_i , the domain of the solution denoted by $\text{dom}(\mu)$ will contain the variable v_i . Hence, the subgraph matched for the solution μ , $\Gamma = \bigcup_{i: v_i \in \text{dom}(\mu)} \mu(G_i)$.

Let the sequence of intermediate solutions (before applying operators “Project”, “Distinct” and “Slice”) be $(\mu_1, \mu_2, \dots, \mu_M)$ and the corresponding sequence of subgraphs matched for the solutions be $(\Gamma_1, \Gamma_2, \dots, \Gamma_M)$, where M is the number of intermediate solutions. First, the operator “Project” is applied to the intermediate solution sequence to have $\text{Project}((\mu_1, \mu_2, \dots, \mu_M), PV) = (\mu_1|_{PV}, \mu_2|_{PV}, \dots, \mu_M|_{PV})$, where PV is the set of variables mentioned in the “select” clause. Then, the operator “Distinct” is applied. $\text{Distinct}((\mu_1|_{PV}, \mu_2|_{PV}, \dots, \mu_M|_{PV})) = (\mu'_1, \mu'_2, \dots, \mu'_{M'})$, where $M' \leq M$ is the number of intermediate solutions after duplicate solutions are eliminated. Finally, the operator “Slice” is applied to produce the final solution sequence $\text{Slice}((\mu'_1, \mu'_2, \dots, \mu'_{M'}), s, l) = (\mu'_s, \mu'_{s+1}, \dots, \mu'_{s+l-1})$. The probability of the final solution μ'_i , $P(\mu'_i) = P(\bigvee_{j: \mu'_i = \mu'_j|_{PV}} \Gamma_j)$.

In general, the probability of a solution $P(\mu)$ is of the form $P(\bigvee_i \Gamma_i)$ where each Γ_i is a subgraph matched for the solution μ . If the number of Γ_i is 1, $P(\mu)$ can be computed exactly using the probability distribution function of R . $P(\mu) = P(\Gamma) = P_R(\{(s, p, o, T) \mid (s, p, o, T) \in \Gamma\} \cup \{(s, p, o, F) \mid (s, p, o, F) \in \Gamma\})$. Otherwise, the calculation of $P(\mu)$ is formulated as a disjunctive normal form (DNF) probability problem with the DNF formula $F = \bigvee_i (\bigwedge_{(s,p,o,T) \in \Gamma_i} (s, p, o) \wedge \bigwedge_{(s,p,o,F) \in \Gamma_i} \neg(s, p, o))$. The coverage algorithm [5] is used to compute an estimate $\tilde{P}(\mu)$ such that the probability $P((1 - \epsilon)P(\mu) \leq \tilde{P}(\mu) \leq (1 + \epsilon)P(\mu)) \geq 1 - \delta$, where ϵ and δ are input parameters. It is a polynomial time Monte-Carlo algorithm with respect to the number of conjunctions of literals¹ of F .

¹ A literal here is an RDF triple or a negated triple.

Table 1. A truth table for a derived instance triple e and its negation $\neg e$. A is the disjunction of all justifications for e in $(H \cup R)$, where (H, R, θ) is a pRDFS theory.

A	$\neg A$	e	$\neg e$
true	false	true	false
false	true	true or false	true or false

3.3 Query Evaluation on a pRDFS Theory with RDFS Reasoning

To support RDFS reasoning, two changes in the query evaluation are made. The first one is graph pattern matching. Graphs entailed by $(H \cup R)$ instead of subgraphs of $(H \cup R)$ are sought. The operation of $\text{BGP}(G)$ is changed to finding solutions $\{\mu \mid (H \cup R) \models_{\text{rdfs}} \mu(G)\}$ instead of $\{\mu \mid (H \cup R) \supseteq \mu(G)\}$.

The second one is the probability calculation. In general, we do not have enough information to compute the exact probability of the DNF formula F if F contains derived instance triples. The probability bounds of F are computed instead and a member of the query result is of the form $(s, [P_l(s), P_u(s)])$, where $P_l(s)$ and $P_u(s)$ are the lower and upper bounds of $P(s)$ respectively.

A justification for a triple t is a set of triples that derives t . This term in OWL-DL [6] is formally defined in [4] to provide a debugging service. We define it in RDFS data similarly as follows.

Definition 1 (Justification). *A set $J \subseteq R$ is a justification for a triple t in a set of triples R if $J \models_{\text{rdfs}} t$ and $J' \not\models_{\text{rdfs}} t$ for every $J' \subset J$.*

Given a pRDFS theory (H, R, θ) , let e be a derived triple, $\text{JUST}(e, H \cup R)$ be the set of all justifications for e in $(H \cup R)$ and $A = \bigvee_{J \in \text{JUST}(e, H \cup R)} \bigwedge_{j \in J} j$ be the disjunction of all justifications. Table 1 is a truth table for e and $\neg e$ given the truth value of A . If A is true, e is also true by RDFS reasoning. However, if A is false, e could be true or false. The truth value of e is not specified in the theory. Therefore, the minimum probability of e equals the probability of A and the maximum probability of e is 1. Similarly, we only know that $\neg e$ is false by RDFS reasoning if $\neg A$ is false. However, $\neg e$ could be true or false if $\neg A$ is true. Therefore, the minimum probability of $\neg e$ is 0 and the maximum probability of $\neg e$ equals the probability of $\neg A$.

Algorithms 1 and 2 compute the approximate lower and upper probability bounds of a DNF formula respectively. They assume that the probabilities of conjunctions of literals of the DNF formula can be independently optimized. That is, the probability of a DNF formula is maximized (or minimized) if the probabilities of its conjunctions of literals are maximized (or minimized). Therefore, these algorithms cannot be applied to one type of DNF formulas, where a derived triple is in one conjunction of literals and the negation of the same derived triple is in another conjunction of literals. $(r_1 \wedge e) \vee (r_2 \wedge \neg e)$ is an example of this type of DNF formulas, where r_1, r_2 are declared instance triples and e is a derived instance triple.

Algorithm 1. Compute the approximate lower probability bound of a DNF formula F given a pRDFS theory (H, R, θ)

- 1: Replace every negation of a derived instance triple $\neg e$ in F with symbol **false**.
 - 2: Replace every derived instance triple e in the resulting formula with $\bigvee_{J \in \text{JUST}(e, H \cup R)} \bigwedge_{j \in J} j$.
 - 3: Replace every schema triple in the resulting formula with symbol **true**.
 - 4: Convert the resulting formula into disjunctive normal form.
 - 5: Use the coverage algorithm [5] to find the approximate probability of the resulting formula and return it as the result.
-

Algorithm 2. Compute the approximate upper probability bound of a DNF formula F given a pRDFS theory (H, R, θ)

- 1: Replace every negation of a derived instance triple $\neg e$ in F with $\neg(\bigvee_{J \in \text{JUST}(e, H \cup R)} \bigwedge_{j \in J} j)$.
 - 2: Replace every derived instance triple e in the resulting formula with symbol **true**.
 - 3: Replace every schema triple in the resulting formula with symbol **true**.
 - 4: Convert the resulting formula into disjunctive normal form.
 - 5: Use the coverage algorithm [5] to find the approximate probability of the resulting formula and return it as the result.
-

Example 1 (Calculation of probability bounds). Let (H, R, θ) be a pRDFS theory, where $H = \{h_1, h_2, h_3, h_4\}$ and $R = \{r_1, r_2, \dots, r_5\}$. The DNF formula for a solution s is $F = (e_1 \wedge \neg r_2) \vee (r_1 \wedge \neg e_2)$, where e_1 and e_2 are derived instance triples. $\text{JUST}(e_1, H \cup R) = \{\{h_2, r_2\}, \{h_3, r_3\}\}$ and $\text{JUST}(e_2, H \cup R) = \{\{h_1, r_1, r_4\}, \{h_4, r_5\}\}$. To find the lower bound of $P(s)$, Algorithm [1] replaces $\neg e_2$ with symbol **false**, e_1 with its disjunction of all justifications $((h_2 \wedge r_2) \vee (h_3 \wedge r_3))$ and schema triples with symbol **true**. The resulting formula is $((\text{true} \wedge r_2) \vee (\text{true} \wedge r_3)) \wedge \neg r_2 \vee (r_1 \wedge \text{false}) = (\neg r_2 \wedge r_3)$, which has only one conjunction of literals. Its probability can be computed exactly using the probability distribution function of R . To find the upper bound of $P(s)$, Algorithm [2] replaces $\neg e_2$ with its negated disjunction of all justifications $\neg((h_1 \wedge r_1 \wedge r_4) \vee (h_4 \wedge r_5))$ and replaces e_1 and schema triples with symbol **true**. The resulting formula is $(\text{true} \wedge \neg r_2) \vee (r_1 \wedge \neg((\text{true} \wedge r_1 \wedge r_4) \vee (\text{true} \wedge r_5))) = \neg r_2 \vee (r_1 \wedge \neg r_4 \wedge \neg r_5)$. The coverage algorithm is applied to this formula to find the approximate probability value.

The disjunction of all justifications for a triple e , $A = \bigvee_{J \in \text{JUST}(e, H \cup R)} \bigwedge_{j \in J} j$ in Algorithms [1] and [2] can be simplified by replacing $\text{JUST}(r, H \cup R)$ with its subset $\text{JUST}'(r, H \cup R) = \{J \in \text{JUST}(r, H \cup R) \mid \nexists J' \in \text{JUST}(r, H \cup R) (J \models_{rdfs} J' \wedge J \neq J')\}$. It can be proved that the set of possible worlds that A is true is the same after replacing JUST with JUST' . JUST' can be further simplified by removing schema triples, which are assumed to be true in all possible worlds. The final simplified set is $\text{JUST}''(r, H \cup R) = \{J \cap R \mid J \in \text{JUST}'(r, H \cup R)\}$.

Algorithm [3] finds the set $\text{JUST}''(r, H \cup R)$, which is stored in the final value of \mathbb{R} . Lines 4-26 handle triples with property `rdf:type` while Lines 28-29 handle triples the properties of which are not `rdf:type`. In both cases, backward chaining

Algorithm 3. Find $\text{JUST}''(r, H \cup R)$ where $r = (s, p, o)$ is a derived triple, H is a set of schema triples and R is a set of instance triples

```

1:  $\mathbb{R} \leftarrow \emptyset$ .
2:  $H \leftarrow H \cup$  transitive closures of rdfs:subClassOf and rdfs:subPropertyOf.
3: if  $p = \text{rdf:type}$  then
4:   /* rule rdfs9 */
5:    $\mathbb{R} \leftarrow \mathbb{R} \cup \{(s, p, o') \subseteq R \mid (o', \text{rdfs:subClassOf}, o) \in H \wedge o \neq o'\}$ .
6:   /* combination of rules rdfs9 and rdfs2 */
7:    $O \leftarrow \{o' \mid (s, p, o') \notin R \wedge (o', \text{rdfs:subClassOf}, o) \in H \wedge o \neq o'\}$ .
8:    $\mathbb{R} \leftarrow \mathbb{R} \cup \{(s, p', o'') \subseteq R \mid o' \in O \wedge (p', \text{rdfs:domain}, o') \in H\}$ .
9:   /* combination of rules rdfs9, rdfs2 and rdfs7 */
10:   $P \leftarrow \{p' \mid o' \in O \wedge \nexists o''((s, p', o'') \in R) \wedge (p', \text{rdfs:domain}, o') \in H\}$ .
11:   $\mathbb{R} \leftarrow \mathbb{R} \cup \{(s, p'', o') \subseteq R \mid p' \in P \wedge (p'', \text{rdfs:subPropertyOf}, p') \in H \wedge p' \neq p''\}$ .
12:  /* combination of rules rdfs9 and rdfs3 */
13:   $\mathbb{R} \leftarrow \mathbb{R} \cup \{(s', p', s) \subseteq R \mid o' \in O \wedge (p', \text{rdfs:range}, o') \in H\}$ .
14:  /* combination of rules rdfs9, rdfs3 and rdfs7 */
15:   $P \leftarrow \{p' \mid o' \in O \wedge \nexists s'((s', p', s) \in R) \wedge (p', \text{rdfs:range}, o') \in H\}$ .
16:   $\mathbb{R} \leftarrow \mathbb{R} \cup \{(s', p'', s) \subseteq R \mid p' \in P \wedge (p'', \text{rdfs:subPropertyOf}, p') \in H \wedge p' \neq p''\}$ .
17:  /* rule rdfs2 */
18:   $\mathbb{R} \leftarrow \mathbb{R} \cup \{(s, p', o') \subseteq R \mid (p', \text{rdfs:domain}, o) \in H\}$ .
19:  /* combination of rules rdfs2 and rdfs7 */
20:   $P \leftarrow \{p' \mid \nexists o'((s, p', o') \in R) \wedge (p', \text{rdfs:domain}, o) \in H\}$ .
21:   $\mathbb{R} \leftarrow \mathbb{R} \cup \{(s, p'', o') \subseteq R \mid p' \in P \wedge (p'', \text{rdfs:subPropertyOf}, p') \in H \wedge p' \neq p''\}$ .
22:  /* rule rdfs3 */
23:   $\mathbb{R} \leftarrow \mathbb{R} \cup \{(s', p', s) \subseteq R \mid (p', \text{rdfs:range}, o) \in H\}$ .
24:  /* combination of rules rdfs3 and rdfs7 */
25:   $P \leftarrow \{p' \mid \nexists s'((s', p', s) \in R) \wedge (p', \text{rdfs:range}, o) \in H\}$ .
26:   $\mathbb{R} \leftarrow \mathbb{R} \cup \{(s', p'', s) \subseteq R \mid p' \in P \wedge (p'', \text{rdfs:subPropertyOf}, p') \in H \wedge p' \neq p''\}$ .
27: else
28:   /* rule rdfs7 */
29:    $\mathbb{R} \leftarrow \mathbb{R} \cup \{(s, p', o) \subseteq R \mid (p', \text{rdfs:subPropertyOf}, p) \in H \wedge p \neq p'\}$ .
30: end if
31: return  $\mathbb{R}$ .

```

approach is used to find the sets of triples each of which derives r . The former uses combinations of rules `rdfs2`, `rdfs3`, `rdfs7` and `rdfs9` (listed in the following) while the latter uses rule `rdfs7`.

1. $\frac{(a, \text{rdfs:domain}, x) (u, a, y)}{(u, \text{rdf:type}, x)}$ `rdfs2`
2. $\frac{(a, \text{rdfs:range}, x) (u, a, v)}{(v, \text{rdf:type}, x)}$ `rdfs3`
3. $\frac{(a, \text{rdfs:subPropertyOf}, b) (u, a, y)}{(u, b, y)}$ `rdfs7`
4. $\frac{(a, \text{rdfs:subClassOf}, x) (v, \text{rdf:type}, u)}{(v, \text{rdf:type}, x)}$ `rdfs9`

Table 2. Number of solutions obtained by executing the queries with different reasoning against the LUBM data of one-department size

Query	1	2	3	4	5	6	7	8	9	10	11	12	13	14
no reasoning	4	0	6	0	0	0	0	0	0	0	0	0	0	532
RDFS reasoning	4	0	6	34	719	571	61	571	8	0	0	0	0	532
OWL-DL reasoning	4	0	6	34	719	678	67	678	13	4	10	1	1	532

4 Experimental Study

This section examines how the execution time of our proposed query answering on a pRDFS theory scales with the data size and the percentage of probabilistic triples in the data. The data that we use is the Lehigh University Benchmark (LUBM) [2], which has a schema [2] for the university domain, and an instance data generator. The generator can output data of different sizes. The instance triples do not contain any blank nodes. This satisfies the assumption of pRDFS.

To generate the probabilistic knowledge, we randomly select four triples each time from the instance data. We assume these four triples are statistically correlated and generate a random probability distribution function for them. The probabilistic knowledge is encoded in RDF triples using a specialized vocabulary [3].

The LUBM has 14 sample queries, all of which are of the “select” form and contain one basic graph pattern. Table 2 shows the number of solutions obtained by executing the queries with and without RDFS reasoning against the LUBM data of one-department size. It illustrates the usefulness of having reasoning during query answering. Queries 10-13 are excluded from the experiments because OWL-DL [6] reasoning is required to produce solutions. Finally, all triple patterns of the queries are set to have the truth value of true.

The experiments were carried out on a computer with an Intel Core 2 Duo processor E6320 and 2 GB memory. The software code was written in Java. Jena [4], which is an RDF toolkit including an RDF/XML parser, reasoners and a SPARQL query engine, was used.

Two experiments were performed. In the first one, we varied the number of departments from 1 to 8, and kept the percentage of probabilistic triples at 10%. Fig. 1 shows the total execution time of answering LUBM sample queries. It also shows the time of two major steps, graph pattern matching and probability calculation. The parameters for approximating the probability ϵ and δ are both set to 0.01. That is, the probability that the approximate value differs from the true value by more than 1% is less than 0.01. The total execution time scales linearly with the data size in 6 out of 10 queries (Queries 1, 3, 4, 5, 6, 14), and scales polynomially or exponentially in other queries. In 6 out of 10 queries (Queries 1, 2, 3, 4, 5, 7), the time of calculating probability (Algorithms 1 and 2)

² <http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl>

³ <http://www.comp.polyu.edu.hk/~cscszeto/prdfs/prdfs.owl>

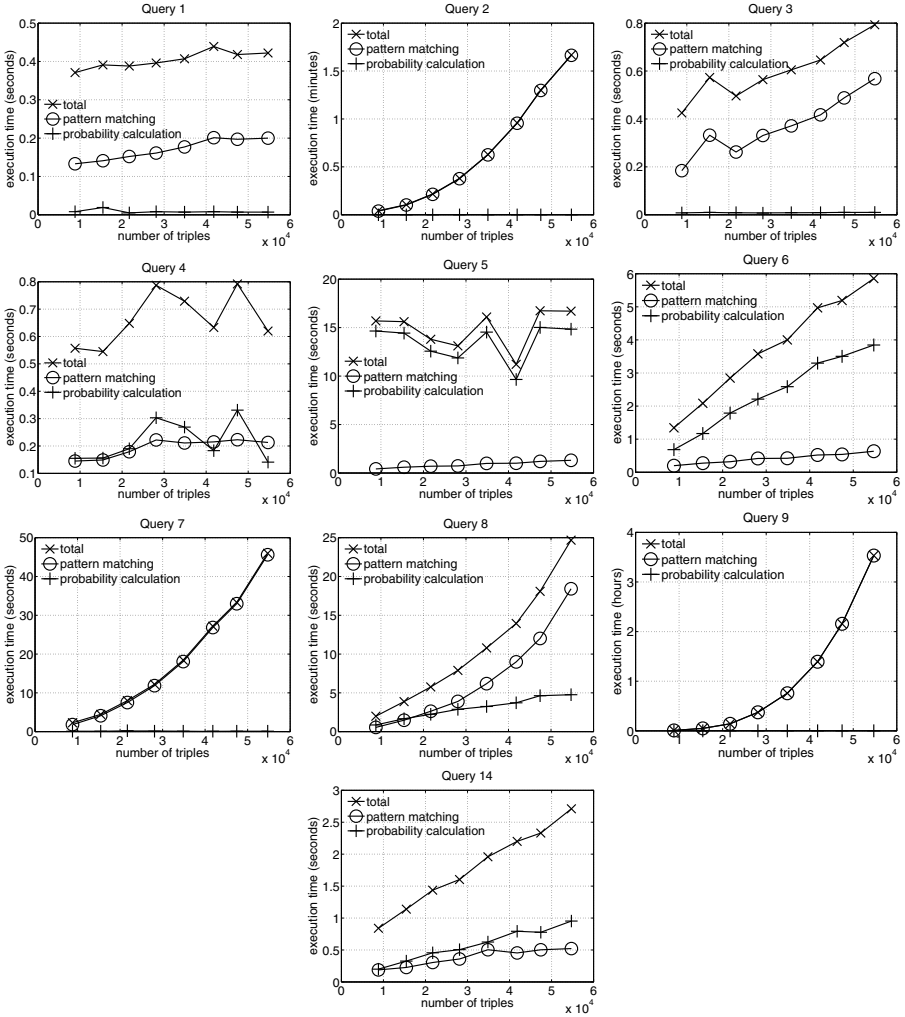


Fig. 1. Execution time of answering LUBM sample queries as the number of schema and instance triples increases

does not increase with the data size. In the rest of queries, it increases linearly because the number of solutions to these queries also increases linearly. The polynomial or exponential scaling behavior of the total time is due to the step of graph pattern matching.

In the second experiment, we varied the percentage of probabilistic triples from 10% to 80% and kept the number of departments at 1. Fig. 2 shows that the time of pattern matching does not change with the percentage of probabilistic triples. This is expected since the time does not depend on the percentage of probabilistic triples. In all queries except Queries 4 and 5, the time of

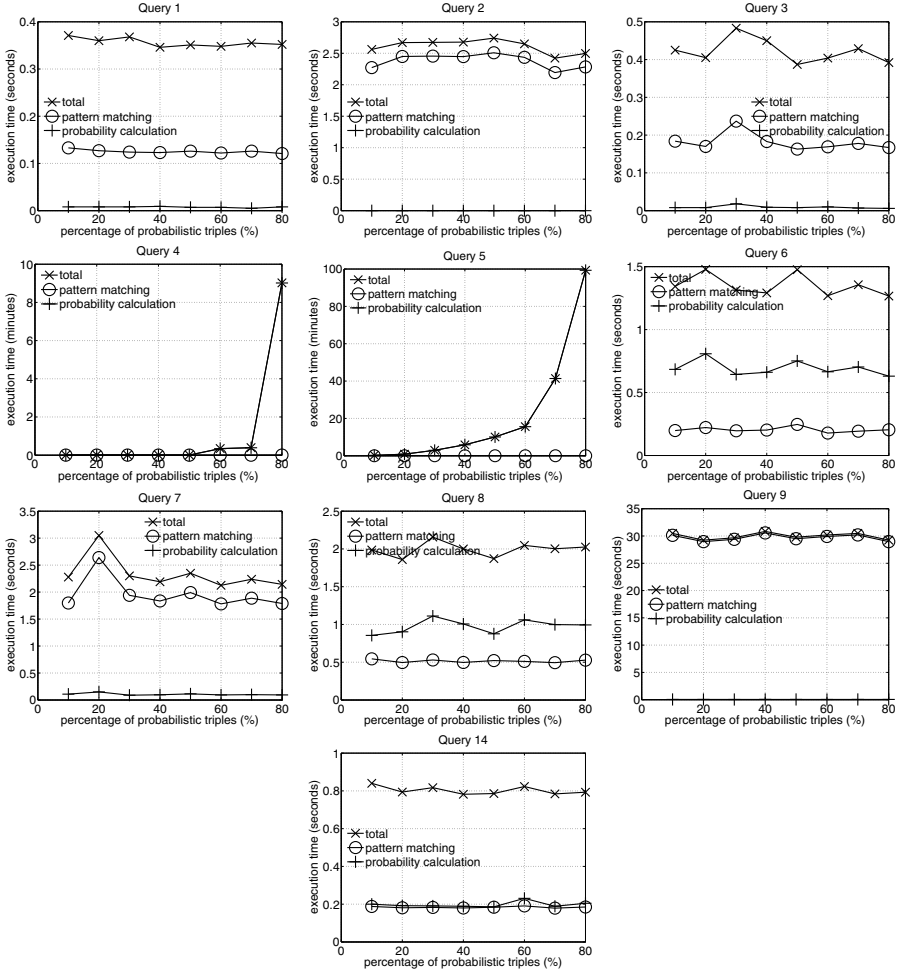


Fig. 2. Execution time of answering LUBM sample queries as the percentage of probabilistic triples increases

calculating probability (Algorithms 1 and 2) does not vary with the percentage of probabilistic triples. In Queries 4 and 5, the time increases with the percentage of probabilistic triples. It increases rapidly when the percentages of probabilistic triples increase beyond 70% and 60% respectively. The query patterns of these two queries match derived triples that have more than one justification, so the length of the DNF formula is larger than 1. If the percentage of probabilistic triples is low, the DNF formula can be simplified to one with a shorter length. As the percentage of probabilistic triples increases, the length of the formula increases. Therefore, the time of computing probability also increases.

5 Conclusion

In this paper, we have proposed answering SPARQL queries with RDFS reasoning on pRDFS to match against both declared and derived triples, and an algorithm to compute the probability interval of a solution that involves derived triples. When the percentage of probabilistic triples is low, our experimental study shows that the time of calculating probability increases linearly with the data size and does not vary with the percentage of probabilistic triples. However, when the percentage of probabilistic triples is high, the time of computing probability could increase with the percentage of probabilistic triples.

Acknowledgments. The work described in this paper was partially supported by grants from the Research Grants Council of the Hong Kong Special Administrative Region, China (PolyU 5174/07E, PolyU 5181/06E, PolyU 5182/08E, PolyU 5191/09E).

References

1. Carroll, J.J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., Wilkinson, K.: Jena: implementing the semantic web recommendations. In: Feldman, S.I., Uretsky, M., Najork, M., Wills, C.E. (eds.) WWW (Alternate Track Papers & Posters), pp. 74–83. ACM, New York (2004)
2. Guo, Y., Pan, Z., Heflin, J.: An evaluation of knowledge base systems for large OWL datasets. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 274–288. Springer, Heidelberg (2004)
3. Huang, H., Liu, C.: Query evaluation on probabilistic RDF databases. In: Vossen, G., Long, D.D.E., Yu, J.X. (eds.) WISE 2009. LNCS, vol. 5802, pp. 307–320. Springer, Heidelberg (2009)
4. Kalyanpur, A.: Debugging and repair of OWL ontologies. Ph.D. thesis, University of Maryland at College Park, College Park, MD, USA (2006)
5. Karp, R.M., Luby, M., Madras, N.: Monte-carlo approximation algorithms for enumeration problems. *Journal of Algorithms* 10(3), 429–448 (1989)
6. OWL 2 web ontology language, <http://www.w3.org/TR/owl2-overview/>
7. Resource description framework (RDF), <http://www.w3.org/RDF/>
8. SPARQL query language for RDF, <http://www.w3.org/TR/rdf-sparql-query/>
9. Szeto, C.C., Hung, E., Deng, Y.: Modeling and querying probabilistic RDFS data sets with correlated triples. In: Du, X., Fan, W., Wang, J., Peng, Z., Sharaf, M.A. (eds.) APWeb 2011. LNCS, vol. 6612, pp. 333–344. Springer, Heidelberg (2011)
10. Szeto, C.C., Hung, E., Deng, Y.: Modeling and querying probabilistic RDFS data sets with correlated triples (extended version), technical report, The Hong Kong Polytechnic University (2011), http://www.comp.polyu.edu.hk/~cscszeto/publications/prdfs_tr.pdf
11. Udrea, O., Subrahmanian, V.S., Majkic, Z.: Probabilistic RDF. In: IRI, pp. 172–177. IEEE Systems, Man, and Cybernetics Society (2006)

Probabilistic Threshold Join over Distributed Uncertain Data^{*}

Lei Deng, Fei Wang, and Benxiong Huang

Department of Electronics and Information Engineering, Huazhong University of Science and Technology, 430074 Wuhan, China
denglei86@gmail.com, {wangfei, huangbx}@hust.edu.cn

Abstract. Large amount of uncertain data is collected by many emerging applications which contain multiple sources in a distributed manner. Previous efforts on querying uncertain data in distributed environment have only focus on ranking and skyline, join queries have not been addressed in earlier work despite their importance in databases. In this paper, we address distributed probabilistic threshold join query, which retrieves results satisfying the join condition with combining probabilities that meet the threshold requirement from distributed sites. We propose a new kind of bloom filters called *Probability Bloom Filters* (PBF) to represent set with probabilistic attribute and design a PBF based Bloomjoin algorithm for executing distributed probabilistic threshold join query with communication efficiency. Furthermore, we provide theoretical analysis of the network cost of our algorithm and demonstrate it by simulation. The experiment results show that our algorithm can save network cost efficiently by comparing to original Bloomjoin algorithm in most scenarios.

Keywords: Distributed query processing, joins, uncertain data, Bloom filters.

1 Introduction

Due to the increasing number of available data sources and the available network service, a large amount of uncertain data is collected by many emerging applications which contain multiple sources in a distributed manner, such as distributed sensor networks [1], and multiple geographically distant data integration [2]. Extensive research efforts have been given for querying uncertain data. While these works produce insightful results, they typically assume a centralized rather than a distributed environment setting. Until recently, there appears some works target at distributed queries on uncertain data, such as ranking [3, 4] and skyline [5]. However, none of existing works deal with the join queries despite their significance in database.

In uncertain databases, data items are usually associated with probabilistic values which denote the probabilities of specific data items, and probabilistic values are also

^{*} This work was supported by the National Natural Science Foundation of China (NSFC) under grant No. 61001070. We would like to thank anonymous reviewers for the insightful comments.

assigned to query results based on combining probabilities from the input data [6]. Since users want to get results with higher possibilities, they may wish to apply a threshold on result's probabilistic values. Threshold queries on uncertain have attracted much attention in recent works [7, 8]. Agrawal *et al.* [9] study the problem of threshold join queries on uncertain data, but they assume a centralized setting.

T ₁		
TID	Name	Possibility
t1	Honda	0.9
t2	Toyota	0.6
t3	BMW	0.7
t4	Ford	0.5

T ₂		
TID	Speed(mph)	Possibility
t1	90	0.9
t2	80	0.8
t4	60	0.8
t5	100	0.7

Join Result			
Name	Speed(mph)	Possibility	
Honda	90	0.9	τ=0.8
Toyota	80	0.7	
Ford	60	0.65	

Fig. 1. An example of probabilistic threshold join query

In this paper, for the first time, we study the problem of probabilistic threshold join query in distributed environment. A probabilistic threshold query returns result tuples with probability above a threshold τ . The joined probabilities of result tuples are computed by using a *combining function* that combines the probabilistic values of joining tuples. For example, in Figure 1, a join of $\langle \text{TID}, \text{Name} \rangle$ in T_1 with $\langle \text{TID}, \text{Speed} \rangle$ in T_2 produces $\langle \text{Named}, \text{Speed} \rangle$ with associated possibilities combining from the possibilities of the joining tuples. The combine function simply uses average. When applying a threshold $\tau=0.8$, only $\{\text{Honda}, 90\}$ is returned.

Since communication cost is the dominating factor due to the network delay and the economic cost associated with transmitting large amounts of data over a network when querying distributed uncertain data [3, 5], the challenge of distributed probabilistic threshold join query is to retrieve result tuples from all the distributed sites with minimum communication cost.

There are a lot of works on network cost efficient distributed join algorithm in relational databases [10]. The *Bloomjoin* [11] algorithm using *Bloom filters* [12] to compress the related join attributes. This approach can significantly reduce the network cost and can be seen as current state of the art [13]. The Bloomjoin algorithm can be applied in the process of handling distributed probabilistic threshold join query directly by treating probability as a common attribute. However, the threshold condition cannot be utilized like join condition during the process, which is the main concern of our approach.

We made the following contributions in this paper. First, we propose a new type of Bloom Filters called *Probability Bloom Filters* (PBF), which can not only answer membership query, but also return its associated probability. Second, based on PBF, we extend Bloomjoin to a network efficient algorithm called *Distributed Probabilistic Threshold Bloomjoin* (DPTBJ) for distributed probabilistic threshold join query. Finally, theoretically evaluation and experiment results show that our proposed method can save network cost efficiently by comparing to original Bloomjoin algorithm in most scenarios.

The rest of this paper is organized as follows: Section 2 introduces preliminaries; PBF and DPTBJ are presented in Section 3 and Section 4; Theoretical analysis and

experiment studies are reported in Section 5; Section 6 reviews the related work; Finally, Section 7 concludes the paper.

2 Preliminaries

Bloom filter. Bloom filter was proposed in [12] as a space-efficient data structure for representing a set S . It allows answer membership queries with a certain probability of false positive that can be made arbitrarily small. A Bloom filter consists of a m bits length array A and a group of k hash functions $\{h_1, h_2, \dots, h_k\}$, these functions are independent and hash element of S to an integer in the range $\{1 \dots m\}$ uniformly. The m bits in A are set to 0 initially, then the insertion of an element e is just set all $A[h_i(e)]$ to 1. Once the set S is represented by a Bloom Filter, a membership query that check x weather in S can be just check weather all $A[h_i(x)]$ are 1. If all the positions are 1, then x is a member of S , otherwise, x is not a member of S . However, there may yield a false positive caused by hash collisions, which means B is actually not in S , but we get a false answer. Denote the number of element in S by n , we have the probability of false positive:

$$P_{fp} = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k \approx \left(1 - e^{-kn/m}\right)^k \quad (1)$$

Bloomjoins. The Bloomjoin algorithm [11] was proposed for efficiently executing joins in distributed database. Consider tables T_1 and T_2 stored in different sites $site_1$ and $site_2$ respectively. For an equal-join $T_1 \bowtie_{\theta} T_2$, applying Bloomjoin algorithm, $site_1$ sends its Bloom filter of the join key ID to remote $site_2$. Then $site_2$ uses the Bloom filter to filter the tuples that do not belong to it. Then, $site_2$ send the remaining tuples to $site_1$, where actual join happens and false positives can be filtered.

Definiton 1. (Probabilistic threshold join). *Given table $\{T_1, T_2, \dots, T_n\}$, the probabilistic attribute in each table is $\{P_1, P_2, \dots, P_n\}$. A probabilistic threshold join on $\{T_1, T_2, \dots, T_n\}$ with join conditions $\{\theta_1, \theta_2, \dots, \theta_{n-1}\}$, combine function $CF()$ and threshold τ is:*

$$\{T_1 \bowtie_{\theta_1} T_2 \bowtie_{\theta_2} \dots \bowtie_{\theta_{n-1}} T_n, CF(), \tau\} = \{T_1 \bowtie_{\theta_1} T_2 \bowtie_{\theta_2} \dots \bowtie_{\theta_{n-1}} T_n \mid CF(P_1, P_2, \dots, P_n) \geq \tau\} \quad (2)$$

In the followings of this paper, we only consider equality join since join with nonequality conditions cannot be handled by using bloom filters.

3 Probability Bloom Filters

In this section, we present Probability Bloom Filters (PBF), which can not only answer membership query like standard Bloom Filters (SBF), but also get the probabilistic value associated with the element. We enable PBF to have this feature by extending SBF. The key idea is similar with Counting Bloom Filters (CBF) [14]. Instead of using a counter at each position as CBF, PBF uses a container which stores the associated probabilistic value for the element. In the followings, we describe the design of PBF and its features.

3.1 Overview of PBF

Now we consider a probabilistic set $S_p = \{x_1, x_2, \dots, x_n\}$, each element x_i in S_p is associated with a probabilistic value p_i which represents the probability of this element. Without loss of generality, we set p_i in $[0, 1]$. PBF represents S_p by using an array A of $m \times c$ bits and a group of independent hash functions $\{h_1, h_2, \dots, h_k\}$ in range $\{1 \dots m\}$. In other words, each entry in PBF is a small container of c bits associated with a basic Bloom filter. We can use PBF to answer membership query and get the associated probabilistic value. For example, to check element B , there are two kinds of answer, one is that B is not a member of S_p , the other is that B is in S_p and returns its associated probabilistic value.

Algorithm 1 Insert (element x , probability p)

1. **Encode** p with c bits data $p^{(c)}$
2. **for** $i = 1$ to k **do**
3. **if** $A[h_i(x)] < p^{(c)}$ **then**
4. $A[h_i(x)] = p^{(c)}$

Fig. 2. Insert operation of PBF

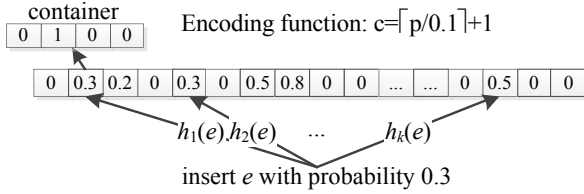


Fig. 3. Example of insert operation in PBF

The insert algorithm of PBF is shown in Fig. 2 and an example in Fig.3 illustrates the insert operation of e with probability 0.3. During insertion of element x , the associated probability p is encoded by a *encode function* to a c bits data $p^{(c)}$. Then use the k hash functions to set $A[h_i(x)]$ to $p^{(c)}$ when $A[h_i(x)] < p^{(c)}$. Remind that A is a $m \times c$ bits array, here we use $A[i]$ to denote the i^{th} container in A . We assume the encode function is monotonic and rounds off to the nearest integers greater than or equal to the original: encoding higher probabilistic value produces a higher result and the result after decoding is always less than original value. Moreover, the encode function never encodes a value to 0, which means we can get an answer that x is in set S_p , and the probabilistic value attained to it is 0. For example, the encoding function in Fig.3 is $c = \lfloor p/0.1 + 1 \rfloor$, and it meets the demands above. We set $A[h_i(x)]$ to $p^{(c)}$ only when it is large than original value, for ensuring that each container of element in PBF is not less than its encoded value $p^{(c)}$. Then we can get the probability by choosing the minimum value in $A[h_i(x)]$ when querying.

After achieving the PBF, we can answer probabilistic membership query based on PBF instead of the whole set. The detailed query process is illustrated in Fig. 4. First, we check every $A[h_i(x)]$ in PBF, if there is any $A[h_i(x)]$ values 0, then return -1, which means that x is not in this set. If all the values are nonzero, then we get the minimal $A[h_i(x)]$ and use decode function to decode $A[h_i(x)]$ to p_{min} , which denotes x is in that set and the associated probability is p_{min} .

According to the insert and query operations, it is obvious that when the value of c is 1, PBF is actually SBF.

Algorithm 2 Query (element x)

1. $tmp = 0$
2. **for** $i = 1$ to k **do**
3. **if** $A[h_i(x)] = 0$ **then**
4. **break**
5. **else**
6. **if** $A[h_i(x)] < tmp$ **or** $tmp=0$
7. $tmp = A[h_i(x)]$
8. **if** $i = k$ **then**
9. **Decode** tmp to p_{min}
10. **return** p_{min}
11. **return** -1

Fig. 4. Query operation in PBF

3.2 Accuracy

Since PBF is a direct extension of SBF, it has the same false positive rate (denotes as P_{fp}) with SPF when answering membership query as formula (1). However, PBF can not only answer whether an item is in a set, but also get the item's associated probability when the item is guaranteed in set by PBF. Despite false positive, we concern with the accuracy of returned probabilistic value.

Theorem 1. *Let an item x in set S_p is associated with probability p , and a query for x on PBF of S_p returns a probability p' . The false positive rate of PBF is P_{fp} . Let the encode function is $E()$, and $E()$ is monotonic and rounding off to ceiling integer. The followings hold:*

$$P_{(E(p')=E(p))} > 1 - P_{fp} \quad (3)$$

$$P_{(E(p')>E(p))} < P_{fp} \quad (4)$$

$$P_{(E(p')<E(p))} = 0 \quad (5)$$

Proof. If PBF returns a wrong value, the k positions of x in PBF must be all rewritten by other insertions. The probability of one position is not rewritten during one insertion is $(1 - 1/m)^k$. So despite the order of insertion, the probability of one position

is rewritten after all n items have been inserted is great or equal than $1 - (1 - 1/m)^{k \times n}$. If all k positions of x are rewritten, the probability is greater or equal than $1 - (1 - (1 - 1/m)^{k \times n})^k$. Since $(1 - (1 - 1/m)^{k \times n})^k$ is the false positive probability of PBF, we get formula (3). In Algorithm 1, since the rewrite operation occurs only when the insertion value is great than the current value, we never get a lower value in PBF than the true value, so formula (4) holds. According to formula (3) and formula (4), formula (5) holds naturally and Theorem 1 is proved to be true.

According to Theorem 1, we can find that the accuracy of returned probabilistic value in PBF is guaranteed by P_{fp} , a lower P_{fp} leads to higher accuracy and query on PBF never return a probabilistic value higher than the true value.

Theorem 2. *Let the encode function is $E()$ which encodes a probability p to a c bits length data $p^{(c)}$, and $E()$ is monotonic and rounding off to ceiling integer. The decode function denotes as $D()$ decodes $p^{(c)}$ to p' . $E()$ is monotonic and rounding off to ceiling integer. The following holds:*

$$p' = D(p^{(c)}) \geq p > D(p^{(c)} - 1) \quad (6)$$

Proof. Since $E()$ is monotonic and rounding off to ceiling integer, it is obvious that $p' \geq p$ and $p > D(p^{(c)} - 1)$.

Through Theorem 2, PBF will never return a probabilistic value less than the true value despite false positives. This feature is very useful for our DPTBJ algorithm in Section 4.

3.3 Composition Operations of PBF

We can use PBF to do set union and intersection like SBF [15, 16], and the result is also a PBF. However, as PBF stores the probability of set item additionally, the compositions of its probability need to be considered.

Suppose there are two probabilistic sets A and B , item x in set A or B have an associated probability $P_A(x)$ or $P_B(x)$, we use two probability bloom filters $PBF(A)$ and $PBF(B)$ as the representation of A and B .

Definiton 2. (Union of probabilistic sets). *The union of probabilistic sets A and B can be represented as:*

$$A \cup B = \{x, \max(p_A(x), p_B(x)) \mid x \in A \text{ or } x \in B\} \quad (7)$$

Definiton 3. (Union of probability bloom filters). *Assume that PBF (A) and PBF (B) use the same $m \times c$ bits length array and hash functions. $A[i]$ or $B[i]$ denotes the i^{th} container of array. The union of PBF(A) and PBF(B), can be represented as:*

$$PBF(A) \cup PBF(B) = \{\max(A[i], B[i]), 1 \leq i \leq m\} \quad (8)$$

Theorem 3. *If $PBF(A \cup B)$, $PBF(A)$ and $PBF(B)$ use the same $m \times c$ bits length array, k hash functions and encode/decode functions, then $PBF(A \cup B) = PBF(A) \cup PBF(B)$.*

Proof. Assume $x \in A$, $y \in B$, we have $A[i] = \text{Encode}(p_A(x))$, $B[i] = \text{Encode}(p_B(y))$ for $1 \leq i \leq m$ in $PBF(A)$ and $PBF(B)$, $p_A(x)$ and $p_B(y)$ are the maximum value when the insert operation comes to position i respectively. According to Definition 2, $PBF(C) = PBF(A) \cup PBF(B)$, $C[i] = \max(A[i], B[i])$. On the other hand, since there are no other items in $A \cup B$ which hash to position i and have a probabilistic value great than $\max(p_A(x), p_B(y))$, for $PBF(D) = PBF(A \cup B)$, we have $D[i] = \max(\text{Encode}(p_A(x)), \text{Encode}(p_B(y))) = \max(A[i], B[i])$. Theorem 3 is proved to be true.

Definiton 4. (Intersection of probabilistic sets). *The intersection of probabilistic sets A and B can be represented as:*

$$A \cap B = \{x, \max(p_A(x), p_B(x)) \mid x \in A \text{ and } x \in B\} \quad (9)$$

Definiton 5. (Intersection of probability bloom filters). *Assume that $PBF(A)$ and $PBF(B)$ use the same $m \times c$ bits length array and hash functions. $A[i]$ or $B[i]$ denotes the i^{th} container of array. The intersection of $PBF(A)$ and $PBF(B)$, denoted as $PBF(C)$ can be represented as:*

$$\begin{aligned} PBF(C) &= PBF(A) \cap PBF(B) = \{C[i], 1 \leq i \leq m\} \\ \text{and } C[i] &= \begin{cases} \max(A[i], B[i]) & , \quad A[i] > 0, B[i] > 0 \\ 0 & , \quad \text{others} \end{cases} \end{aligned} \quad (10)$$

Theorem 4. *If $PBF(A \cap B)$, $PBF(A)$ and $PBF(B)$ use the same $m \times c$ bits length array, k hash functions and encode/decode functions, then $PBF(A \cap B) = PBF(A) \cap PBF(B)$ with the probability $(1 - 1/m)^{k^2 \times (|A| \times |B| - |A \cap B|^2)}$.*

Proof. According to Definition 3, Definition 5 and Theorem 3, we have:

$$\begin{aligned} PBF(A) \cap PBF(B) &= (PBF(A - A \cap B) \cap PBF(B - A \cap B)) \cup \\ &\quad (PBF(A - A \cap B) \cap PBF(A \cap B)) \cup \\ &\quad (PBF(B - A \cap B) \cap PBF(A \cap B)) \cup PBF(A \cap B) \end{aligned} \quad (11)$$

According to (11), it is easy to find that $PBF(A \cap B) = PBF(A) \cap PBF(B)$ only if $(PBF(A - A \cap B) \cap PBF(B - A \cap B)) \cup (PBF(A - A \cap B) \cap PBF(A \cap B)) \cup (PBF(B - A \cap B) \cap PBF(A \cap B)) = 0$.

For any item $x \in (A - A \cap B)$, the probability of which $A[hash_1(x)], \dots, A[hash_k(x)]$ of $PBF(B - A \cap B)$ are 0 should be $(1 - 1/m)^{k^2 \times |A - A \cap B|}$, then the probability that $PBF(A - A \cap B) \cap PBF(B - A \cap B)$ is 0 should be $(1 - 1/m)^{k^2 \times |A - A \cap B| \times |B - A \cap B|}$. Similarly, we can infer the probability that $PBF(A \cap B) = PBF(A) \cap PBF(B)$ is:

$$\begin{aligned}
 P &= (1 - 1/m)^{k^2 \times (|A - A \cap B| \times |B - A \cap B| + |A - A \cap B| \times |A \cap B| + |B - A \cap B| \times |A \cap B|)} \\
 &= (1 - 1/m)^{k^2 \times (|A - A \cap B| \times |B| + |B - A \cap B| \times |A \cap B|)} \\
 &= (1 - 1/m)^{k^2 \times (|A| \times |B| - |A \cap B|^2)}
 \end{aligned} \tag{12}$$

Theorem 4 is proved to be true.

4 Distributed Probabilistic Threshold Bloomjoin Algorithm

In this section, we propose *distributed probabilistic threshold bloomjoin algorithm* (DPTBJ) for distributed probabilistic threshold join query. DPTBJ extends Bloomjoin algorithm [11] by using PBF proposed in Section 3 to represent data which need to be transmitted in distributed join operation. Our algorithm can save the network cost by reducing the data size that need to be transmitted between different sites. We mainly consider two sites distributed join since multi-sites join can be handled by extending DPTBJ naturally, and we will discuss some properties of DPTBJ in 4.2.

4.1 Algorithm

Suppose we have a distributed database consist of two sites $Site_1$ and $Site_2$, each site has a relation table to be joined denotes as A and B . Each tuple in the table have a probabilistic value p denotes the confidence or probability of the tuple. We probabilistic threshold join query as $\{A \bowtie_{\theta} B, CF(), \tau\}$, which means return the tuples joined under condition θ in table A and B , and the probabilistic value combined by a user defined combine function $CF()$ of the joined tuples is above or equal than τ .

The DPTBJ algorithm is shown in Fig.5. First, we construct $PBF_{A(\theta)}$ on the attributes relevant to the join condition θ of A . Note that the probabilistic values of tuples in A are encoded in $PBF_{A(\theta)}$. Then we send $PBF_{A(\theta)}$ to $Site_2$, and scan B for tuples which match the join condition θ by querying $PBF_{A(\theta)}$. A tuple that matches any elements in $PBF_{A(\theta)}$ will return a nonnegative probabilistic value, and if the probabilistic value combined by $CF()$ is above or equal than threshold τ , this tuple of B is added to *Resultlist*. Tuples in candidate set which are irrelevant to the final result can be pruned by this process. Then the *Resultlist* is send back to A to compute a local probabilistic threshold join at $Site_1$.

The join process at $Site_1$ can utilize any join algorithms that is proper to a certain situation. For example, we can use the nested-loop index join algorithm if there are indices on the join condition attributes in A .

<p>Algorithm 3 DPTBJ ($R_1, R_2, \theta, \tau, CF()$)</p> <ol style="list-style-type: none"> 1. Generate $PBF_{A(\theta)}$ for A in $Site_1$ 2. Send $PBF_{A(\theta)}$ to $Site_2$ 3. Foreach $tuple$ in B 4. $p = \text{Query}(tuple_{(\theta)})$ 5. If $p \geq 0$ then 6. If $CF(p, tuple.p) \geq \tau$ then 7. Add $tuple$ to $Resultlist$ 8. Send $Resultlist$ to $Site_1$ 9. Excute $\{A \bowtie_{\theta} Resultlist, CF(), \tau\}$ at $Site_1$ 11. Return final result

Fig. 5. DPTBJ algorithm

4.2 Discussion

In this section, we cover a few of properties of our algorithm that not discuss in the initial form.

1) Combine function

According to Theorem 2, we know that the probabilistic value returned by querying PBF is great or equal than the true value. To avoid false negative, the combine function in probability combination should be monotonic. However, it is reasonable to have monotonic combine function in probabilistic threshold join since combining higher confidence probability usually produces higher resulting.

2) The optimal PBF in DPTBJ

Ramesh *et al.* [13] proposed optimal Bloom filters for Bloomjoin. The same method can be applied to optimal PBF in DPTBJ. However, we should consider the size of container additionally in DPTBJ. Typically, c is smaller than the size of which stores the original value. A higher c means higher precision when query PBF and could lead to prune more candidates need to be transmitted, meanwhile, it also lead a larger size of PBF. To minimize the network cost, we should choose the value of c properly. The choice of c will be discussed in Section 5.

3) Multi-sites join

The Bloomjoin algorithm can extend to more than 2 sites to handle equal join due to the composition operations of Bloom Filters [15, 16]. The composition operations include set union and set intersection. Since PBF have similar composition operations which is mentioned in Section 3, we can extend DPTBJ algorithm by using the composition operations of PBF to handle probabilistic threshold join query for any number of sites.

5 Evaluations

Since network cost is the dominating cost factor in distributed join execution [13], the main concern of our algorithm is network cost. In this section, we demonstrate the effectiveness of DPTBJ on network cost. Specifically, we compared DPTBJ with Bloomjoin [11] through theoretical and experimental analysis. Note that although Bloomjoin is not design for distributed probabilistic threshold join, it can handle it naturally without any modification. Actually, DPTBJ algorithm degenerates to Bloomjoin when the size of container c is 1.

5.1 Theoretical Analysis

1) Network cost

Suppose $Site_1$ holds table A and $Site_2$ holds table B , consider a probabilistic threshold join query $\{A \bowtie_{\theta} B, CF(), \tau\}$ using DPTBJ, the total network cost consists of the cost of sending $PBF_{A(\theta)}$ from $Site_1$ to $Site_2$, and sending *Resultlist* from $Site_2$ to $Site_1$. Let $m \times c$ denote the length of $PBF_{A(\theta)}$ in bits, the size of *Resultlist* is r , and record length of B is l . The network cost of DPTBJ is:

$$C_{DPTBJ} = m \times c + r \times l \quad (13)$$

Resultlist contains the true tuples of the probabilistic threshold join and the false positives caused by $PBF_{A(\theta)}$. Let α denote the join selectivity, β_D denote the selectivity by using threshold condition at $Site_2$, and total record size of B is n_2 . The size of *Resultlist* can be represented as:

$$r = (n_2 \times \alpha + (n_2 - n_2 \times \alpha) \times P_{fp}) \times \beta_D \quad (14)$$

Then we have:

$$C_{DPTBJ} = m \times c + (n_2 \times \alpha + (n_2 - n_2 \times \alpha) \times P_{fp}) \times l \times \beta_D \quad (15)$$

Similarly, we can get the network cost of Bloomjoin. Note that Bloomjoin can also utilize the threshold condition to prune candidate set at $Site_2$ since the combine function is monotonic and we can assume the probability of each item in SBF is 1. So there is also a threshold selectivity β_B for Bloomjoin, and β_B is great than β_D .

$$C_{Bloomjoin} = m + (n_2 \times \alpha + (n_2 - n_2 \times \alpha) \times P_{fp}) \times l \times \beta_B \quad (16)$$

Denote $(n_2 \times \alpha + (n_2 - n_2 \times \alpha) \times P_{fp}) \times l$ as L . As c is great or equal than 2, to gain network cost reduction against Bloomjoin, we should have:

$$2 \leq c < L \times \beta_B \times (1 - \beta_D) / m + 1 \quad (17)$$

Then

$$L \times \beta_B \times (1 - \beta_D) > m \quad (18)$$

It is reasonable that β_B is relatively high because the lack of probabilities of probe tuples. Assume the false positive rate is very lower by properly choosing m and k , since $n_2 \times l$ is much higher than m , the dominant factors of gaining network reduction are join selectivity α and threshold selectivity β_D . According to inequality (17), we know that DPTBJ can have a better performance with a proper choice of c except α is very low or β_D is very high, which means DPTBJ can outperform Bloomjoin in most scenarios.

2) The optimal c

Now we consider the optimal c to minimize C_{DPTBJ} . As the distribution of probabilistic value of A can be known in prior, we can make the values stored in the containers of $PBF_{A(\theta)}$ to have a uniform distribution by choosing a proper *encode function*. According to Theorem 2, for the threshold selectivity β_D at $Site_2$ by using DPTBJ and the real threshold selectivity β , we have:

$$0 \leq \beta_D - \beta < 2^{-c} \quad (19)$$

Then, equation (15) can be rewritten as:

$$m \times c + L \times \beta \leq C_{DPTBJ} < m \times c + L \times (\beta + 2^{-c}) \quad (20)$$

To minimize the upper bound of C_{DPTBJ} , we differentiate the upper bound with respect to c . Then we get the optimal c :

$$c = \log_2^{(L \times \ln 2 / m)} \quad (21)$$

As L and m can be estimated by query optimizer of modern database, we can choose the optimal c in advance.

5.2 Experiment

We generated synthetic data to evaluate the network cost of DPBTJ against Bloomjoin. The experiment was conducted on vertically fragmented databases. We synthetically generated two relation tables R and S in different sites, each table has 10^5 tuples with a primary key and a probabilistic value denotes its confidence. Regard R as probe table and S as build table, the length of S can vary by adding other attributes. The probability distribution is uniformly from 0 to 1 and the combine function is multiplication. We set $k=5$ and $m=8 \times 10^5$ where k is the number of hash functions and m is the length of the filter using both by DPTBJ and Bloomjoin.

Fig. 6 illustrates how network cost varies with size of container under different join selectivity and threshold selectivity. Note that in this figure, it means network cost of Bloomjoin when the size of container is 1. It is clear that DPTBJ can outperform Bloomjoin by properly choosing c except join selectivity is very low or threshold selectivity is very high, that is because we can't meet the demand of inequality (18) when having very low join selectivity and very high threshold selectivity.

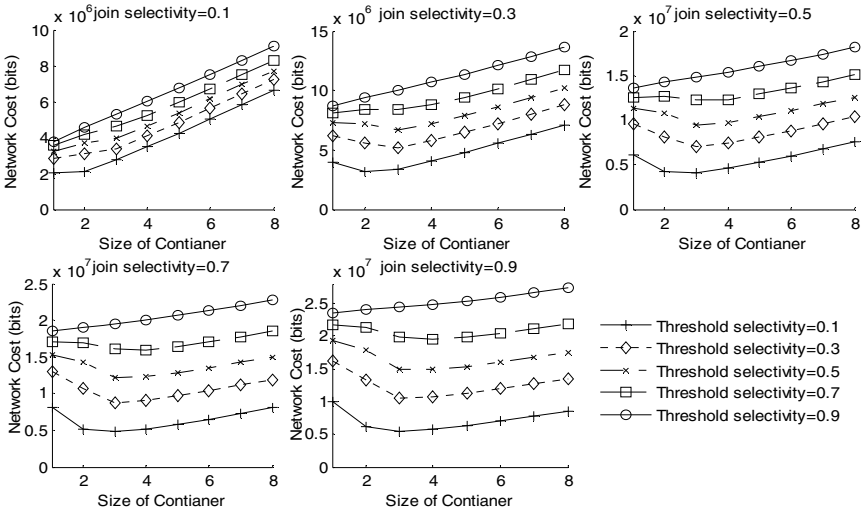


Fig. 6. Original Bloomjoin Vs DPTBJ

6 Related Work

Considerable research effort has been put on queries in distributed environment. A broad summary of query processing techniques for structured data at distributed sites is provided in [10]. Techniques based on bloom filters for optimizing join queries in distributed databases have been proposed in [13, 15].

An extensive survey of problems in uncertain data management is provided in [17]. Recently, there has been some work addressing join queries in uncertain databases. The work in [18] addresses the problem of joins over uncertain attributes described as probability distribution functions, while we consider discrete probability. A recent work [9] addresses confidence aware joins, which is similar to ours, but they assume a centralized environment.

Until recently, there are some works on querying distributed uncertain data. Li *et al.* [3] study probabilistic top-k query with “expected rank” semantic in distributed systems. Ye *et al.* [4] propose a general approach for efficient processing of any probabilistic top-k query in distributed wireless sensor networks. Ding *et al.* [5] consider distributed skyline queries over uncertain data. To the best of our knowledge, this work is the first one to address probabilistic threshold join query over distributed uncertain data.

7 Conclusions

In this paper, we considered probabilistic threshold join query in a distributed environment. To represent probabilistic set, we proposed Probabilistic Bloom Filters extending from standard bloom filters which can not only answer membership query like standard bloom filters, but also get the probability associated with the element.

We demonstrated the accuracy of the answered probabilistic value which is guaranteed by PBF's false positive rate. Then we proposed DPTBJ algorithm extending from Bloomjoin to handle this query. Theoretical and experimental analysis on comparing DPTBJ with Bloomjoin showed that DPTBJ can have a better performance in most scenarios except join selectivity is very low or threshold selectivity is very high. Our future work includes more general probabilistic query in distributed settings and other uncertain data models.

References

1. Deshpande, A., Guestrin, C., Madden, S., Hellerstein, J., Hong, W.: Model-driven data acquisition in sensor networks. In: VLDB (2004)
2. Deng, K., Zhou, X., Shen, H.T.: Multi-source skyline query processing in road networks. In: ICDE (2007)
3. Li, F., Yi, K., Jests, J.: Ranking Distributed Probabilistic Data. In: SIGMOD (2009)
4. Ye, M., Liu, X., Lee, W., Lee, D.: Probabilistic Top-k Query Processing in Distributed Sensor Networks. In: ICDE (2010)
5. Ding, X., Jin, H.: Efficient and Progressive Algorithms for Distributed Skyline Queries over Uncertain Data. In: ICDCS (2010)
6. Fuhr, N., Rölleke, T.: A probabilistic relational algebra for the integration of information retrieval and database systems. ACM TOIS 14(1) (1997)
7. Perez, L., Arumugam, S., Jermaine, C.: Evaluation of Probabilistic Threshold Queries in MCDB. In: SIGMOD (2010)
8. Yang, S., Zhang, W., Zhang, Y., Lin, X.: Probabilistic Threshold Range Aggregate Query Processing over Uncertain Data. In: Li, Q., Feng, L., Pei, J., Wang, S.X., Zhou, X., Zhu, Q.-M. (eds.) APWeb/WAIM 2009. LNCS, vol. 5446, pp. 51–62. Springer, Heidelberg (2009)
9. Agrawal, P., Widom, J.: Confidence-aware join algorithms. In: ICDE (2009)
10. Kossmann, D.: The state of the art in distributed query processing. ACM Comput. Surv. 32(4), 422–469 (2000)
11. Mackert, L.F., Lohman, G.M.: R* optimizer validation and performance evaluation for distributed queries. In: VLDB (1986)
12. Bloom, B.: Space/time tradeoffs in hash coding with allowable errors. Commun. ACM 13(7), 422–426 (1970)
13. Ramesh, S., Papapetrou, O., Siberski, W.: Optimizing distributed joins with bloom filters. In: Parashar, M., Aggarwal, S.K. (eds.) ICDCIT 2008. LNCS, vol. 5375, pp. 145–156. Springer, Heidelberg (2008)
14. Fan, L., Cao, P., Almeida, J., Broder, A.Z.: Summary cache: a scalable wide-area web cache sharing protocol. SIGCOMM Comput. Commun. Rev. 28(4), 254–265 (1998)
15. Michael, L., Nejd, W., Papapetrou, O., Siberski, W.: Improving distributed join efficiency with extended bloom filter operations. In: AINA (2007)
16. Papapetrou, O., Siberski, W., Nejd, W.: Cardinality estimation and dynamic length adaptation for Bloom filters. Distrib Parallel Databases 28, 119–156 (2010)
17. Aggarwal, C.C., Yu, P.S.: A survey of uncertain data algorithms and applications. IEEE Trans. Knowl. Data Eng. 21(5), 609–623 (2009)
18. Cheng, R., Singh, S., Prabhakar, S., Shah, R., Vitter, J.S., Xia, Y.: Efficient join processing over uncertain data. In: CIKM (2006)

Bayesian Classifiers for Positive Unlabeled Learning*

Jiazhen He¹, Yang Zhang^{1,2,**}, Xue Li³, and Yong Wang⁴

¹ College of Information Engineering, Northwest A&F University, China

² State Key Laboratory for Novel Software Technology, Nanjing University, China

³ School of Information Technology and Electrical Engineering, The University of Queensland, Australia

⁴ School of Computer, Northwestern Polytechnical University, China

Abstract. This paper studies the problem of Positive Unlabeled learning (PU learning), where positive and unlabeled examples are used for training. Naive Bayes (NB) and Tree Augmented Naive Bayes (TAN) have been extended to PU learning algorithms (PNB and PTAN). However, they require user-specified parameter, which is difficult for the user to provide in practice. We estimate this parameter following [2] by taking the “selected completely at random” assumption and reformulate these two algorithms with this assumption. Furthermore, based on supervised algorithms Averaged One-Dependence Estimators (AODE), Hidden Naive Bayes (HNB) and Full Bayesian network Classifier (FBC), we extend these algorithms to PU learning algorithms (PAODE, PHNB and PFBC respectively). Experimental results on 20 UCI datasets show that the performance of the Bayesian algorithms for PU learning are comparable to corresponding supervised ones in most cases. Additionally, PNB and PFBC are more robust against unlabeled data, and PFBC generally performs the best.

Keywords: Positive Unlabeled Learning, Bayesian Classifiers, Naive Bayes, Hidden Naive Bayes, Full Bayesian Network Classifiers.

1 Introduction

In supervised classification analysis, examples from all the predefined classes are required to produce a classifier. In particular, for binary classification, a classifier is built using both positive and negative examples. In many real-life classification tasks, however, it is often hard and expensive to get labeled examples from all the classes, while examples of one specific class and unlabeled examples are relatively easy to obtain. This has motivated the problem of PU learning. In PU learning framework, we are interested in one specific class of data, named as positive class, and a classifier is learned from only labeled positive examples and unlabeled examples. The problem of PU learning is prevalent in real-world applications. For example, to discover future customers for direct marketing, the current customers of the company can be used as positive examples while unlabeled examples can be purchased at a low cost compared with obtaining negative examples[1].

* This work is supported by the National Natural Science Foundation of China (60873196) and Chinese Universities Scientific Fund (QN2009092).

** Corresponding author.

Bayesian classifiers, as statistical classifiers, which can predict class membership probabilities, have been widely used in classification tasks. Naive Bayes is a simple Bayesian classifier, which assumes that all the attributes are independent of each other given the class label. However, the assumption may not often hold in practice, and hence the performance can be harmed. TAN[3] relaxes the independence assumption and allows at most one parent expect class variable for each attribute. These two Bayesian classifiers have been extended to PU learning scenario. However, they require the user to provide the prior probability of positive class (p), which is often difficult in practice. In this paper, we utilize the method proposed in [2] to estimate p so as to free the user from providing it. Furthermore, we extend supervised Bayesian algorithms AODE[4], HNB[5] and FBC[6] to PU learning scenario. The comparison results between Bayesian algorithms for PU learning and their corresponding supervised Bayesian algorithms, which use both positive and negative examples for training show that the proposed Bayesian algorithms for PU learning perform comparably to supervised ones in most cases. In addition, PNB and PFBC are relatively robust against unlabeled data, and PFBC generally performs the best, especially in terms of F1.

This paper is organized as follows. In the next section, we discuss related work. Section 3 illustrates the proposed algorithms in detail. The experimental results are shown in Section 4. And finally, Section 5 concludes this paper.

2 Related Work

Numerous studies on PU learning have been conducted in recent years. In [7], Schölkopf *et al.* learn one-class SVM from only labeled positive examples ignoring the unlabeled examples. However, this method cannot exploit the useful information in the unlabeled examples, which may make contribution to classification. It can be beneficial to build a classifier using both positive and unlabeled examples. Currently, most of the PU learning algorithms are devised for text classification, and there are two main approaches. One classical approach is to take the two-step strategy, which firstly identifies a set of reliable negative examples from the unlabeled set, and then constructs a classifier on the positive and reliable negative examples iteratively[8,9,10]. Another approach is to assign weights to the examples, and then, supervised learning algorithms are used for learning[2,11,12]. PU learning problems can also be dealt with by estimating statistical information from positive and unlabeled examples, such as PNB[13] which is devised for text classification and POSC4.5[14] which is devised for general classification tasks. The problem of PU learning is also studied under other scenarios, such as data stream classification[15,16] and uncertain data classification[17].

Most of PU learning algorithms focus on text classification[8,9,10,11,12,13]. In this paper, we focus on Bayesian algorithms for PU learning to handle general attributes. In [18], Calvo *et al.* generalized PNB[13], which is only applicable to text classification tasks where text documents are represented in bag-of-words format, to handle general discrete attributes. Furthermore, they extended it to Positive Tree Augmented Naive Bayes (PTAN), a more complex Bayesian classifier. However, the algorithms they proposed require the user to provide p as a parameter. Thus, the averaged version of these classifiers (APNB and APTAN) was introduced by stimulating p by means of

Beta distribution. Nevertheless, the parameters of the Beta distribution still need to be provided by the user, which is often difficult in practice. In [2], Elkan *et al.* assumed that labeled positive examples are selected completely randomly from all positive examples. And under this assumption, they proposed a method to estimate p from positive and unlabeled examples. Therefore, in this paper, we also make this assumption and utilize the method in [2] to estimate p so as to free the user from providing it. Furthermore, we extend supervised Bayesian algorithms AODE, HNB and FBC to PU learning algorithms.

3 Bayesian Classifiers for Positive Unlabeled Learning

In this section, firstly, we give the problem description. Then, we present different Bayesian classifiers for PU learning. Because we utilize the method proposed in [2] to estimate p under the “selected completely at random” assumption, we reformulate the estimation of some parameters in PNB and PTAN, and then, we extend supervised Bayesian algorithms AODE, HNB, FBC to PU learning algorithms.

3.1 Problem Description

Let D denote the training set, which consists of a set of positive examples P and a set of unlabeled examples U , $D = P \cup U$. Examples can be represented by $\langle \mathbf{X}, C, S \rangle$, where $\mathbf{X} = \langle X_1, \dots, X_n \rangle$ represents the attribute vector with n attributes; C represents the class variable which can take values from $\{0, 1\}$, 0 for negative and 1 for positive; S represents the label variable which can take values from $\{0, 1\}$, 0 for unlabeled, 1 for labeled. Examples in P are labeled as positive, represented by $\langle \mathbf{X}, C = 1, S = 1 \rangle$, while examples in U are unlabeled, denoted as $\langle \mathbf{X}, C = ?, S = 0 \rangle$. We write $V(X_i)$ for the domain from which attribute $X_i (1 \leq i \leq n)$ may take values.

We follow the model used in [12][11][2]. In this model, an assumption is made that the labeled positive examples are selected completely randomly from all positive examples. This means that positive examples are randomly labeled positive with probability $1-a$, and are left unlabeled with probability a . This assumption was formalized and characterized as “selected completely at random” assumption in [2]. This assumption is important because it allows p to be estimated from positive and unlabeled examples.

Based on the above formalization, we will build Bayesian classifiers from positive and unlabeled examples.

3.2 Positive Naive Bayes (PNB)

Based on Bayes theorem and conditional independence assumption, the posterior probability that an example \mathbf{x} is classified to class $c (c \in \{0, 1\})$ is formulated as

$$P(C = c | \mathbf{X} = \mathbf{x}) \propto P(C = c) \prod_{i=1}^n P(X_i = x_{ij} | C = c) \quad (1)$$

We need to estimate $P(C = c)$, $P(X_i = x_{ij} | C = 1)$ and $P(X_i = x_{ij} | C = 0)$ (from now on p , $P(x_{ij}|1)$, $P(x_{ij}|0)$ respectively), where $x_{ij} \in V(X_i) (i = 1, \dots, n, j =$

$1, \dots, |V(X_i)|$). In PU learning scenario, the key problem is the estimation of $P(x_{ij}|0)$ because there are no negative examples available. However, considering [18]

$$P(x_{ij}) = P(x_{ij}|1)p + P(x_{ij}|0)(1 - p) \quad (2)$$

$P(x_{ij}|0)$ can be estimated by estimating $P(x_{ij}|1)$ and $P(x_{ij})$. In [13][18], an underlying generative model is assumed. In this model, a class is selected according to class prior probabilities. They assume that U is generated according to the underlying model. Therefore, $P(x_{ij})$ is estimated on U and p is viewed as the proportion of positive class in U . However, under the ‘‘selected completely at random’’ assumption, $P(x_{ij})$ is estimated on D and p is viewed as the proportion of positive class in D . Therefore, the estimation of $P(x_{ij}|0)$ is different from [18]. Here, $P(x_{ij}|0)$ can be estimated by

$$P(x_{ij}|0) = \frac{N(x_{ij}, U) - P(x_{ij}|1)(|D|p - |P|)}{|D|(1 - p)} \quad (3)$$

where $N(x_{ij}, U)$ denotes the number of unlabeled examples where $X_i = x_{ij}$, $|D|$ denotes the cardinality of the training set, and $|P|$ denotes the cardinality of the set of positive examples. $P(x_{ij}|1)$ can be estimated on P by means of maximum likelihood estimator with Laplace correction. In order to avoid $P(x_{ij}|0)$ to be negative, it can be handled following the method in [18].

Note that p is estimated according to the method in [2], instead of being provided by the user. In [2], a classifier is built from positive and unlabeled examples using standard classification algorithm. Further detail can be found in [2]. In this paper, we use naive Bayes to obtain the classifier and calibrate it using isotonic regression [19] so as to estimate p . In the following subsections, p is estimated by this method.

3.3 Positive Tree Augmented Naive Bayes (PTAN)

TAN allows at most one parent expect class variable for each attribute [3]. The key point of the structure learning of TAN is computing the conditional mutual information between two attributes X_i and X_k given class variable C , which is defined as

$$\begin{aligned} I(X_i, X_k|C) &= \sum_{j=1}^{|V(X_i)|} \sum_{l=1}^{|V(X_k)|} P(x_{ij}, x_{kl}, 1) \log \frac{P(x_{ij}, x_{kl}|1)}{P(x_{ij}|1)P(x_{kl}|1)} \\ &+ \sum_{j=1}^{|V(X_i)|} \sum_{l=1}^{|V(X_k)|} P(x_{ij}, x_{kl}, 0) \log \frac{P(x_{ij}, x_{kl}|0)}{P(x_{ij}|0)P(x_{kl}|0)} \end{aligned} \quad (4)$$

Here, the different estimations under the ‘‘selected completely at random’’ assumption compared with [18] is $P(x_{ij}|0)$, $P(x_{kl}|0)$, $P(x_{ij}, x_{kl}|0)$ and $P(x_{ij}, x_{kl}, 0)$. $P(x_{ij}|0)$ and $P(x_{kl}|0)$ can be estimated by the estimations in Section 3.2 and $P(x_{ij}, x_{kl}|0)$ can be estimated similarly. For $P(x_{ij}, x_{kl}, 0)$, we estimate it as

$$P(x_{ij}, x_{kl}, 0) = P(x_{ij}, x_{kl}|0)(1 - p) \quad (5)$$

In the parameter learning, we need to estimate $P(x_{ij}|Pa(X_i) = pa_i)$, where $Pa(X_i)$ is the parent set of X_i . In particular, we need to estimate $P(x_{ij}|1, pa_i^*)$ and $P(x_{ij}|0, pa_i^*)$

as illustrated in [18]. The different estimation here is $P(x_{ij}|0, pa_i^*)$, which is estimated by

$$P(x_i|0, pa_i^*) = \frac{P(x_i, pa_i^*|0)}{P(pa_i^*|0)} \quad (6)$$

$P(pa_i^*|0)$ and $P(x_i, pa_i^*|0)$ can be estimated by the estimations in Section 3.2

3.4 Positive Averaged One-Dependence Estimators (PAODE)

In AODE [4], structure learning is not required. Instead, it builds n classifiers by letting each attribute be the parent of all other attributes, and the prediction is the averaged predictions of n classifiers. Actually, it is an ensemble learning method. For an example \mathbf{x} , the class of \mathbf{x} is

$$f(\mathbf{X} = \mathbf{x}) = \arg \max_c \sum_{i=1}^n P(C = c, X_i = x_{ij}) \prod_{k=1}^n P(X_k = x_{kl} | C = c, X_i = x_{ij}) \quad (7)$$

We need to estimate $P(x_{ij}, 1)$, $P(x_{ij}, 0)$, $P(x_{kl}|1, x_{ij})$ and $P(x_{kl}|0, x_{ij})$. In PU learning scenario, $P(x_{ij}, 1)$ and $P(x_{ij}, 0)$ can be estimated by

$$P(x_{ij}, 1) = P(x_{ij}|1)p \quad (8)$$

$$P(x_{ij}, 0) = P(x_{ij}|0)(1 - p) \quad (9)$$

$P(x_{kl}|1, x_{ij})$ can be estimated on P by means of maximum likelihood estimator with Laplace correction, while $P(x_{kl}|0, x_{ij})$ can be estimated by

$$P(x_{kl}|0, x_{ij}) = \frac{P(x_{kl}, x_{ij}|0)}{P(x_{ij}|0)} \quad (10)$$

Based on the above estimations, we can learn AODE in PU learning scenario. This algorithm is named as Positive Averaged One-Dependence Estimators (PAODE).

3.5 Positive Hidden Naive Bayes (PHNB)

HNB considers the influences from all attributes [5]. Each attribute has a hidden parent, which combines the influences from all other attributes. For an example \mathbf{x} , the class of \mathbf{x} is

$$f(\mathbf{X} = \mathbf{x}) = \arg \max_c P(C = c) \prod_{i=1}^n P(X_i = x_{ij} | X_{hp_i} = x_{hp_i}, C = c) \quad (11)$$

where $P(X_i = x_{ij} | X_{hp_i} = x_{hp_i}, C = c) = \sum_{k=1, k \neq i}^n W_{ik} * P(X_i = x_{ij} | X_k = x_{kl}, C)$.

In [5], Jiang *et al.* use the conditional mutual information between two attributes X_i and X_k as the weight of $P(X_i = x_{ij} | X_k = x_{kl}, C)$, and W_{ik} is defined as

$$W_{ik} = \frac{I(X_i, X_k | C)}{\sum_{k=1, k \neq i}^n I(X_i, X_k | C)} \quad (12)$$

Therefore, the key problem of learning HNB is the estimation of $P(X_i = x_{ij}|X_k = x_{kl}, C)$ and $I(X_i, X_k|C)$. In PU learning scenario, we need to estimate $P(x_{ij}|x_{kl}, 1)$, $P(x_{ij}|x_{kl}, 0)$ and $I(X_i, X_k|C)$. The estimation of $I(X_i, X_k|C)$ can be found in PTAN in Section 3.3 and the estimation of $P(x_{ij}|x_{kl}, 1)$ and $P(x_{ij}|x_{kl}, 0)$ can be found in PAODE in Section 3.4. Note that in [5], Jiang *et al.* use the M-estimation to estimate above probabilities, so we also use it to estimate the probabilities.

Based on the estimations, we can learn a HNB from positive and unlabeled examples. This algorithm is named as Positive Hidden Naive Bayes (PHNB).

3.6 Positive Full Bayesian Network Classifier (PFBC)

In FBC[6], the structure is a full Bayesian network (BN) while the conditional probability table (CPT) for each attribute is represented in a decision tree, called CPT-tree. A CPT-tree is built for each attribute after building a full BN. The key point of learning a full BN is learning an order of attributes. Su *et al.* rank the attributes according to the total influence of each attribute on all other attributes. The influence (dependency) between two attributes is evaluated by mutual information $I(X_i, X_k)$. In order to avoid unreliable dependence, Su *et al.* judge the reliability of dependence using a threshold $\varphi(X_i, X_k)$. Then, the total influence of an attribute X_i on all other attributes is defined as follows [6]

$$W(X_i) = \sum_{k(k \neq i)}^{I(X_i, X_k) > \varphi(X_i, X_k)} I(X_i, X_k) \tag{13}$$

Therefore, the key problem of structure learning is the computation of $I(X_i, X_k)$ and $\varphi(X_i, X_k)$. In supervised learning framework, the training set is divided into c subsets according to the possible values of c , and $I(X_i, X_k)$ is computed on each subsets. In other words, $I(X_i, X_k)$ is actually the conditional mutual information $I(X_i, X_k|c)$,

$$I(X_i, X_k|c) = \sum_{j=1}^{|V(X_i)|} \sum_{l=1}^{|V(X_k)|} P(x_{ij}, x_{kl}|c) \log \frac{P(x_{ij}, x_{kl}|c)}{P(x_{ij}|c)P(x_{kl}|c)}$$

In PU learning scenario, we need to estimate $I(X_i, X_k|1)$ and $I(X_i, X_k|0)$, which can be estimated taking into account estimations in previous subsections. Additionally, $\varphi^P(X_i, X_k)$ and $\varphi^N(X_i, X_k)$, which judge the reliability of the dependence of two attributes $I(X_i, X_k|1)$ and $I(X_i, X_k|0)$, can be estimated by

$$\varphi^P(X_i, X_k) = \frac{\log |P|}{2|P|} \times T_{ik} \tag{14}$$

$$\varphi^N(X_i, X_k) = \frac{\log |D * (1 - p)|}{2|D * (1 - p)|} \times T_{ik} \tag{15}$$

where $T_{ik} = |X_i| \times |X_k|$ [6]. Based on above estimations, we can learn full BN on positive and unlabeled examples using the structure learning algorithm for FBC in [6].

Once we have the structure of a full BN, we need to learn the CPT-tree for each attribute in each subset. In PU learning scenario, in the subset of positive examples,

we can learn CPT-tree using the CPT-tree learning algorithm in [6]. However, we cannot use it to learn CPT-trees for attributes in the subset of negative examples directly due to lacking of negative examples. But we can learn CPT-tree with the help of positive and unlabeled examples. In PU learning scenario, the key problem of CPT-tree learning algorithm [6] in the negative examples is the computation of the local mutual information $I^{D_N}(X_i, X_k)$ and $\varphi^{D_N}(X_i, X_k)$ on current set of negative examples D_N . In addition, in order to judge if D_N is empty and the examples in D_N are all of the same value of X_i , we need to estimate $|D_N|$ which is the cardinality of the current set of negative examples, and $N(X_{ij}, D_N)$ which is the number of current negative examples where $X_i = x_{ij}$. Based on the current set of positive examples D_P and the current set of unlabeled examples D_U , we can estimate $|D_N|$ as

$$|D_N| = |D_U| - \frac{|D_P|}{|P|} \times (|D|p - |P|) \quad (16)$$

And $N(X_{ij}, D_N)$ can be estimated by

$$N(X_{ij}, D_N) = N(X_{ij}, D_U) - \frac{N(X_{ij}, D_P)}{|D_P|} \times (|D| \frac{|D_P|}{|P|} p - |D_P|) \quad (17)$$

Note that estimating $I^{D_N}(X_i, X_k)$ actually means estimating $P^{D_N}(x_{ij}|0)$, $P^{D_N}(x_{kl}|0)$ and $P^{D_N}(x_{ij}, x_{kl}|0)$. $P^{D_N}(x_{ij}|0)$ can be estimated by

$$P^{D_N}(x_{ij}|0) = \frac{N(X_{ij}, D_N)}{|D_N|} \quad (18)$$

$P^{D_N}(x_{ij}|0)$ can be handled following the method in [18] so as to avoid negative value. $P^{D_N}(x_{kl}|0)$ and $P^{D_N}(x_{ij}, x_{kl}|0)$ can be estimated in a similar way. On the other hand, $\varphi^{D_N}(X_i, X_k)$ can be estimated by

$$\varphi^{D_N}(X_i, X_k) = \frac{\log |D_N|}{2|D_N|} \times T_{ik} \quad (19)$$

Here, the main difference compared with CPT-tree learning algorithm [6] is that because of the absence of negative examples, we need to store the statistic information of both positive and unlabeled examples in learning CPT-tree, so as to estimate the statistic information for negative examples. Therefore, both D_P and D_U are needed to be partitioned into different subsets in the process of learning CPT-tree. This algorithm is named as Positive Full Bayesian Network Classifier (PFBC).

4 Experiment

In this section, we present the experimental results of our algorithms. 20 real-world benchmark datasets from UCI repository [20] are used in the experiments. Information about these datasets is shown in Table 1.

In Table 1, Pos class represents the class label used as positive class, and Pos/Neg represents the percentage of the number of positive examples against that of negative

Table 1. Experimental datasets

Dataset	Size	#Attribute	#Class	Pos Class	Pos/Neg
adult	48842	14	2	<=50K	76.1%/23.9%
audiology	226	69	24	cochlear_age	25.2%/74.8%
blance-scale	625	4	3	L	36.5%/63.5%
car evaluation	1728	6	4	unacc	70.0%/30.0%
cmc	1473	9	3	1	42.7%/57.3%
colic	368	22	2	no	37.0%/63.0%
dermatology	366	34	6	1	30.6%/69.4%
harberman	306	3	2	1	73.5%/26.5%
kr-vs-kp	3196	36	2	won	52.2%/47.8%
lymph	148	18	5	malign_lymph	41.2%/58.8%
mushroom	8124	22	2	e	51.8%/48.2%
nursery	12960	8	5	priority	32.9%/67.1%
page-blocks	5473	10	5	1	89.8%/10.2%
primary-tumor	339	17	21	lung	24.8%/75.2%
splice	3190	61	3	N	51.9%/48.1%
tic-tac-toe	958	9	2	positive	65.3%/34.7%
transfusion	748	4	2	0	76.2%/23.8%
vote	435	16	2	democrat	61.4%/38.6%
waveform	5000	40	3	0	33.8%/66.2%
zoo	101	17	7	mammal	40.6%/59.4%

examples. For datasets containing more than two classes, we select one of them as positive class, and all the other classes are used as negative class.

For numeric attributes, we discretize them using unsupervised 10-bin discretization in WEKA¹. For missing value, we handle them using ReplaceMissingValues in WEKA. In addition, we remove the attributes, whose number of values is almost equal to the size of training data, because such attributes rarely contribute to classification. There are two such attributes, attribute “animal” in dataset zoo and attribute “Instance_name” in splice. For each dataset, positive examples are randomly labeled positive with probability $1-a$ in P , and are left unlabeled with probability a . These examples, together with all the negative examples constitute the set of unlabeled examples U .

We evaluate the performance of classifier on the test set in term of F1, which is widely used in PU learning research[2,18]. We also give the results in terms of Accuracy and AUC. However, note that Accuracy cannot always reflect the real performance of our algorithms, because if most of the examples in a dataset are negative examples, even if Accuracy is high, the number of positive examples which are classified correctly may be small. In addition, a ranking of examples is more desirable in many real-world applications. For examples, we are more interested in the ranking of the potential future customers in direct marketing, and AUC is widely used to evaluate the performance of rankings generated by a classifier[21,22,23]. All the experiments are conducted on a PC with Core 2 CPU and 2.0 GB main Memory. In each group of experiment, ten trails of experiment are conducted, and the averaged results are reported.

4.1 PU Bayesian Algorithms vs. Supervised Bayesian Algorithms

In this group of experiment, in order to evaluate the performance of Bayesian algorithms for PU learning, we compare them to supervised Bayesian algorithms which use both

¹ <http://www.cs.waikato.ac.nz/ml/weka/>

positive and negative examples. Although Cavol *et al.* has proposed PNB and PTAN, the model for PU learning here is different from theirs and we do not require user-specified parameter. Furthermore, Cavol *et al.* did not compare PNB and PTAN with NB and TAN. Therefore, we also present the results of the comparison between them.

For supervised algorithms NB, TAN, HNB and AODE, we use the implementation in WEKA. For FBC, we use the source code provided by [6]. For the PU Bayesian algorithms proposed in Section 3 we implement them based on the WEKA framework. In both supervised and PU learning, 70% of the examples is used as training set and 30% of the examples is used as test set. However, in PU learning scenario, firstly, 50% of the examples is used as training set and 20% is used as validation set to estimate p ; and then the classifiers are retrained on the combined training and validation set. For datasets containing more than two class labels in Table 1 the supervised algorithms are run on the datasets after preprocessing, which contains only two class labels.

We write UnLevel for the proportion of unlabeled examples in training data and simulate different PU learning scenario by UnLevel. We experiment with UnLevel=40%, 50%, 60%, 70%, 80% and compare the performance of PU Bayesian algorithms with supervised ones. Because of limited space, we only give the averaged performance of the algorithms on all the experimented datasets, which are shown in Table 2.

Table 2. Experimental Results on F1, Accuracy and AUC

UnLevel Performance		NB	PNB	TAN	PTAN	HNB	PHNB	AODE	PAODE	FBN	PFBN
40%	F1	0.8966	0.8995	0.9055	0.9047	0.8993	0.8989	0.9093	0.9009	0.9080	0.9056
	Accuracy	83.82	83.71	84.76	84.68	83.99	83.82	85.38	84.18	85.14	84.47
	AUC	0.8309	0.8186	0.8221	0.8110	0.8072	0.8087	0.8274	0.8166	0.8369	0.8241
50%	F1	0.8894	0.8936	0.9067	0.8952	0.9030	0.8985	0.9076	0.8974	0.9035	0.9027
	Accuracy	84.00	83.98	86.07	84.39	85.62	84.87	86.21	84.70	85.59	84.91
	AUC	0.8563	0.8455	0.8625	0.8419	0.8588	0.8515	0.8685	0.8505	0.8626	0.8528
60%	F1	0.8659	0.8713	0.8875	0.8785	0.8838	0.8805	0.8876	0.8771	0.8861	0.8863
	Accuracy	83.76	82.63	86.21	83.70	85.95	84.19	86.20	83.57	86.19	84.25
	AUC	0.8639	0.8491	0.8721	0.8457	0.8698	0.8574	0.8764	0.8581	0.8740	0.8535
70%	F1	0.8614	0.8528	0.8676	0.8418	0.8745	0.8499	0.8776	0.8471	0.8712	0.8582
	Accuracy	85.62	83.59	86.68	82.96	87.08	83.82	87.40	83.56	87.04	84.24
	AUC	0.8889	0.8733	0.8907	0.8587	0.8940	0.8733	0.8990	0.8730	0.8950	0.8734
80%	F1	0.8573	0.8362	0.8615	0.8238	0.8708	0.8353	0.8703	0.8138	0.8692	0.8403
	Accuracy	86.61	83.56	87.46	82.83	88.02	84.17	88.15	82.87	87.98	83.88
	AUC	0.8972	0.8730	0.8987	0.8588	0.9021	0.8701	0.9061	0.8668	0.9032	0.8745

It can be observed from Table 2 that comparing with supervised Bayesian algorithms, generally speaking, when UnLevel=40%,50%,60%, F1 of PU Bayesian algorithms decrease within 1%, AUC and Accuracy decrease within 2%. When Unlevel=70%, the decrement of F1 and AUC for PTAN, PAODE and PHNB is about 2%~3%, while the decrement of Accuracy is about 3%~4%. The performance of PNB and PFBC decrease relatively less, about 1% for F1, 1%~2% for AUC and 2%~3% for Accuracy. When Unlevel=80%, F1, AUC and Accuracy of PTAN, PAODE and PHNB decrease about 3%~4% while F1 and AUC of PNB decrease about 2%~3%, and Accuracy decrease about 3%~4%. Overall, the five Bayesian algorithms for PU learning perform well

compared with supervised ones, especially in terms of F1 with minimum decrement. Additionally, PNB and PFBC are relatively robust against unlabeled data.

We also find that PFBC can generally remain best performance in different scenarios, especially in terms of F1, which is the highest in all cases. Although the performance of other PU Bayesian algorithm may be comparable with PFBC, they cannot perform consistently well in different scenarios.

4.2 Experiment on a

In this group of experiment, we study the performance of the Bayesian algorithms for PU learning with different values of a . We set $a=0.2,0.4,0.6$, and the averaged results on all the datasets in Table 1 is shown in Table 3.

Table 3. Experiment on a

Performance	a	PNB	PTAN	PHNB	PAODE	PFBC
F1	0.2	0.8594	0.8584	0.8668	0.8595	0.8680
	0.4	0.8460	0.8345	0.8455	0.8326	0.8529
	0.6	0.8199	0.7915	0.8140	0.7727	0.8215
Accuracy	0.2	85.65	85.98	86.81	86.55	86.55
	0.4	84.84	84.45	85.44	85.04	85.44
	0.6	83.25	81.99	83.68	82.17	83.18
AUC	0.2	0.8951	0.8907	0.8972	0.8957	0.9016
	0.4	0.8857	0.8753	0.8881	0.8868	0.8897
	0.6	0.8733	0.8567	0.8713	0.8689	0.8738

As shown in Table 3, with the increasing of a , the performance of all algorithms decline. The reason is that with growing number of positive examples used as unlabeled examples, the size of positive examples, $|P|$, becomes less and less, which undermines the estimation of the conditional probabilities. Therefore, the performance of Bayesian classifiers is decreasing.

4.3 Experiment on p

In this group of experiment, we study the performance of the Bayesian algorithms for PU learning with different values of p . We set $a=0.4$ and $p=0.3,0.4,0.5,0.6,0.7$. The size of training data remain the same, while the size of positive examples varies. In order to satisfy the above requirements, we select a subset of examples from the original datasets and the size of the generated datasets is shown in Table 4. Due to lacking of space, we only give the averaged result of F1 on all datasets experimented with, which is illustrated in Fig. 1.

It can be observed from Fig. 1 that with the increasing of p , F1 of all algorithms increase. In other words, if the size of training data and the proportion of positive examples used as unlabeled examples remain the same, then, the larger the prior probability of positive class p is, the better the Bayesian algorithms for PU learning perform in terms of F1.

Table 4. Size of training data

Dataset	Size	Dataset	Size
adult	15000	mushroom	5500
audiology	80	nursery	6000
balance-scale	400	page-blocks	800
car evaluation	700	primary-tumor	120
cmc	1200	splice	2000
colic	180	tic-tac-toe	450
dermatology	160	transfusion	250
haberman	100	vote	240
kr-vs-kp	2000	waveform	2400
lymph	80	zoo	50

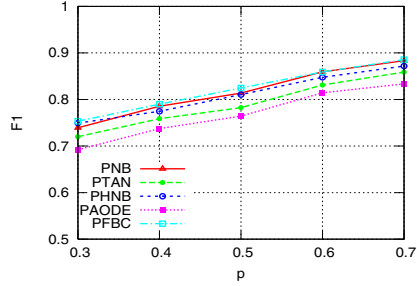


Fig. 1. Experiment on p

4.4 Evaluation on Time and Space

In this group of experiment, we evaluate the running time and memory usage of the five Bayesian algorithms for PU learning to build classifiers. We set $a=0.4$, and Fig. 2 shows the results on 10 datasets in Table 1, which include a wide range of the number of attributes and the size of training data. The experiment result of the rest of dataset is omitted here for lacking of space.

As shown in Fig. 2, PFBC takes more running time. The reason is that it requires learning not only the structure of full BN, but also CPT-tree for each attribute in each subset of training data. Additionally, the running time of PTAN is basically the longest one among the rest algorithms because it requires structure learning while PNB, PHNB and PAODE do not.

Furthermore, PNB takes the least memory usage basically while the memory usage of PFBC is quite large on dataset mushroom, splice and waveform, where the size of these datasets and the number of attributes are relatively large. This is because in PFBC, we need to learn CPT-tree for each attribute and the examples are required to be stored in the process of learning CPT-trees.

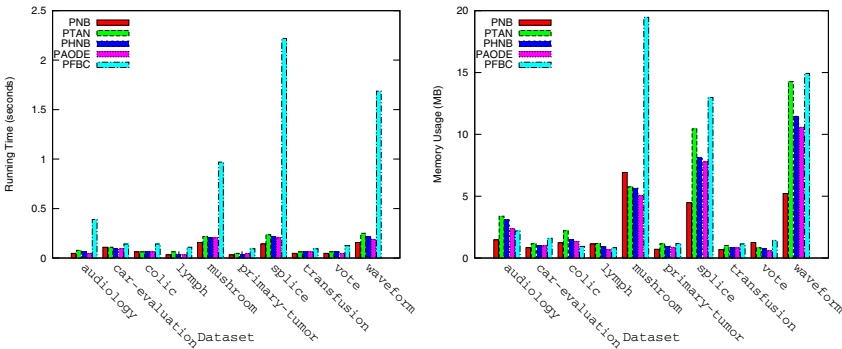


Fig. 2. Evaluation on Time and Space

5 Conclusion and Future Work

Existing Bayesian algorithms for PU learning require the user-specified parameter, which is general difficult for the user to provide in practice. In this paper, we utilize a method to avoid this parameter under the “selected completely at random” assumption, and reformulate algorithms PNB and PTAN under this assumption. Furthermore, we extend supervised Bayesian algorithms HNB, AODE, FBC to cope with PU learning scenarios. Experimental results show that the performance of the proposed Bayesian algorithms for PU learning are comparable to corresponding supervised ones in most cases. Additionally, PNB and PFBC are more robust against unlabeled data, and PFBC generally performs the best. In the future, we will extend more complex Bayesian classifiers to PU learning scenarios.

References

1. Zhang, D., Lee, W.S.: A Simple Probabilistic Approach to Learning from Positive and Unlabeled Examples. In: Proc. of UKCI 2005, pp. 83–87 (2005)
2. Elkan, C., Noto, K.: Learning Classifiers from Only Positive and Unlabeled Data. In: Proc. of SIGKDD 2008, pp. 213–220 (2008)
3. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. *Machine Learning* 29(2), 131–163 (1997)
4. Webb, G.I., Boughton, J.R., Wang, Z.: Not So Naive Bayes: Aggregating One-Dependence Estimators. *Machine Learning* 58(1), 5–24 (2005)
5. Jiang, L., Zhang, H., Cai, Z.: A Novel Bayes Model: Hidden Naive Bayes. *IEEE Transactions on Knowledge and Data Engineering* 21(10), 1361–1371 (2009)
6. Su, J., Zhang, H.: Full Bayesian Network Classifiers. In: Proc. of the 23rd ICML, pp. 897–904 (2006)
7. Schölkopf, B., Platt, J., Shawe-Taylor, J., Smola, A., Williamson, R.: Estimating the Support of a High-Dimensional Distribution. *Neural Computation* 13(7), 1443–1471 (2001)
8. Yu, H., Han, J., Chang, K.C.: PEBL: Positive Example Based Learning for Web Page Classification Using SVM. In: Proc. of the 8th SIGKDD, pp. 239–248 (2002)
9. Liu, B., Lee, W.S., Yu, P.S., Li, X.: Partially Supervised Classification of Text Documents. In: Proc. of the 9th ICML, pp. 387–394 (2002)
10. Li, X., Liu, B.: Learning to Classify Texts Using Positive and Unlabeled Data. In: Proc. of the 18th IJCAI, pp. 587–592 (2003)
11. Lee, W.S., Liu, B.: Learning with Positive and Unlabeled Examples Using Weighted Logistic Regression. In: Proc. of the 3rd ICDE, pp. 448–455 (2003)
12. Liu, B., Dai, Y., Li, X., Lee, W.S., Yu, P.S.: Building Text Classifiers Using Positive and Unlabeled Examples. In: Proc. of the 3rd ICDM, pp. 179–186 (2003)
13. Denis, F., Gilleron, R., Tommasi, M.: Text Classification from Positive and Unlabeled Examples. In: Proc. of the 9th IPMU, pp. 1927–1934 (2002)
14. Denis, F., Gilleron, R., Letouzey, F.: Learning from Positive and Unlabeled Examples. *Theoretical Computer Science* 38(1), 70–83 (2005)
15. Zhang, Y., Li, X., Orlowska, M.: One-Class Classification of Text Streams with Concept Drift. In: Proc. of ICDMW, pp. 116–125 (2008)
16. Li, X.L., Yu, P.S., Liu, B., Ng, S.K.: Positive Unlabeled Learning for Data Stream Classification. In: Proc. of the 9th SIAM SDM, pp. 257–268 (2009)
17. He, J., Zhang, Y., Li, X., Wang, Y.: Naive Bayes Classifier for Positive Unlabeled Learning with Uncertainty. In: Proc. of the 10th SIAM SDM, pp. 361–372 (2010)

18. Calvo, B., Larranaga, P., Lozano, J.A.: Learning Bayesian Classifiers from Positive and Unlabeled Examples. *Pattern Recognition Letters* 28(16), 2375–2384 (2007)
19. Zadrozny, B., Elkan, C.: Transforming Classifier Scores into Accurate Multiclass Probability Estimates. In: *Proc. of the 8th SIGKDD*, pp. 694–699 (2002)
20. Blake, C.L., Merz, C.J.: UCI repository of machine learning databases, <http://www.ics.uci.edu/~mllearn/MLRepository.html>
21. Zhang, H., Jiang, L., Su, J.: Augmenting Naive Bayes for Ranking. In: *Proc. of the 22nd ICML*, pp. 1020–1027 (2005)
22. Zhang, H., Jiang, L., Su, J.: Learning Weighted Naive Bayes with Accurate Ranking. In: *Proc. of the 4th ICDM*, pp. 567–570 (2004)
23. Su, J., Zhang, H.: Learning Conditional Independence Tree for Ranking. In: *Proc. of the 4th ICDM*, pp. 531–534 (2004)

Measuring Social Tag Confidence: Is It a Good or Bad Tag?

Xiwu Gu¹, Xianbing Wang¹, Ruixuan Li¹, Kunmei Wen¹, Yufei Yang¹,
and Weijun Xiao^{2,*}

¹ Intelligent and Distributed Computing Lab, College of Computer Science and Technology
Huazhong University of Science and Technology, Wuhan 430074, P.R. China

{guxiwu, rxli, kmwen}@mail.hust.edu.cn,

{wxbl225, yfyang}@smail.hust.edu.cn

² Department of Electrical and Computer Engineering, University of Minnesota, twin cities
200 Union Street SE, Minneapolis, MN 55455, US

wxiao@umn.edu

Abstract. Social tagging is an increasingly popular way to describe latent semantic information of web resources and thus is widely used to improve the performance of information retrieval system. However, there also has been significant variance of the quality of social tags because they can be annotated by folks on the web freely. As a consequence, how to measure the quality of social tags (referred to as social tag confidence) becomes an important issue. In this paper, we propose a statistic model to measure the confidence of social tags by utilizing a combination of three attributes of a social tag: web resource, tag, and tagging user. In order to evaluate the effectiveness of our model, two experiments are performed with datasets crawled from del.icio.us. Experimental results show that our model has a better performance than other approaches with respect to Normalized Discounted Cumulated Gain (NDCG). In addition, F-1 measure of tagged web page clustering performance is also increased when our model is applied to filter the noisy social tags with low tag confidence.

Keywords: tag confidence, semantic similarity, clustering, information retrieval.

1 Introduction

Along with the dramatic increase in the amount of web resources during recent years, web users are overwhelmed by the massive contents on the web. It is a non-trivial task for users to search and select information of their favorites. On the other hand, social tags have emerged as a valuable complement data source because to some extent they can describe latent semantic information of web resources. Consequently, social tags are widely used to improve information retrieval system effectively in various ways, such as optimizing page ranking using social tag [1][2], retrieving more semantically related images for an image retrieval system by exploiting textural social tag information of images[3], enhancing classifying and clustering effectiveness of

* Correspondence author.

web resources by means of social tag exploration[4][5], and utilizing social tag to improve language model for information retrieval[6].

However, social tag is a form of folksonomy, which refers to an open Internet-based collaborative method allowing folks on the web to describe or categorize web resources such as web pages, photographs and videos with text labels freely. It can be viewed as metadata of web resources generated by normal web users, which means there is a significant difference of quality between social tag and professional creation of metadata. Instead of being generated by specialists with expertise, social tags are usually created by the user arbitrarily without any control or expertise knowledge. Therefore, social tags also reveal problems caused by ambiguity and synonymy. Mathes et al. [7] cited several examples of ambiguous tags and synonymous tags in Delicious. For instance, the tag "Tiger" is used by many users to annotate web resources about creature tiger. However, some users may use it to tag web resources about Tiger Woods, who is an American professional golfer. Synonymous tags, like "mac" and "macintosh", "blog" and "weblog" are also widely used. On the other hand, some social tags are more likely created based on personal subjective judgment [8] so that these social tags have no semantic association with the tagged web resource. Besides, there also have been other problems of social tags such as mis-tagging due to spelling errors and tagging the same web resource cross different language. As a consequence, there are a large amount of noisy tags with low confidence on the web. More seriously, noisy social tags with low confidence will reduce the performance when they are utilized in IR system.

In order to deal with the problems of social tags, tag recommendation technique has been proposed and widely used to help user to annotate web resource more accurately [9]. However, tag recommendation is mainly based on the analysis the contents of web resource, the credibility of social annotation user usually is not taken into account.

In this paper, we propose a model to measure the confidence of social tags so that it can identify and filter the noisy tags with low confidence. Our model is based on the following assumptions: (1) Tags created by the user with higher credibility should have higher confidence score. (2) The tags with higher semantic relation with tagged resource should have higher confidence score. Following these assumptions, we utilize a combination of three attributes of social tag: resource, tag and tagging user for measuring the confidence of social tags. In order to evaluate our model, we conducted two experiments on a dataset crawled from del.icio.us. Experimental results show that our model can achieve much higher NDCG value for tag ranking and improve the accuracy for web page clustering after filtering the noisy tags with lower tag confidence. To the best of our knowledge, this is the first work to measure the tag confidence for enhancements of an information retrieval system by making full use of three attributes of social tag: resource, tag, and tagging user.

The rest of the paper is organized as follows. Section 2 summarizes related work on social tags and the applications of confidence of web resources and social tags. Section 3 sets out the definition of the confidence of social tags. The core of our paper, Section 4, proposes a method that is used to measure the confidence of social tags. Section 5 presents the experiments and numerical results. Finally, we draw a conclusion and discuss future work in Section 6.

2 Related Work

2.1 Research on Social Tags

As a new way of user-generated data, social tag can benefit many applications, such as information retrieval [10][11], semantic web [12], web page clustering [13] and user interest mining [14].

Besides the work mentioned above, Krestel et al. [15] introduce an approach based on Latent Dirichlet Allocation (LDA) to extract latent topics from resources with a fairly stable and complete tag set to recommend topics for new resources with only a few tags, which is more useful for searching and recommending resources to users. Ramage et al. [5] address one central question of how tagging data can be used to improve web document clustering and propose a novel generative clustering algorithm based on Latent Dirichlet Allocation. Körner et al. [16] analyze the influence of individual tagging practices in collaborative tagging systems on the emergence of global tag semantics. They raise a hypothesis that the quality of the emergent semantics depends on the pragmatics of tagging and propose four different statistical measures to assign users to two broad classes of categorizers and describers. Koutrika et al. [17] present that tagging systems become more susceptible to tag spam, they propose a framework to model tagging systems and user tagging behavior, and describe a novel approach to rank documents based on taggers' reliability.

Additionally, social tags are also widely used in the image retrieval area. Wu et al. [18] introduce a tag recommendation approach based on the learning, which generates ranking features from multi-modality correlations, and learns an optimal combination of these ranking features from different modalities. Liu et al. [19] propose a tag ranking approach in which the tags of an image can be automatically ranked according to their relevance with the image content by performing a random walk over a tag similarity to refine the relevance scores.

2.2 Research on the Confidence of Web Resources and Social Tags

The research on the confidence of web information has not been explored widely in the information retrieval area. Akamine et al. [20] describe an information confidence analysis system named WISDOM, which can automatically evaluate the confidence of information available on the Web from multiple viewpoints.

Meanwhile, there also has been a few works on the confidence (or the quality) of social tags. Lee et al. [8] propose a method measuring tag confidentiality from visual semantics of image by analyzing the associated visual information and ontology information in image retrieval. Zhu et al. [21] use tag to summarize web document, they measure tag confidence by calculating the user-tag adjacency matrix iteratively. This work has achieved some interesting results, however, this kind of tag confidence measurement doesn't consider the contents of web pages. In order to identify appropriate tags and eliminate spam and noise, Xu et al. [22] formulate some general criteria for a good tagging system, and by using these criteria, they propose a collaborative tag suggestion algorithm to single out high quality tags.

Different from the works mentioned above, we propose a model to measure the confidence of social tags using the combination of the credibility of users, the content items of web pages, and the relationships among users, tags, and resources.

3 Definition of Tag Confidence

In this paper, the confidence of a tag t with respect to a web page d is denoted by $Conf(t, d)$, which represents how much the tag t is semantically related to the tagged web page d . The value of $Conf(t, d)$ is ranged from 0 to 1. If the tag t is quite bound up with the web page d , its confidence value is close to 1. Otherwise, tag confidence value is close to 0.

The main idea of our work is inspired by the paper of Lee et al. [8], in which they propose a method measuring tag confidence from visual semantics of image by analyzing the associated visual information and ontology information in image retrieval. However, they haven't considered the impact of tagging user on the confidence of tag.

Intuitively, $Conf(t, d)$ should be determined by the following three factors:

- (1) The credibility of the tagging user. Since social tags of web resources are created by users, the confidence of tags will be determined by the tagging users. The higher credibility of the user, the higher confidence of the tag generated by the user should be.
- (2) The semantic similarity between web pages. Tag t of web page d can be viewed as keywords or summarization of d . Given a web page d' which is mostly semantically similar to d , if the tag t' has high confidence with respect to web page d' and tag t' is highly similar to the tag t , then tag t should also has high confidence with respect to web page d . Therefore it is necessary to assess the semantic of web page content items in order to measure the confidence of a tag.
- (3) The semantic similarity between tags. Under the above consideration, the semantic similarity between tags will also make contribution to the measurement of tag confidence. In order to calculate the semantic similarity between tags, the co-occurrence relationship between tags can be utilized. Two tags have co-occurrence relationship when they are annotated to a same web page.

Based on these considerations, the confidence value of social tag t with respect to document d is determined by a function F with three factors, which can be defined by:

$$Conf(t, d) = F(C(u), CS(d, d'), TS(t, t')) \quad (1)$$

where $Conf(t, d)$ is the confidence value of tag t with respect to d , $C(u)$ is the credibility of user u , $CS(d, d')$ is the semantic similarity between web page d and d' , $TS(t, t')$ is the semantic similarity between tag t and t' , tag t and t' are annotated to the web page d and d' , respectively. In particular, the higher value of three factors of F , the higher value of $Conf(t, d)$ should be.

In the next section, we will provide a specific implementation of this idea to measure the confidence of social tags.

4 Proposed Method

As mentioned in section 3, we will use a combination of the credibility of users, the semantic similarity among web pages and the semantic similarity among tags to measure the confidence of social tags. The calculating procedure of our model is described as follows.

In order to calculate the confidence of a target tag t with respect to a target web page d , we need to calculate the credibility of users who use the target tag t to annotate the target web page d at first. In the next step, we select top N semantically similar web pages of the target web page d by calculating the semantic similarity between d and each other web page in our dataset based on vector space model. Furthermore, we build a similarity measurement model based on probability statistics to calculate the semantic similarity between each pair of tags, in which one is annotated to the target web page d and the other is annotated to the one of the selected top N semantically similar web pages. Finally, we use a model named UCTM (user, page content items and social tag) to combine the aforementioned three results to calculate the confidence of target tag t with respect to a target web page d . The procedure of our method is illustrated in Fig. 1.

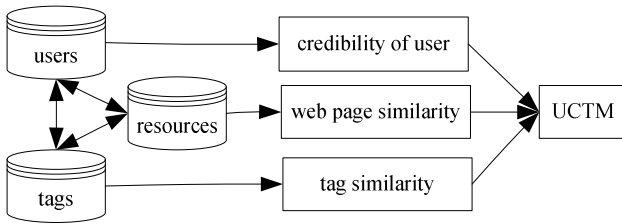


Fig. 1. The procedure for calculating social tag confidence

4.1 The Credibility of Users

In general, users play a significant role in social annotation system. A high credibility user is apt to create high quality tags. Therefore, the higher credibility of a user, the higher confidence of social tags generated by the user will be.

Based on this consideration, we quantify the credibility of users through an iterative model. The iterative algorithm is inspired by the work of Zhu et al [21]. They propose an algorithm named EigenTag to calculate user scores and tag scores. They deem that there exists a mutually reinforcing relationship between users and tags and describe this relationship as a tag-user graph with an adjacency matrix. However, they only consider the relationship between user and tag, the annotated web resource has not been taken into account. In order to calculate the credibility of users, we firstly model the relationships between users, tags and resources by three adjacency matrixes, as shown in Fig. 2. The relationships are illustrated by the user-tag-resources graph and the three corresponding adjacency matrixes M , N , P . which are tag-user matrix, resource-tag matrix and user-resource matrix, respectively.

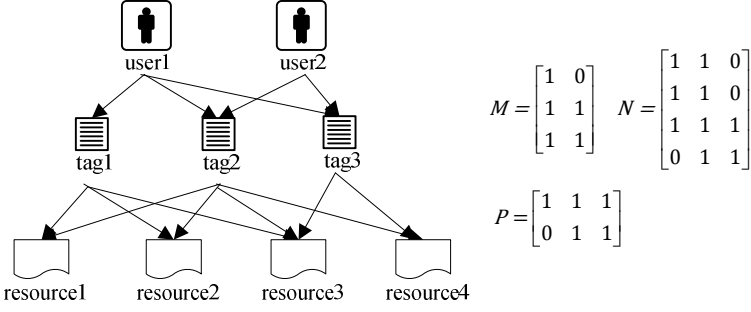


Fig. 2. Users-tags-resources relationship graph and adjacency matrixes

If a user u uses a tag t to annotate resource r , then the entry (t, u) of matrix M , the entry (r, t) of matrix N and the entry (u, r) of matrix P are all equals to 1. Let U be the vector of user scores $(u_1, u_2, \dots, u_x)^T$, T the vector of tag scores $(t_1, t_2, \dots, t_y)^T$, R the vector of resource scores $(r_1, r_2, \dots, r_z)^T$. Then the user score can be calculated by the following equations iteratively:

$$T = MU \quad R = NT \quad U = PR \quad (2)$$

We initialize the vector U as $(1, 1, \dots, 1)$, which means that we are impartial to every user at first and they have the same credibility. In each iteration step, the score of each tag is updated to the sum of the scores of all users annotating it, the score of each resource is updated to the sum of scores of all tags annotated to it, and the score of each user is updated to the sum of scores of all resources annotated by the user. The three vectors need to be normalized after each iteration step. We iterate the above three equations until the user scores converge. The credibility of a user will be set to the corresponding score in the final user score vector.

4.2 Select Top N Semantically Similar Web Pages

In order to select top N similar web pages of the target web page, we model a web page as a vector in vector space and calculating the similarity of each pair of web pages. There are two general approaches to weighting the components of vector: tf and tf-idf. In our paper, we use term frequency to weight the components of each vector because tf-idf approach over-emphasizes the rarest terms.

Once we have vector representation of web pages, we apply cosine similarity to calculate the semantic similarity between the target web page and the each other web page in our dataset. We set a threshold to select top N web pages which are the most similar to the target web page.

4.3 Tag Similarity

Social annotation is viewed as a set of triples constituted by users, tags and resources. Each triple (u, r, t) represents user u annotates resource r with tag t . The similarity measure between each pair of tags can be elicited by exploiting the relationships of

these triples. In order to simplify the calculation, we use dimension reduction method and represent a tag as a vector of resources.

The tag similarity calculation is based on the work proposed by Markines et al. [23]. We define the notations as follows: t represents a tag and T is its vector representation with resource components r_1 to r_n . In order to weight the components of vector T , we use the entropy of resource r , denoted by $-\log p(r)$, as the weight of the component r in vector T , where $p(r)$ is the fraction of tags annotated to r . Additionally, since there may be many users who use the tag t annotating the resource r , we need to take the number of users into account in order to enlarge the weight of the resource r . Hence the vector T of tag t can be represented as:

$$T = (R_1, R_2, \dots, R_n) \quad (3)$$

$$R_i = -N(t, r_i) * \log p(r_i) \quad (4)$$

where $N(t, r_i)$ is the number of users who use tag t to annotate resource r_i .

However, the Eq. (4) will encounter such a problem that if two tags are not annotated to the same web page, the measure of similarity will yield a zero similarity. In order to resolve the problem of zero similarity, it is reasonable to make the assumption that if many users all employ the same pair of tags, the pair of tags might be related even the pair of tags is not tagged to the same web page. Based on this assumption, we add R^* to the vector T :

$$T = (R_1, R_2, \dots, R_n, R^*) \quad (5)$$

$$R_i = -N(t, r_i) * \log p(r_i) \quad (6)$$

$$R^* = -\log [1/(N(t) + \lambda)] \quad (7)$$

where $N(t)$ is the number of users who own the tag t , λ is a smooth parameter which is not less than 1.

When we obtain the vector representation of two tags, we can use cosine theorem to calculate the similarity of two tags, the formula is given by

$$\theta(t_1, t_2) = \frac{T_1}{\|T_1\|} * \frac{T_2}{\|T_2\|} \quad (8)$$

where $\theta(t_1, t_2)$ represents the semantic similarity between tag t_1 and tag t_2 .

4.4 Evaluating Tag Confidence

In order to measure the confidence of tag t with respect to the target web page d , we build a model named UCTM which combine three attributes of social tag: the credibility of user, the semantic similarity among web pages, and the semantic similarity among social tags.

For a target web page TF with tag t , we can get top N similar web pages, which are denoted by LF_1, LF_2, \dots, LF_n , respectively. Suppose each web page $LF_i (i=1, 2, \dots, n)$ is annotated by tags $lt_1, lt_2, \dots, lt_{im}$, where im is the number of tags annotated to page LF_i . The confidence of tag t with respect to the target web page TF can be defined as follows.

$$Conf(t, TF) = \frac{C(u)}{n} * \sum_{i=1}^n [CS(TF, LF_i) * \max(TS(t, lt_j) | j = 1, 2, \dots, im)] \quad (9)$$

$C(u)$ is calculated by:

$$C(u) = \max(C(u_1), C(u_2), \dots, C(u_n)) \quad (10)$$

where u_1, u_2, \dots, u_n are the users who use tag t to annotate web page TF , $C(u)$ is the credibility of user u whose credibility value is maximum among u_1, u_2, \dots, u_n , $CS(TF, LF_i)$ is the semantic similarity between web page TF and page LF_i , $TS(t, lt_j)$ is the semantic similarity between tag t and tag lt_j .

For each top N similar web page LF_i , we firstly calculate the maximum tag similarity among its tags and the target tag t . Next, we integrate the credibility of users, the web page similarity between LF_i and TF , maximum tag similarity to calculate intermediate results of each top N similar web page. Finally, the confidence of tag t with respect to the target web page TF is calculated by averaging all intermediate results of the top N similar web pages. The higher confidence value signifies that the tag is highly related to associate web page.

5 Performance Evaluation

5.1 Experimental Settings and Evaluation Measurement

The experiments are conducted on a dataset partly crawled from the social annotation site del.icio.us during July, 2010. After dataset preprocessing, the experimental dataset consists of total number of 21770 tags, 14818 web pages and 20195 users.

We evaluated our model with two different measurements. The first measurement is Normalized Discounted Cumulated Gain (NDCG), a kind of ranking accuracy performance measure which shows how exactly an ordered list of objects ranked automatically by a certain criteria matches the ordered list ranked manually by users [24]. Once having calculated the confidence of all social tags with respect to each document in our dataset, tags of each document are ranked by confidence value. Then three hundred web pages are randomly selected from our dataset and 20 students are invited to label each tag of each web page as one of four levels of relevance: (1) irrelevant; (2) marginally relevant; (3) fairly relevant; (4) highly relevant. Given a web page annotated by the tags t_1, t_2, \dots, t_n , we ranked them according to the confidence value and the NDCG value is calculated by :

$$N_n = Z_n * \sum_{i=1}^n (2^{r(i)} - 1) / \log_2(1 + i) \quad (11)$$

where $r(i)$ is the relevance level of the i th tag and Z_n is a normalization constant. After computing the NDCG measures of each web page's tag ranking list, we average 20 students' NDCG value to obtain an evaluation measurement of tag confidence for each web page.

Another evaluation measurement is to apply the confidence value calculated by our model to filter the noisy tags which confidence value is lower than the predefined threshold. After that, remained tags with higher confidence are utilized for a practical application (web page clustering) to see that whether tags with higher confidence can improve the F-1 measure of clustering results.

5.2 NDCG Evaluation Result and Analysis

We use the following three tag ranking criteria as NDCG measurement comparison baseline of our tag confidence strategy.

- (1) Tag TF Frequency. The tag TF frequency is defined as how many times a tag is annotated to a web page. We rank the tags by tag TF frequency and compare the NDCG value with our model.
- (2) Tag TF-IDF Frequency. A tag's TF-IDF is the term frequency downweighted by the log ratio of the total number of web pages to the number of web pages annotating by that tag. We rank the tags by tag TF-IDF frequency and compare the NDCG value with our model.
- (3) EigenTag. In order to compare with the method proposed by Zhu et al. [21], we also have implemented the algorithm described in this paper and calculation result is compared with our model.

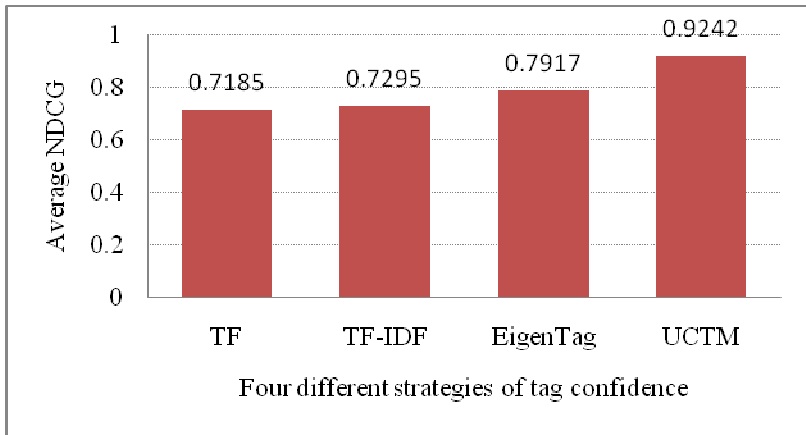


Fig. 3. Performance comparison of different strategies of tag confidence

The experimental results are shown in Fig. 3. One can see that our method outperforms TF, TF-IDF and EigenTag by about 29%, 27% and 17%, respectively. TF and TF-IDF only take statistical information of tags into count. EigenTag, which consider both users and tags, has better performance than TF and TF-IDF. However, it doesn't consider the semantic information of web page content items. Our model achieves the best performance because it considers the information from the quality of users, web page content items, and the relationships among users, tags, and resources.

5.3 Evaluation Results of Web Clustering

In order to evaluate our model more effectively, we took the web clustering algorithm proposed by Ramage et al. [5] as a baseline for comparison. Their method uses K-means clustering algorithm with the combination of social tags and web page words in three ways: Using Word-only, Using Tag-only and Using Tag+Word.

To evaluate the F-1 performance of clustering, we use a subset of partly crawled data collection from del.icio.us that is also presented in Open Directory Project (ODP) (<http://www.dmoz.org/>). ODP is the largest, most comprehensive human-edited directory of the Web. For every web page presented in ODP, we use the root category of that page as our clustering evaluation gold standard. The clustering evaluation measure we used is F-1 metrics [25].

First, we cluster the web pages without applying the strategy of tag confidence. In order to evaluate our model more objectively, we perform three experiments and in each experiment we use feature selection algorithm to select 500, 800, 1000 words, respectively. We also set the number of tags equal to the number of words. The F-1 values for Word-only, Tag-only, and Tag+Word are shown in Table 1.

Table 1. The clustering result not using the strategy of tag confidence

	tag, word = 500	tag, word = 800	tag, word = 1000
Word-only	0.1739	0.1904	0.1937
Tag-only	0.2101	0.2101	0.2103
Tag+Word	0.2160	0.2031	0.2039

Table 1 show that the performance of clustering using Tag-only is better than using Word-only. However, the performance of clustering using Tag+Word doesn't always outperform Tag-only method. The main reason is some words occurred in web pages act as noise so that they affect the performance of clustering.

Next, we apply the strategy of tag confidence to cluster our web pages. We firstly filter some tags whose confidence is relatively low for a web page. Next we mark the remained tags of a web page as Tag' and cluster the dataset using Tag'-only and Tag'+Word. The final results of clustering are shown in Table 2 and Table 3.

Table 2. The clustering result using Tag-only and Tag'-only

	tag, word = 500	tag, word = 800	tag, word = 1000
Tag-only	0.2101	0.2101	0.2103
Tag'-only	0.2150	0.2155	0.2153

Table 3. The clustering result using Tag+Word and Tag'+Word

	tag, word = 500	tag, word = 800	tag, word = 1000
Tag+Word	0.2160	0.2031	0.2039
Tag'+Word	0.2215	0.2242	0.2338

The clustering results show that our model can improve F-1 score by about 2.4% averagely when using Tag-only and F-1 score is increased more significantly by 2.5% to 14.7% when using Tag+Word. By examining the F-1 results, one can see that Tag'+Word has the best clustering result after applying the strategy of tag confidence to web page clustering. The reason is that there are some noisy tags with lower confidence

in our dataset which will downgrade the clustering performance and these noisy tags are filtered according their tag confidence calculated by our model. Thus we can obtain a new tag set with higher confidence which is more beneficial for web clustering.

6 Conclusion

During recent years, social tags are emerged as complement metadata source and widely used for improving the performance of an information retrieval system. However, there also has been considerable variation in the confidence of social tags while the confidence of social tags has a lot of impacts on the performance of information retrieval system. In this paper, we have proposed a novel measurement model of tag confidence for estimating the quality of social tags. In order to measure the confidence more accurately, the model takes full account of three attributes of social tag: resource, tag, and tagging user. Experimental results show that our tag confidence model can assess the quality of social tag more accurately than other methods. In addition, the F-1 measure of tagged web page clustering performance is also increased when our model is applied to filter the noisy social tags.

Acknowledgments. This work is supported by National Natural Science Foundation of China under Grant 60873225, 60773191, 70771043, National High Technology Research and Development Program of China under Grant 2007AA01Z403.

References

1. Bao, S., Xue, G., Wu, X., Yu, Y., Fei, B., Su, Z.: Optimizing web search using social annotations. In: The 16th International Conference on World Wide Web (WWW 2007), pp. 501–510. ACM Press, Banff (2007)
2. Park, L., Ramamohanarao, K.: Mining Web Multi-resolution Community-based Popularity for Information Retrieval. In: The 16th International ACM Conference on Conference on Information and Knowledge Management (CIKM 2007), pp. 545–554. ACM Press, Lisboa (2007)
3. Wang, C., Zhang, L., Zhang, H.: Learning to Reduce the Semantic Gap in Web Image Retrieval and Annotation. In: The 31th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2008), pp. 355–362. ACM Press, Singapore (2008)
4. Noll, M.G., Meinel, C.: Exploring social annotations for web document classification. In: The 2008 ACM Symposium on Applied Computing (SAC 2008), pp. 2315–2320. ACM Press, Brazil (2008)
5. Ramage, D., Heymann, P.: Clustering the Tagged Web. In: The Second ACM International Conference on Web Search and Data Mining, pp. 54–63. ACM Press, Barcelona (2009)
6. Xu, S., Bao, S., Cao, Y., Yu, Y.: Using social annotations to improve language model for information retrieval. In: The 16th International ACM Conference on Conference on Information and Knowledge Management (CIKM 2007), pp. 1003–1006. ACM Press, Lisboa (2007)
7. Mathes, A.: Folksonomies - cooperative classification and communication through shared metadata. Computer Mediated Communication, LIS590CMC (Doctoral Seminar), Graduate School of Library and Information Science, University of Illinois Urbana-Champaign (2004)

8. Lee, S., Min, H., Lee, Y.B., Ro, Y.M.: Measurement of Tag Confidence in User Generated Contents Retrieval. In: Proc. of SPIE, pp. 7257–7262 (2009)
9. Wu, L., Yang, L., Yu, N.: Learning to Tag. In: The 18th International Conference on World Wide Web (WWW 2009), pp. 361–370. ACM Press, Marid (2009)
10. Zhou, D., Bian, J., Zheng, S., Zha, H., Giles, C.L.: Exploring social annotations for information retrieval. In: The 17th International Conference on World Wide Web (WWW 2008), pp. 715–724. ACM Press, Beijing (2008)
11. Schenkel, R., Crecelius, T., Kacimi, M., Michel, S., Neumann, T., Parreira, J.X., Weikum, G.: Efficient top-k querying over social-tagging networks. In: The 31th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2008), pp. 523–530. ACM Press, Singapore (2008)
12. Wu, X., Zhang, L., Yu, Y.: Exploring social annotations for the semantic web. In: The 15th International Conference on World Wide Web (WWW 2006), pp. 417–426. ACM Press, Edinburgh (2006)
13. Brooks, C.H., Montanez, N.: Improved annotation of the blogosphere via autotagging and hierarchical clustering. In: The 15th International Conference on World Wide Web (WWW 2006), pp. 625–632. ACM Press, Edinburgh (2006)
14. Li, X., Guo, L., Zhao, Y.E.: Tag-based social interest discovery. In: The 17th International Conference on World Wide Web (WWW 2008), pp. 675–684. ACM Press, Beijing (2008)
15. Krestel, R., Fankhauser, P., Nejdl, W.: Latent Dirichlet Allocation for Tag Recommendation. In: The 3rd ACM Conference on Recommender Systems, New York, USA (2009)
16. Körner, C., Benz, D., Hotho, A., Strohmaier, M., Stumme, B.: Stop Thinking, Start Tagging: Tag Semantics Emerge from Collaborative Verbosity. In: The International Conference on World Wide Web (WWW 2010), pp. 251–260. ACM Press, USA (2010)
17. Koutrika, G., Effendi, F.A., Gyöngyi, Z., Heymann, P., Molina, H.G.: Combating Spam in Tagging Systems. In: The Third International Workshop on Adversarial Information Retrieval on the Web (AIRWeb 2007), Alberta, Canada (2007)
18. Wu, L., Yang, L., Yu, N.: Learning to Tag. In: The 18th International Conference on World Wide Web (WWW 2009), pp. 361–370. ACM Press, Marid (2009)
19. Liu, D., Hua, X., Yang, L.: Tag Ranking. In: The 18th International Conference on World Wide Web (WWW 2009), pp. 351–360. ACM Press, Marid (2009)
20. Akamine, S., Kawahara, D., Kato, Y.: WISDOM: A Web Information Credibility Analysis System. In: Proceedings of the ACL-IJCNLP, Suntec, Singapore, pp. 1–4 (2009)
21. Zhu, J., Wang, C., He, X., Bu, J., Chen, C., Qu, M., Lu, G.: Tag-Oriented Document Summarization. In: The 18th International Conference on World Wide Web (WWW 2009), pp. 1195–1196. ACM Press, Marid (2009)
22. Xu, Z., Fu, Y., Mao, J., Su, D.: Towards the Semantic Web: Collaborative Tag Suggestions. In: The 15th International Conference on World Wide Web (WWW 2006). ACM Press, Edinburgh (2006)
23. Markines, B., Cattuto, C., Menczer, F., Benz, D., Hotho, A., Stumme, G.: Evaluating similarity measures for emergent semantics of social tagging. In: The 18th International Conference on World Wide Web (WWW 2009), pp. 641–650. ACM Press, Marid (2009)
24. Jarvelin, K., Kekalainen, J.: Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems* 20(4), 422–466 (2002)
25. Manning, C., Raghavan, P., Schütze, H.: Introduction to information retrieval. Cambridge University Press, Cambridge (2008)

A New Vector Space Model Exploiting Semantic Correlations of Social Annotations for Web Page Clustering

Xiwu Gu¹, Xianbing Wang¹, Ruixuan Li¹, Kunmei Wen¹, Yufei Yang¹,
and Weijun Xiao^{2,*}

¹ Intelligent and Distributed Computing Lab, College of Computer Science and Technology
Huazhong University of Science and Technology, Wuhan 430074, P.R. China

{guxiwu, rxli, kmwen}@mail.hust.edu.cn,
{wxbl225, yfyang}@smail.hust.edu.cn

² Department of Electrical and Computer Engineering, University of Minnesota, twin cities
200 Union Street SE, Minneapolis, MN 55455, US
wxiao@umn.edu

Abstract. Text clustering can effectively improve search results and user experience of information retrieval system. Traditional text clustering approaches are based on vector space model, in which a document is represented as a vector using term frequency based weighting scheme. The main disadvantage of this model is that it cannot fully exploit semantic correlations between social annotations and document contents because term frequency based weighting scheme only captures the number of occurrences of terms in the document. However, social annotation of web pages implicates fundamental and valuable semantic information thus can be fully utilized to improve information retrieval system. In this paper, we investigate and evaluate several extended vector space models which can combine social annotation and web page text. In particular, we propose a novel vector space model by computing the semantic correlations between social annotations and web page words. Comparing with other vector space models, our experiments show that using semantic correlations between social tags and web page words improves the clustering accuracy with RI score increase of 4% ~ 7%.

Keywords: social annotation, clustering, information retrieval.

1 Introduction

Recent years the advance of Web 2.0 technology has influenced the ways that users interact with Internet. Unlike the traditional publishing-browsing style of interaction between users and Internet, Web 2.0 facilitates users the abilities of information sharing, exploration and collaboration on the Internet. This brings on the evolution of new web applications such as social annotations and social networking.

* Correspondence author.

In particular, a social annotation is referred to as an online annotation associated with a Web resource (typically a Web page, Web image and Web video). By means of online Web annotation tools, users can collaboratively add and modify text labels that describe or categorize Web resource without modifying the resource itself. At present, widely used online Web annotation tools include Delicious (annotating Web pages), Flickr (annotating Web images), YouTube (annotating Web videos), etc.

Essentially, social annotations can be viewed as socially classification layer building on the top of existing annotated Web resources. Moreover, social annotation is user-provided metadata implicating fundamental and valuable semantic information of Web resources. In recent years, many of academic research have been focused on social annotations [1]. The main purpose of these research works is to combine the analytical technique of social annotations with traditional information retrieval technologies to enhance the performance and user experience of information retrieval systems.

Generally, a social annotation is a triple consists of three elements: tag, user, and resource. In the triple of a social annotation, the resource is tagged by user based on the resource content; the tag (also referred as annotation) reflects the category of the resource; the user, by which the tag is provided, hides the latent social community. The semantic correlation among of elements of the triple can be utilized to improve the information retrieval system.

This paper aims to improve web page clustering accuracy for information retrieval system using social annotation. As the fundament of web page clustering, traditional vector space model (VSM) only takes the word occurrence frequency in the document into account. Due to the “Bag of Words” nature it cannot represent the contents of document more precisely. On the other hand, as a sort of user-provided metadata, social tag should be fully exploited to enhance the traditional VSM. Based on this consideration, we propose a social annotation based vector space model that makes use of the semantic of social annotations to cluster Web pages more accurately. In particular, our model is constructed by calculating the semantic correlations between social annotations and web page words. Different from the other enhanced VSM in which the words in a document and tags annotated to the document are simply treated with the same weighting, we project the calculated semantic correlations of social tags to the axis of vector space so that the axis with higher semantic correlation between tags and words has higher weight. This is the main contribution of our work.

Our work can be summarized as: we firstly calculate a semantic correlation matrix between social tags and web page words. Then a web page can be represented by a vector in three ways: (1) the axes of vector space are social tags and the coordinate of axis t for document d is determined by the semantic correlations between document d and tag t , which is denoted by $P(d, t)$. (2) the axes are web page words and the coordinate of axis w for document d is determined by the semantic correlations between document d and word w , which is denoted by $P(d, w)$. (3) the axes are words plus tags and the coordinates is weighted sum of $P(d, t)$ and $P(d, w)$. In order to evaluate the performance of our method, we compare it with other vector space models with K-means clustering algorithm. Our experiments show that our proposed model can improve the clustering accuracy by 4% ~ 7%.

Different from the existing works, the main contributions of our work are: (1) we exploited the semantic correlations between social tags and the words of web pages.

(2) we proposed a new vector space model by making full use of exploited semantic correlations of social annotations. This work will be an effort to help clustering the tagged web.

The rest of this paper is organized as follows: Section 2 introduces the related research work of social annotations. Section 3 proposes an extended vector space model by exploiting semantic correlations of social annotations. Experiments and numerical results are presented in Section 4. The conclusion is drawn in Section 5.

2 Related Work

Applying social annotations to web-based information systems is a hot topic and has been received a lot of attentions in recent years [2]. We briefly summarize the existing related works into the following three directions.

2.1 Language Models of Social Annotation

The research of probabilistic language model of information retrieval system is based on the idea which is originally proposed by Ponte and Croft [3]: Given a specific query, a document is a good match to query if the generative probabilistic language model of for the document is more likely to generate the query. Along with the emergence of social annotation, later research works on probabilistic language model seek to combine the probabilistic model of social annotation and annotated document content. Zhou et al. proposed a unified framework to combine the modeling of social annotations with the language modeling-based methods for information retrieval. The framework enhances document and query language models by incorporating user domain interests as well as topical background models [1]. Xu et al. analyzed two properties of social annotation, namely keyword property and structure property. These two properties of social annotations are leveraged for the use of language modeling with a mixture language model LAM (Language Annotation Model) and LAM is utilized to strengthen existing smoothing methods for the language model for information retrieval [4] [5].

2.2 Semantics of Social Annotation

The semantics of social annotations are captured by the following ways: calculating similarity between social annotations, establishing probabilistic model of social annotations and measuring the relationships among social annotations or annotated Web resources. Wu et al. established a global semantic model from social annotations which can be inferred from social annotations statistically [2]. Markines et al. evaluated similarity measures for emergent semantics [6] [7] of social annotations by building an evaluation framework to compare various general social annotations-based similarity measures statistically [8] [9]. Cattuto et al. analyzed several measures of tag similarity and used validated measures of semantic distance to characterize the semantic relation between the mapped tags [10].

2.3 Social Annotation Applications

The probabilistic language model and semantics of social annotations can be employed in many aspects of information retrieval systems such as ranking, classification, clustering to improve the effectiveness and user experience.

Similarity ranking measures the relevance between user query and resulting Web resources. Some proposed ranking models seek to improve query-resource similarity more precisely by making use of social annotation. Bao et al. proposed two novel algorithms SocialSimRank and SocialPageRank to incorporate social annotations into search result ranking [11]. Liu et al. proposed a tag ranking scheme to automatically rank the tags of images. Tag ranking scheme is applied to tag-based image search [12]. Schenkel et al. are focused on search ranking in social networks [13].

Since Social annotation provides a natural way for people to classify Web resources, some other research works try to explore this feature of social annotation to enhance the accuracy of Web resource classification and clustering. Pedro and Siersdorfer proposed a novel multi-modal approach for automatically ranking and classifying photos by exploiting image features and social annotations [14]. M.G. Noll and C. Meinel explored and studied the characteristics of social annotations with regard to their usefulness for Web document classification [15]. Shepitsen et al. presented a personalization algorithm for recommendation in folksonomies which relies on hierarchical social annotations clustering [16] [17]. Begelman et al. clustered social tags by defining a set of similarity measures among tags [18]. Ramage et al. used social annotations as complementary data source to improve automatic clustering of Web pages by means of combining social annotations with Web pages in an extended vector space model [19].

3 Vector Space Model Based on Semantic Correlation of Social Annotation

The objective of our work is to investigate how to exploit the semantic relationship between social annotations and annotated documents and how this semantic relationship can be employed to enhance vector space model so that the accuracy of web page clustering can be improved. Our work is mainly inspired by the previous work [16] [17] and [19].

In research work of [16] and [17], a hierarchical tag clustering algorithm is proposed based on tag's vector space model. In this tag-based vector space model, an annotated Web resource is modeled as a vector over the set of tags, and a user is modeled in the same way. Each component of a vector is calculated based on the well-known TF-IDF measure [20] of tags. To calculate the similarity between a user query and a Web resource, a query is also modeled as a unit vector consisting of a single tag, which is based on the assumption that the user interacts with the system by selecting a query tag and expects to receive resource recommendations. We argue that the content of annotated Web resource should also be considered in order to model the annotated web resource in tag-based vector space model more precisely, and in the case that user query consists of arbitrary terms, how to model user query in tag-based vector space model is still an unsolved problem.

On the other hand, the research work [19] proposed an extended vector space model, which is constructed in following ways with bags of words and tags: Words only, Tags Only, Words combining Tags. The combination of words and tags can be concatenation of l_2 -normalized word and tag vector with equal weight, treating term of tag as n terms of word or treat tags simply as additional words. We also argue that the combination of words and tags should consider the correlation between words and tags rather than treating them with equal weight. Another problem of word term based vector space model is that a document may contain many informationless terms, and these informationless terms are more likely to act as noises which will increase the errors of clustering.

3.1 Problem Definition

Since an annotated web page is associated with a group of tags, we describe an annotated web page D as a tuple:

$$D = \langle W, T \rangle \quad (1)$$

where $W = \{w_1, w_2, \dots, w_l\}$ is the set of words occurred in the web page D , $T = \{t_1, t_2, \dots, t_m\}$ is the set of social tags of the web page D .

With this document description, the goal of social annotation based web page clustering task can be defined as: Given a set of documents $\mathcal{D} = \{D_1, D_2, \dots, D_N\}$ and a target number of clusters K , we want to find an assignment function C :

$$C: \mathcal{D} \rightarrow \{1, 2, \dots, K\} \quad (2)$$

which maps each document in \mathcal{D} to a cluster number $x \in \{1, 2, \dots, K\}$, where $D_i (i = 1, \dots, N)$ is represented in a bi-tuple defined in (1).

3.2 Weighted Matrix of Social Annotation

There are two kinds of elements related to the social annotations: web page and social tag. Their relationship can be described by weighted tag-document matrix, which is illustrated in Figure 1.

	d ₁	d ₂	d ₃
t ₁	1	1	1
t ₂	1	1	1
t ₃	1	1	1

Fig. 1. The illustration of the relationship between web page and tag

As shown in Figure 1, the relationship between web page and tag can be described by binary matrixes, in which “1” element of i th row and j th column represents a relationship exists between corresponding element i and j . However, the matrix doesn’t consider the factor of frequency, that is, how many times a tag is annotated to

a web page. In order to count the factor of frequency, the tf-idf measure is applied to the $m \times n$ tag-document matrix A_{TD} .

$$A_{TD} = \begin{bmatrix} A_{TD}(1,1) & \cdots & A_{TD}(1,n) \\ \vdots & \ddots & \vdots \\ A_{TD}(m,1) & \cdots & A_{TD}(m,n) \end{bmatrix} \quad (3)$$

where m is the number of social tags and n is the number of annotated web pages. The matrix element $A_{TD}(i,j)$ denotes the relationship between the i th social tag and j th web page, which can be obtained as:

$$A_{TD}(i,j) = tf_{t_i,d_j} \times idf_{t_i} \quad (4)$$

where

$$idf_{t_i} = \log \frac{N}{df_{t_i}} \quad (5)$$

In formula (4) and (5), tf_{t_i,d_j} denotes how many times the tag t_i is applied to the web page d_j ; idf_{t_i} is the measure of the importance of the tag t_i in which N is the total number of annotated web pages and df_{t_i} denotes the number of web pages to which the tag t_i is applied.

Because social tags can be viewed as the terms marking social categorization of web pages, so an alternative choice is to assign the same importance for each tag. In this case idf_{t_i} can be ignored and $A_{TD}(i,j)$ is obtained as:

$$A_{TD}(i,j) = tf_{t_i,d_j} \quad (6)$$

3.3 Tag Similarity

The similarity measure between tags can be elicited by exploiting underline tags co-occurrence of matrix A_{TD} , in which element $A_{TD}(i,j)$ is a relationship measure for tag i and document j . Based on the assumption that semantically similar annotations are more likely assigned to the same documents, we can derive similarity matrix of tags from tag-document matrix A_{TD} , and we denote the similarity matrix of tags derived from A_{TD} by S_T :

$$S_T = A_{TD} \times A_{TD}^T \quad (7)$$

where the element $S_T(i,j)$ measure the similarity between tag t_i and tag t_j .

3.4 Semantic Correlation between Tags and Words

The matrix A_{TD} can be viewed as a semantic correlation matrix between social tags and documents. However, A_{TD} is only based on the statistical feature of tags. In order to characterize semantic correlation between tags and documents more precisely, the contents of documents should also be exploited.

An intuitive approach to exploit the contents of a document is based on the total times a tag annotated to the document. But this approach ignores the semantic correlation between the tag and word occurred in document. Our approach to

calculate the semantic correlation between tag and document is based on the tag-word correlation matrix.

In vector space mode, each document is viewed as vector in which each component corresponds to a word in dictionary with TF-IDF weight. We denote the $l \times n$ word-document matrix as A_{WD} :

$$A_{WD} = \begin{bmatrix} A_{WD}(1,1) & \cdots & A_{WD}(1,n) \\ \vdots & \ddots & \vdots \\ A_{WD}(l,1) & \cdots & A_{WD}(l,n) \end{bmatrix} \quad (8)$$

where l is the number of words in dictionary and n is the number of web pages. The word w_j can be represented as the j th row vector $\overrightarrow{V_{WD}}(w_j)$ in A_{WD} :

$$\overrightarrow{V_{WD}}(w_j) = [A_{WD}(j,1), \dots, A_{WD}(j,n)] \quad (9)$$

Similarly, the tag t_i can be represented as the i th row vector $\overrightarrow{V_{TD}}(t_i)$ in A_{TD} :

$$\overrightarrow{V_{TD}}(t_i) = [A_{TD}(i,1), \dots, A_{TD}(i,n)] \quad (10)$$

Since $\overrightarrow{V_{WD}}(w_j)$ and $\overrightarrow{V_{TD}}(t_i)$ are vectors in document based vector space, the semantic correlation between tag t_i and word w_j , which is denoted by $S_{TW}(t_i, w_j)$, can be measured as:

$$S_{TW}(t_i, w_j) = \overrightarrow{V_{TD}}(t_i) \times \overrightarrow{V_{WD}}(w_j)^T \quad (11)$$

Finally, we denote the semantic correlation matrix between tag and word by S_{TW} , and S_{TW} can be calculated as:

$$S_{TW} = A_{TD} \times A_{WD}^T \quad (12)$$

where the element $S_{TW}(i,j)$ measure the semantic correlation between tag t_i and word w_j .

3.5 Extended Vector Space Model

Based on the similarity matrix of tags and the semantic correlation matrix between tag and word, we can model web page as a vector in tag-based vector space, in which each social tag is axis and each component of the vector is determined by the projection of the web page with respect to each tag.

The projection of the web page with respect to each axis is calculated according to semantic correlation between web page and tag. Given a web page d and tag t , we denote the projection of document d with respect to axis t by $P(d, t)$. The vector of document d , denoted by $\overrightarrow{V_T}(d)$, is defined as:

$$\overrightarrow{V_T}(d) = [P(d, t_1), \dots, P(d, t_m)] \quad (13)$$

where m is total numbers of tags, and each component of $\overrightarrow{V_T}(d)$ is calculated by $P(d, t)$:

$$P(d, t) = \mu \times A_{DT}(d, t) + \varphi \times C(d, t) \quad (14)$$

μ and φ are all real constant between 0 and 1. Formula (14) shows that $P(d, t)$ is the linear combination of two parts: the first part, which denoted by $A_{TD}(t, d)$, is determined by how many times the tag t is applied to the document d ; another part, denoted by $C(d, t)$ is determined by the semantic correlation between tag t and document d .

Before giving the definition of $C(d, t)$, we firstly denote the set of tags by T and the set of words occurred in web pages by W , thus $C(d, t)$ is defined as:

$$C(d, t) = \sum_{w \in W} Sim(w, t) \quad (15)$$

and $Sim(w, t)$ is defined as:

$$Sim(w, t) = \begin{cases} S_T(w, t), & w \in T \\ S_{TW}(w, t), & w \in W - T \end{cases} \quad (16)$$

Similarly, we can also model web page as a vector in word-based vector space, in which each word occurred in web pages is axis and each component of the document vector is determined by the projection of the document with respect to each word. Given a document d and word w , we denote the projection of document d with respect to axis w by $P(d, w)$. The vector of document d , denoted by $\vec{V}_W(d)$, is defined as:

$$\vec{V}_W(d) = [P(d, w_1), \dots, P(d, w_n)] \quad (17)$$

where n is total numbers of words, and $P(d, w)$ is calculated by:

$$P(d, w) = \mu \times A_{DW}(d, w) + \varphi \times C(d, w) \quad (18)$$

where A_{DW} is the transpose of A_{WD} and $C(d, w)$ is calculated by

$$C(d, w) = \sum_{t \in T} Sim(w, t) \quad (19)$$

Another way to represent a web page is to combine the projection of the web page with respect to tag and word. Given a web page d , we calculate tag-based vector $\vec{V}_T(d)$ and word-based vector $\vec{V}_W(d)$, respectively. Then the web page d is modeled by a vector $\vec{V}_{T+W}(d)$, which is calculated by:

$$\vec{V}_{T+W}(d) = \mu \times \vec{V}_T(d) + \varphi \times \vec{V}_W(d) \quad (20)$$

When combining the $\vec{V}_T(d)$ and $\vec{V}_W(d)$ vector, we need to utilize feature selection algorithm to select words and tags and make the number of words and tags equal to each other.

4 Experiments and Numerical Results

In order to investigate the effectiveness of the clustering method under our social annotation based vector space model, we apply flat clustering algorithm K-means to our model to evaluate clustering accuracy. The evaluation is done by extensive experiments on real world data collections.

4.1 Data Collections

The target data collection is partly crawled from the social annotation site del.icio.us during July, 2010. Our data collections consist of 1502825 tags and 466871 web pages.

4.2 Evaluation Measure and Gold Standard

To evaluate the accuracy of clustering, we use subset of partly crawled data collection from del.icio.us which is also presented in Open Directory Project (ODP). ODP is the largest, most comprehensive human-edited directory of the Web. For every web page presented in ODP, we use the root category of that page as our evaluation gold standard. The evaluation measure we adopted is Rand Index (RI) [21]. RI penalizes both false positive and false negative decisions during clustering and it measures the accuracy of clustering result. If define A as the number of true positive documents, B as the number of false negative documents, C as the number of false positive documents, D as the number of true negative documents, and $A+B+C+D$ is the total number of document ,then RI is defined by

$$RI = \frac{A+D}{A+B+C+D} \quad (21)$$

4.3 Experiment Settings

The total category of crawled web page presented is 17 and we select 100, 200 and all documents in each category for evaluating. We also use feature selection algorithm to select 1000, 1500, 2000 tags for testing, respectively. And the number of words is set to be equal to the number of tags. The parameters μ and φ are all set to 0.5 in formula (14), (18), and (20). For the performance comparing, we also apply the K-means to the following models:

- (1) **Tag-only:** a document is modeled by tag-based tf-idf weighting vector.
- (2) **Word-only:** a document is modeled by word-based tf-idf weighting vector.
- (3) **Tag+Word:** a document is modeled by concatenation of l_2 -normalized word and tag vector with equal weight, which is proposed by Ramage et al. [19] and used as accuracy comparing baseline with our work.
- (4) \mathbf{V}_T : a document is modeled by the projection of web page with respect to each tag, which is defined by formula (13)-(16).
- (5) \mathbf{V}_W : a document is modeled by the projection of web page with respect to each word, which is defined by formula (17)-(19).
- (6) \mathbf{V}_{T+W} : a document is modeled by combining the projection of web page with respect to tag and word, which is defined by formula (20).

4.4 Experimental Results and Analysis

Our first experiment is to randomly select 100 documents from each category in our data collections. Based on selected documents, we perform K-means tests for six

different vector models described in Section 4.3. For each of tests, we use RI values to measure our clustering performance.

As shown in Table 1, Tag-only has the worst RI value, which means Tag-only has the lowest clustering accuracy. The reason is that social tags have limited information and bad quality. Another possible reason is social tags are so sparse that it is bad for our clustering performance. For Word-only and Tag+Word, both of them have better RI values than Tag-only under our expectations because they have detailed information for documents. More importantly, our proposed V_{T+W} model improves RI value further beyond Word-only and Tag+Word. The main reason is that V_{T+W} model considers the semantic correlations between social tags and web page words and it has much richer semantic information than other models. Another reason is there are noise words in either Word-only or Tag+Word models. Compared to other models, our model has up 29% better RI than Word-only and 22% better RI than Tag+Word. The increased value is calculated by averaging RI increase with tags number 1000, 1500 and 2000.

Table 1. Select 100 documents randomly from every category

	tag, word = 1000	tag, word = 1500	tag, word = 2000
Tag-only	0.1148	0.1064	0.1002
Word-only	0.7125	0.5533	0.7331
Tag+Word	0.8065	0.6502	0.6411
V_T	0.8458	0.8361	0.8380
V_w	0.6831	0.5925	0.6337
V_{T+W}	0.8450	0.8370	0.8553

Similarly, we picked up more documents for each category and conducted the second experiments. The result is shown in Table 2. Compared to other models, our model has up 42% better RI than Word-only and 34% better RI than Tag+Word averagely. Again, our model shows much better RI values than other models.

Table 2. Select 200 documents randomly from every category

	tag, word = 1000	tag, word = 1500	tag, word = 2000
Tag-only	0.0947	0.0899	0.0840
Word-only	0.8216	0.6388	0.6161
Tag+Word	0.7731	0.6342	0.5813
V_T	0.8574	0.8620	0.8635
V_w	0.8349	0.7483	0.7175
V_{T+W}	0.8677	0.8727	0.8765

Besides investigating how V_{T+W} can achieve much better clustering accuracy, we also conduct an experiment how sampling documents affect clustering accuracy. Without sampling documents, we select all the documents to conduct an experiment as shown in Table 3. This table shows that our model has 2% better RI value than Word-only averagely, and 4% ~7% better RI value than Tag+Word .

Table 3. Select all documents from every category

	tag, word = 1000	tag, word = 1500	tag, word = 2000
Tag-only	0.1402	0.1623	0.1317
Word-only	0.8277	0.8201	0.8262
Tag+Word	0.8163	0.7857	0.7809
V_T	0.8316	0.8333	0.8291
V_w	0.8230	0.8131	0.7676
V_{T+W}	0.8484	0.8400	0.8327

From Table 1, 2, and 3, one can find increasing samples can benefit RI values. However, when oversampling documents, it will be against the RI values. The possible reason is that some samples have noises information. This observation is consistent with the conclusion that noisy information is not so good for the improvement of clustering performance [22].

5 Conclusion

Social annotations of web pages contain usefully semantic information and thus can be utilized to improve the performance of information retrieval system. In this paper we exploit the semantic of social annotations by calculating the semantic correlation between social annotations and words in web pages. Furthermore, we propose a novel vector space model based on semantic correlation between social annotations and words, and apply the K-means clustering algorithm to our model to evaluate the clustering accuracy. Comparing with other vector space models, our model can achieve the best clustering accuracy by 4% ~ 7%.

Acknowledgments. This work is supported by National Natural Science Foundation of China under Grant 60873225, 60773191, 70771043, National High Technology Research and Development Program of China under Grant 2007AA01Z403.

References

1. Zhou, D., Bian, J., Zheng, S., Zha, H., Giles, C.L.: Exploring social annotations for information retrieval. In: The 17th International Conference on World Wide Web (WWW 2008), pp. 715–724. ACM Press, Beijing (2008)
2. Wu, X., Zhang, L., Yu, Y.: Exploring social annotations for the semantic web. In: The 15th International Conference on World Wide Web (WWW 2006), pp. 417–426. ACM Press, Edinburgh (2006)
3. Ponte, J.M., Croft, W.B.: A Language Modeling Approach to Information Retrieval. In: The 21th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1998), pp. 275–281. ACM Press, Melbourne (1998)
4. Xu, S., Bao, S., Cao, Y., Yu, Y.: Using social annotations to improve language model for information retrieval. In: The 16th International ACM Conference on Conference on Information and Knowledge Management (CIKM 2007), pp. 1003–1006. ACM Press, Lisboa (2007)

5. Xu, S., Bao, S., Yu, Y., Cao, Y.: Using Social Annotations to Smooth the Language Model for IR. In: Zhou, Z.-H., Li, H., Yang, Q. (eds.) PAKDD 2007. LNCS (LNAI), vol. 4426, pp. 1015–1021. Springer, Heidelberg (2007)
6. Aberer, K., Cudré-Mauroux, P., Hauswirth, M.: The chatty web: emergent semantics through gossiping. In: The 12th International Conference on World Wide Web (WWW 2003), pp. 197–206. ACM Press, Budapest (2003)
7. Mika, P.: Ontologies Are Us: A Unified Model of Social Networks and Semantics. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 522–536. Springer, Heidelberg (2005)
8. Markines, B., Cattuto, C., Menczer, F., Benz, D., Hotho, A., Stumme, G.: Evaluating similarity measures for emergent semantics of social tagging. In: The 18th International Conference on World Wide Web (WWW 2009), pp. 641–650. ACM Press, Marid (2009)
9. Markines, B., Menczer, F.: A scalable, collaborative similarity measure for social annotation systems. In: The 20th ACM Conference on Hypertext and Hypermedia, pp. 347–348. ACM Press, Torino (2009)
10. Cattuto, C., Benz, D., Hotho, A., Stumme, G.: Semantic Grounding of Tag Relatedness in Social Bookmarking Systems. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 615–631. Springer, Heidelberg (2008)
11. Bao, S., Xue, G., Wu, X., Yu, Y., Fei, B., Su, Z.: Optimizing web search using social annotations. In: The 16th International Conference on World Wide Web (WWW 2007), pp. 501–510. ACM Press, Banff (2007)
12. Liu, D., Hua, X., Yang, L., Wang, M., Zhang, H.: Tag ranking. In: The 18th International Conference on World Wide Web (WWW 2009), pp. 351–360. ACM Press, Marid (2009)
13. Schenkel, R., Crecelius, T., Kacimi, M., Michel, S., Neumann, T., Parreira, J.X., Weikum, G.: Efficient top-k querying over social-tagging networks. In: The 31th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2008), pp. 523–530. ACM Press, Singapore (2008)
14. Pedro, J.S., Siersdorfer, S.: Ranking and classifying attractiveness of photos in folksonomies. In: The 18th International Conference on World Wide Web (WWW 2009), pp. 771–780. ACM Press, Marid (2009)
15. Noll, M.G., Meinel, C.: Exploring social annotations for web document classification. In: The 2008 ACM Symposium on Applied Computing (SAC 2008), pp. 2315–2320. ACM Press, Brazil (2008)
16. Shepitsen, A., Gemmell, J., Mobasher, B., Burke, R.D.: Personalized recommendation in social tagging systems using hierarchical clustering. In: The 2008 ACM Conference on Recommender Systems, pp. 259–266. ACM Press, Lausanne (2008)
17. Gemmell, J., Shepitsen, A., Mobasher, B.: Personalization in Folksonomies Based on Tag Clustering. In: The AAAI 2008 Workshop on Intelligent Techniques for Web Personalization and Recommender Systems, Chicago, pp. 37–48 (2008)
18. Begelman, G., Keller, P.: Automated Tag Clustering: Improving Search and Exploration in the Tag Space. In: The 15th International Conference on World Wide Web (WWW 2006), Workshop on Collaborative Web Tagging, Edinburgh, UK (2006)
19. Ramage, D., Heymann, P.: Clustering the Tagged Web. In: The Second ACM International Conference on Web Search and Data Mining, pp. 54–63. ACM Press, Barcelona (2009)
20. Salton, G., Buckley, C.: Term-Weighting Approaches in Automatic Text Retrieval. *Information Processing and Management: an International Journal* 24, 513–523 (1988)
21. Yeung, K.Y., Ruzzo, W.L.: Principal component analysis for clustering gene expression data. *Bioinformatics* 17, 763–774 (2001)
22. Manning, C., Raghavan, P., Schütze, H.: Introduction to information retrieval. Cambridge University Press, Cambridge (2008)

A Generalization Based Approach for Anonymizing Weighted Social Network Graphs^{*}

Xiangyu Liu and Xiaochun Yang

College of Information Science and Engineering,
Northeastern University, Shenyang 110819, China
yangxc@mail.neu.edu.cn

Abstract. The increasing popularity of social networks, such as online communities and telecommunication systems, has generated interesting knowledge discovery and data mining problems. Since social networks usually contain personal information of individuals, preserving privacy in the release of social network data becomes an important concern. An adversary can use many types of background knowledge to conduct an attack, such as topological structure and/or basic graph properties. Unfortunately, most of the previous studies on privacy preservation can deal with simple graphs only, and cannot be applied to weighted graphs. Since there exists numerous unique weight-based information in weighted graphs that can be used to attack a victim's privacy, to resist such weight-based re-identification attacks becomes a great challenge. In this paper, we investigate the identity disclosure problem in weighted graphs. We propose k -possible anonymity to protect against weight-based attacks and develop a generalization based anonymization approach (named GA) to achieve k -possible anonymity for a weighted graph. Extensive experiments on real datasets show that the algorithm performs well in terms of protection it provides, and properties of the original weighed network can be recovered with relatively little bias through aggregation on a small number of sampled graphs.

1 Introduction

Social network applications, such as online communities and telecommunication systems, have become popular for information sharing. Exploring the properties of social networks has generated interesting knowledge discovery and data mining problems. However, social networks usually contain individuals' sensitive information. Preserving privacy in the release of social network data becomes an important concern.

One fundamental privacy issue in publishing social network data is identity disclosure problem. Most of existing studies on privacy preservation deal

^{*} The work is partially supported by the National Natural Science Foundation of China (Nos. 60973018, 60973020) and the Fundamental Research Funds for the Central Universities (Nos. N090504004, N100604013, N100704001, N090104001).

with simple graphs(i.e. undirected, unweighted and loopless graphs), which consider graph structure as adversaries’ background knowledge. However, weighted graphs are more descriptive and common in real world than simple graphs. Applying previous privacy preservation techniques to weighted graphs cannot protect privacy effectively.

In weighted graphs, there exists numerous weight related information that can be used to attack a victim’s privacy. [1] studies how the weights on the connections of a target vertex with other vertices in weighted graph results in identity disclosure problem. For a vertex v , the sequence of weights on all edges connecting v with other vertices sorted in descending order is defined as v ’s weight bag, denoted w_v . Fig.1(a) shows a weighted undirect graph which contains 4 vertices. The weight bag for vertex A is $w_A = [w_{AB}, w_{AD}] = [2, 1]$ and the weight bag set for G is $W = \{[2, 1], [3, 2, 1], [3, 1], [1, 1, 1]\}$. Without considering edge weight, anonymized graph G' in Fig.1(b) can resist with previous proposed privacy attacks using graph topological structure as background knowledge, including degree[3], neighborhood[4], subgraph[5], etc. Adversary cannot identify any vertex in G' with confidence larger than 50%. However, when an adversary knows A ’s weight bag $w_A = [2, 1]$, the adversary can easily identify vertex 1 in G' as A , since only vertex 1 has the same weight bag as A , which causes an identity disclosure problem. In real social network NetSci [15], 6.48% of vertices could be uniquely identified based on weight bag information [1].

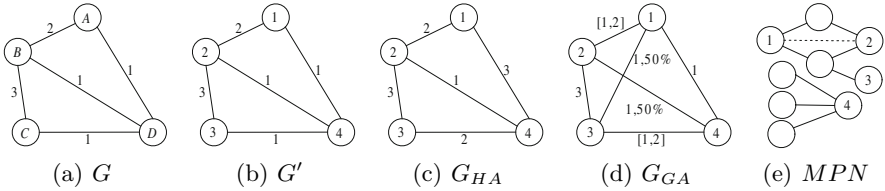


Fig. 1. Anonymization of weighted graph

1.1 Motivation

[1] proposes histogram anonymization(HA) to prevent from weight-based attacking. G_{HA} in Fig.1(c) shows an anonymized graph of G using histogram anonymization. After histogram anonymizing graph G into G_{HA} , for $\forall w \in W_{HA}$, there are at least $k - 1$ other weight bags that are the same with w and graph G_{HA} is defined as k -histogram anonymous. The weight bag set of G_{HA} is $W_{HA} = \{[3, 2, 1], [3, 2, 1], [3, 2], [3, 2]\}$, thus G_{HA} is 2-histogram anonymous and the vertices cannot be identified with confidence larger than 50% when weight bags are considered as the adversary’s background knowledge. However, modifying edges and weights in histogram anonymization reduces the statistical utility of the anonymized graph. Considering graph query Q :

$$Q: \text{SELECT COUNT}(\forall edge \in G) \text{ WHERE } edge.weight \geq 2$$

The result of Q is 4 on graph G_{HA} , which is a little biased to the real answer 2 on G . Since there are a large amount of vertices and edges in weighted graphs, modifying edges and weights in histogram anonymization would cause estimates of a graph property to be systematically biased or highly variable, which will be introduced in the experimental section. Privacy preserving can also be achieved through increasing uncertainty of anonymized graph. Considering G_{GA} in Fig. 1(d), $w_{12} = [1, 2]$ means w_{12} would be any value in the interval of $[1, 2]$, and "50%" on edge e_{13} denotes that e_{13} exists with the probability of 50%, thus w_1 in G_{GA} can be formalized as $[[1, 2], 100\%), (1, 50\%), (1, 100\%)]$. The uncertainty in G_{GA} is achieved through generalization, which is commonly adopted in privacy preserving techniques. If G_{GA} is an anonymized version of G in Fig. 1(a), vertex 1 may correspond to A , since w_A is a possible instance of w_1 . We use *candidate set* in [10] to represent the vertices of G_{GA} that could feasibly correspond to *target* x , denoted $\text{cand}(x)$. In graph G_{GA} , we have $\text{cand}(A) = \text{cand}(D) = \{1, 4\}$, $\text{cand}(B) = \text{cand}(C) = \{2, 3\}$, thus an adversary cannot identify vertex in G_{GA} with confidence larger than 50%, which is the same as G_{HA} . Let's consider query Q on G_{GA} . In G_{GA} , edge e_{23} definitely satisfies Q , edges e_{12} and e_{34} possibly satisfy Q , and the other edges cannot satisfy Q , thus the result of Q on G_{GA} is an interval $[1, 3]$, which encloses the real answer 2 of Q . Generalization based graph anonymization can protect privacy in weighted graphs while preserving statistical properties and utilities.

1.2 Challenges

Although privacy preservation in social network has been studied extensively and several important models such as K -Degree [3], K -Isomorphism [5], K -Symmetry [7] and K -Automorphism [6] as well as many efficient algorithms have been proposed, most of the previous studies can deal with privacy preservation on simple graphs only. As example in Fig. 1 shows, applying those methods to weighted graphs straightforwardly still causes privacy leakage.

In practice, we need to anonymize weighted social network before it is released. However, anonymizing weighted social network graphs is much more challenging than anonymizing simple graphs.

First, it is much more challenging to design privacy preserving model for weighted graphs. As shown in Fig. 1, although anonymizing different weight bags into the same ones through modifying edges and weights can prevent from weight based attacks, statistical properties and utilities of weighted graphs are reduced.

Second, it is much more challenging to measure the information loss in anonymizing weighted graphs. Besides the information loss in each edge weight, the information loss of weight related statistical properties and utilities should also be measured. Anonymizing weighted graphs would affect the topological structures of the graphs which should be considered. Thus, there exists different ways to define the information loss and data utility for weighted graphs. Instead of measuring those information loss separately, all types of information loss should be measured in combination.

Last but not least, it is much more challenging to devise anonymization methods for weighted social networks than simple ones. One simple observation is that when the weight of an edge is modified or generalized, this operation would affect the two vertices that are connected with this edge. When devising anonymization approach for weighted social networks, both information loss on weights and graph properties should be as less as possible.

Our contributions can be summarized as follows. (1) We propose k -possible anonymity model to protect against weight-based attacks. (2) We prove that to achieve optimal k -possible anonymity is NP-hard. (3) We develop a generalization based anonymization approach to achieve k -possible anonymity, and design a number of techniques to make the anonymization process efficient while maintaining the utility. (4) Our extensive empirical studies show that our method performs well in anonymizing weighted graphs of real world.

The remainder of the paper is organized as follows. Related work are summarized in Section 2. In Section 3, we give the problem definition and formalize the k -possible anonymity model. The generalization based anonymization approach that ensures k -possible anonymity is investigated in Section 4. We evaluate our method in Section 5. Section 6 concludes the paper.

2 Related Work

There has been much research on privacy preserving in social network graphs. Backstrom et al. [2] firstly proposed the privacy problems in social network.

Most of existing research has focused on *graph anonymization*, which considers the topological structures of graph as adversary's background knowledge. One popular graph anonymization is to add and/or delete edges for modifying the topological structure of the graph in order to prevent re-identification in the anonymized graph. Liu et al. [3] studies graph anonymizing to prevent from privacy attacking using degree as background knowledge. Zhou et al. [4] provides identity privacy through anonymizing each vertex's neighborhood subgraph. In order to prevent privacy attacks using subgraph as background knowledge, Cheng et al. [5], Zou et al. [6] and Wu et al. [7] propose privacy models K -Isomorphism, K -Automorphism and K -Symmetry respectively for graph anonymization. Ying et al. [8] studies graph randomization while preserving the spectrum of the graph. Yuan et al. [9] gives a solution to satisfy different needs on privacy protecting level. Another main design of existing graph anonymization is based on clustering vertices into super vertices, which makes the vertices in a super vertex indistinguishable from each other. Based on vertices clustering, Hay et al. [10] and Campan et al. [11] study vertex anonymization to prevent re-identification. Zheleva et al. [12] formulate the problem of preventing sensitive edge disclosure in unweighted graph.

Although most of existing research considers anonymization problems for simple unweighted graphs, recent work [11,13,14] has focused on privacy preservation in weighted graphs. Das et al. [13] aims at preserving the linear properties of the graph while modifying edge weights. [14] perturbs edge weights within a

threshold that follows a probability distribution while retaining the cost of the shortest path. However, the threshold used in this approach is always small and the anonymized weights are close to the original ones, there still exists the possibility of privacy leaking. [1] considers re-identification in weighted graphs and proposes histogram anonymization to protect vertex privacy through modifying edge weights and vertex connections, which causes a great information loss on the utilities and statistical properties of weighted graphs.

3 Problem Definition

In this paper, we model a *social network* as an uncertain weighted graph $G = (V, E, W, P)$, where V is a set of vertices, $E \subseteq V \times V$ is a set of edges, W is a set of weights between vertices in V , and $w_{uv} = w_{e_{uv}}$ denotes the weight of edge e_{uv} . P_e denotes the probability that edge e exists. For $\forall e \in E$, P_e is initialized with 100%.

Definition 1. (*edge set*) For each vertex $v \in V$, the edge set for v is defined as $e_v = \bigcup_{e_{vu} \in E} \{e_{vu}\}$.

For instance, in Fig. 1(d), the edge set for vertex 1 is $e_1 = \{e_{12}, e_{13}, e_{14}\}$.

Definition 2. (*possible candidate*) Graph G is anonymized into G_a through generalization, if weight bag $w_v(v \in G)$ is a possible instance of $w_u(u \in G_a)$, u is defined as a possible candidate of v , denoted $u \succeq v$.

In Fig. 1, for vertices $1 \in G_a$ and $A \in G$, $w_1 = [(1, 2], 100\%), (1, 50\%), (1, 100\%)]$ and $w_A = [2, 1]$, w_A is a possible instance of w_1 , thus vertex $1 \succeq A$.

Definition 3. (*k-possible anonymous*) Graph G is anonymized into G_a through generalization, G_a is k -possible anonymous if there exist vertex groups g_1, \dots, g_m of G and g'_1, \dots, g'_m of G_a that satisfy $\bigcup_{i=1}^m g_i = V, \bigcup_{i=1}^m g'_i = V_a, g_i \cap g_j = g'_i \cap g'_j = \emptyset (i \neq j), |g_i| = |g'_i| \geq k (i \in [1, m])$ and $u \succeq v (\forall u \in g'_i, \forall v \in g_i)$, where $g'_i (i \in [1, m])$ is defined as anonymization group.

In Fig. 1, considering vertex groups $\{A, D\}$ and $\{B, C\}$ of G , $\{1, 4\}$ and $\{2, 3\}$ of G_a , since $1 \succeq A, 1 \succeq D, 4 \succeq A, 4 \succeq D, 2 \succeq B, 2 \succeq C, 3 \succeq B$ and $3 \succeq C$, G_a in Fig. 1(d) is 2-possible anonymous.

Vertices in the same anonymization group g are indistinguishable from each other based on weight bags. Thus, if an anonymized graph G_a is k -possible anonymous, the adversary cannot re-identify each vertex $v \in G_a$ with confidence larger than $\frac{1}{k}$.

Problem 1. (*graph anonymization*) Given a weighted graph $G(V, E, W, P)$, and an integer k , find generalizations that anonymize G into a k -possible anonymous graph G_a , such that information loss $L(G, G_a)$ is minimized.

The information loss incurred by generalizations is formalized as Equation 1.

$$L(G, G_a) = \sum_{e \in E_a} |maxW_e - minW_e| \quad (1)$$

where $[minW_e, maxW_e]$ denotes the weight interval assigned to edge e through generalization. Equation 1 is simple and clear to capture the information loss and uncertainties of the edge weights resulted from generalization.

Theorem 1. (Complexity:) *Achieve k -possible anonymity with minimal information loss is NP-hard.*

Proof. The proof is constructed by reducing the NP-hard optimal k -anonymity problem to k -possible anonymity with minimal information loss through mapping weight bag $\mathbf{w}_v (v \in V)$ into d_{max} -dimensional space, where d_{max} is the maximum degree of vertices in V . Due to the space limitation, we omit the details here.

4 Generalization Based Anonymization

In this section, we introduce an efficient heuristic algorithm as a solution to Problem 1. We achieve k -possible anonymity through generalization based anonymization. We adopt a two-step strategy in our graph anonymization method. It first try to generate anonymization groups with minimal information loss L , and then k -possible anonymity is achieved through edge generalization.

4.1 Generating Anonymization Groups

The first challenge is to generate anonymization groups for privacy preserving while retaining high data utility. We first introduce an efficient clustering-based grouping technique in Algorithm 1.

For each vertex v_i , we map its weight bag $\mathbf{w}_i = [w_{i1}, w_{i2}, \dots, w_{id_i}] (w_{i1} \geq w_{i2} \geq \dots \geq w_{id_i})$ into a point in m -dimension space, where d_i is the degree of v_i and $m = \max(d_i), i \in [1, n]$. For vertex v_i with degree $d_i < m$, corresponding weight bag \mathbf{w}_i is expanded by filling $(m - d_i)$ zeros.

In Algorithm 1, Line 1 initializes the number of anonymization groups based on input k value, and then the centers of the groups are randomly picked in Line 3. Line 6 adds the closest k points to each group based on the Euclidean distances between points and group centers. After all points are added to anonymization groups, the center of each group is updated in Line 10. Here we adopt the mean of the points in each group as new center. Line 11 shows that Lines 5-10 are iteratively performed until the distance between old and new center of each group is smaller than predefined threshold ϵ . In our experiments, we adopt $\epsilon = 0$, which means clustering and center updating are performed iteratively until points in each group do not change. Although we set $\epsilon = 0$, Algorithm 1 still has an efficient convergence performance which is shown in our experiments.

Algorithm 1. Generate Anonymization Groups

Input: A weight bag set W and an integer k
Output: Anonymization group set AG

- 1 $N = \lfloor |W|/k \rfloor$;
- 2 $I=1$;
- 3 Randomly pick N points as group centers $\overline{AG}_j(I)$, $j \in [1, N]$;
- 4 **repeat**
- 5 **for** $j = 1$ to N **do**
- 6 Find the closest k points to $\overline{AG}_j(I)$ in W and add to AG_j ;
- 7 $W = W - AG_j$;
- 8 Assign each remaining point in W to the closest group;
- 9 $I = I + 1$;
- 10 Update $\overline{AG}_j(I)$, $j \in [1, N]$;
- 11 **until** $\|\overline{AG}_j(I) - \overline{AG}_j(I-1)\| \leq \epsilon, j \in [1, N]$;
- 12 **return** $AG = \{AG_1, \dots, AG_N\}$;

The computational complexity of Algorithm 1 is $O(t\frac{n^2}{2k})$, where n is the number of points, k is the size of anonymization group, and t is the number of iterations. Each iteration introduces $O(\frac{n^2}{2k})$ operations to calculate distances between points and group centers.

4.2 Edge Generalization

Edge generalization is the second stage of the two-step strategy, which aims to achieve k -possible anonymity through generalizing edges based on generated anonymization groups. We provide *Edge Generalization*(EG) method in Algorithm 2.

We take G' in Fig 1(b) as an example to demonstrate EG algorithm. For vertices in Fig 1(b), $AG = \{AG_1 = \{1, 4\}, AG_2 = \{2, 3\}\}$ is generated when $k = 2$ using Algorithm 1. In order to let each vertex in AG_i be possible candidates for all other vertices in AG_i after anonymization, edges of each vertex should be generalized. Firstly, Lines 4-5 generalize existing edges. $e_v(j)$ denotes the edge in e_v that has the j -th weight $w_v(j)$ of w_v , where d_v is the degree of v and obviously $w_v(j) = w_{e_v(j)}$. $Min_{i,j}$ denotes $\min_{u \in AG_i}(w_u(j))$, which is the minimum value of all the j -th weights of the weight bags in AG_i , and $Max_{i,j}$ denotes $\max_{u \in AG_i}(w_u(j))$ analogously. For example, weight bags of vertices in AG_2 are $w_2 = [w_{e_{23}}, w_{e_{21}}, w_{e_{24}}] = [3, 2, 1]$ and $w_3 = [w_{e_{32}}, w_{e_{34}}] = [3, 1]$, thus $Max_{2,2} = 2$ and $Min_{2,2} = 1$, respectively. Line 4 generalizes $w_{e_{21}}$ and $w_{e_{34}}$ into $[1, 2]$ as shown in Fig 1(d). Since edge weight w_{uv} occurs in both w_u and w_v , w_{uv} would be generalized twice with different $[Min, Max]$ and weight interval of w_{uv} is continuously extended. For instance, generalize $w_{uv} = [1, 3]$ with $[2, 4]$ would extend w_{uv} into $[1, 4]$. Existence probabilities are updated in Line 5, which aims to preserve privacy and statistical properties, where $P_{i,j} = \frac{\text{count}_{u \in AG_i}(d_u \geq j)}{|AG_i|}$. Take $e_2(3) = e_{24}$ for example, Line 5 updates $P_{e_{24}}$ with $\min\{P_{2,3}, P_{e_{24}}\} = \min\{\frac{\text{count}_{u \in AG_2}(d_u \geq 3)}{|AG_2|}, P_{e_{24}}\} = \min\{\frac{1}{2}, 100\%\} = 50\%$.

Algorithm 2. Edge Generalization

Input: Weighted graph $G = (V, E, W, P)$,
anonymization group set $AG = \{AG_1, \dots, AG_N\}$

Output: k -possible anonymous G_a

```

1 for  $i = 1$  to  $N$  do
2   for each  $v \in AG_i$  do
3     for  $j = 1$  to  $d_v$  do           /* Generalize existing edges */
4       Generalize  $w_{e_v(j)}$  with  $[Min_{i,j}, Max_{i,j}]$ ;
5        $P_{e_v(j)} = \min\{P_{i,j}, P_{e_v(j)}\}$ ;
6     for  $t = (d_v + 1)$  to  $maxDegree(v)$  do       /* Add new neighbors */
7        $cand = null; pos = null; Cost_{cand, pos} = +\infty$ ;
8       for each  $u \in V$  with  $e_{uv} \notin E$  and  $u$  needs new neighbor do
9         for  $m = (d_u + 1)$  to  $maxDegree(u)$  do
10          if  $Cost_{u,m} < Cost_{cand, pos}$  then
11             $cand = u; pos = m$ ;
12        if  $cand == null$  then
13          Find the most potential neighbor  $u \in V$  of  $v$ ;
14           $cand = u; pos = d_u + 1$ ;
15        /* Create an edge between  $v$  and  $cand$ , where  $cand \in AG_c$  */
16        Add new edge  $e_{v,cand}$  to  $E$ ;
17         $e_v(t) = e_{cand}(pos) = e_{v,cand}$ ;
18        Generalize  $w_{e_{v,cand}}$  with  $[Min_{i,t}, Max_{i,t}]$  and  $[Min_{c,pos}, Max_{c,pos}]$ ;
19         $P_{e_{v,cand}} = \min\{P_{i,t}, P_{c,pos}\}$ ;
19 return  $G$ ;

```

After existing edge generalization, w_2 is generalized into $[(3, 100\%), ([1, 2], 50\%), (1, 50\%)]$, which makes vertex 2 be a possible candidate of vertex 3.

Generalizing existing edges only cannot achieve k -possible anonymity, some vertices need to be connected with new neighbors. For instance, w_3 is generalized into $[(3, 100\%), ([1, 2], 100\%)]$ after existing edges generalization, of which w_2 is not a possible instance. In order to be possible candidate of vertex 2, vertex 3 needs to add a new neighbor. Function $maxDegree(v)$ in Line 6 returns the max degree of vertices in AG_i that v belongs to, and v needs $(maxDegree(v) - d_v)$ new neighbors in order to be possible candidate for vertices in AG_i . For vertex v that needs a neighbor, Lines 7-14 find a neighbor candidate $cand$ and Lines 15-18 add an edge connection between v and $cand$. When we choose a neighbor candidate $cand$ for v , vertices that also need new neighbors are considered with priority in Lines 8-11. Vertex u that causes the lowest information loss on e_{vu} , which is to be added to E , is picked as neighbor candidate $cand$. For instance, when we choose neighbor candidate for vertex 3, vertex 1 in AG_1 which also needs a new neighbor is picked as neighbor candidate of vertex 3. If Lines 8-11 did not find neighbor candidate, Lines 12-14 choose the most potential neighbor (MPN) of v to be neighbor candidate $cand$. We define the vertex u that has the most common neighbors with v and $e_{uv} \notin E$ as v 's most potential neighbor. In Fig. 1(e), vertex

2 is the most potential neighbor of vertex 1. When new edge $e_{v,cand}$ is created, Line 17 generalizes $w_{e_{v,cand}}$ with both weight intervals of v and $cand$ in order to meet k -possible anonymity requirement. If $cand$ is v 's most potential neighbor, $[Min_{c,pos}, Max_{c,pos}]$ in Line 17 and $P_{c,pos}$ in Line 18 should be neglected.

The computational complexity of Algorithm 2 is $O(mn)$, where n is the number of vertices in V and m is the number of vertices that need to add new neighbors. For each vertex v that needs to add new neighbor, there are maximum n vertices to be v 's neighbor candidates. Therefore, the total complexity is $O(mn)$. Although the worst case for Algorithm 2 is $O(n^2)$ when $m = n$, m is much less than n in real datasets as shown in our experiments.

5 Experiments

In this section, we evaluate our methods using two real data sets, **NetSci** and **Hep-Th**. All these graphs are weighted and undirected. **NetSci** describes a coauthorship network of scientists working on network theory and experiment. There are 1,589 scientists(vertices) and 2742 edges in this data graph, and each edge is assigned with a real weight as described in [15]. The **Hep-Th** database [16] extracts weighted network of coauthorships between scientists posting preprints on the High-Energy Theory E-Print Archive between Jan 1, 1995 and December 31, 1999, which contains 8361 vertices and 15751 edges.

All the programs are implemented in Java. The experiments are performed on a 2.33GHz Intel Core 2 Duo CPU with 4GB DRAM running the Windows XP operating system.

We focus on the utility and statistical properties of the anonymized graphs. Here we use some weight related measurements to evaluate the utilities of the anonymized weighted graphs. (1)*Edge Weight* describes the distribution of all edge weights in the graph. (2)*Degree* is the distribution of the degrees of all vertices in the graph. (3)*Volume* is a distribution of the volumes of vertices in the network, where volume is the sum of a weight bag. (4)*Path length* is a distribution of the lengths of the shortest paths between 1000 randomly sampled pairs of vertices in the network.

We measured these characteristics for the original graph G , the graph G_{HA} anonymized through HA [1] and a set of 20 output graphs sampled from the graph G_{GA} which is anonymized through our proposed GA approach. For each edge e in G_{GA} , we sampled w_e uniformly from the generalized weight interval of e , and whether e exists in the sampled graph depends on the existence probability of e .

5.1 Runtime

Fig 2 shows the runtime of HA and GA on the two data sets with different k values. Our results show that GA requires less runtime than HA. The runtime of HA increases with the increment of k value, while the runtime of GA keeps stable and even decreases with the increment of k value. After anonymizing weight set, HA constructs the graph based on the anonymized weight set, while

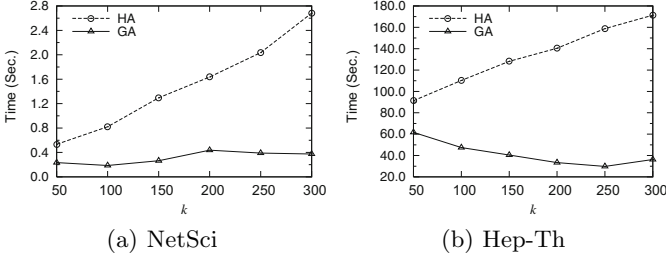


Fig. 2. Runtime for different k values

GA anonymizes on the original graph, thus HA costs more runtime than GA. In Fig 2(b), the runtime of GA decreases when k increases from 50 to 250, and increases when k increases from 250 to 300. This is for the reason that the size of anonymization group depends on k value, and different size of anonymization group would affect the performance of the clustering in generating anonymization groups. Anonymization group size which is suitable to the properties of the data set would result in efficient clustering.

5.2 Data Utilities

In this work, data utilities refer to statistical properties of the graphs. Figs.3 to 6 show the results of the experiments with respect to the four statistical characteristics for both data sets, where Original, GA, and HA denote the distributions for original graph, generalization based anonymization and histogram anonymization [1], respectively.

Fig.4 shows the distributions of degrees when $k = 20, 100$ for both data sets. When $k = 20$, Fig 4(a) and 4(c) show that the distributions of degree for GA are very similar to the original graph, and the distributions for HA are a little biased to the original graph. In Fig 4(b) and 4(d), when k increases to 100, the curves for GA and Original are still aligned, while the curves for HA are quite different. Similar results could be observed in Figs.3, 5 and 6.

Our results in Figs.3 to 6 demonstrate that GA protects privacy in weighted graphs while guaranteeing the accuracy of statistical analysis. When k value increases, GA still preserves the essential graph information, while the bias of the distributions for HA increases. This is for the reason that HA preserves privacy through modifying weights and edges, which results in more information loss when k increases. GA preserves privacy through anonymizing graphs using generalization, which preserves the properties and statistical utilities of original graph. Thus, the network measures of original graph could be recovered with little bias through aggregation on a small number of sampled graphs.

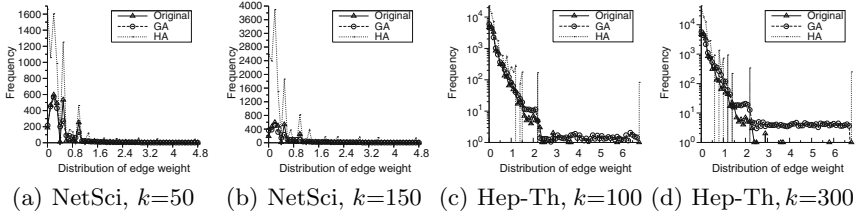


Fig. 3. Distributions of edge weights

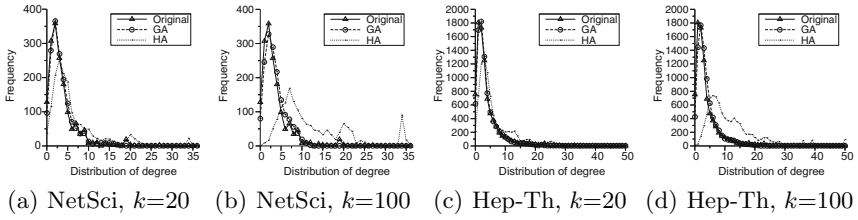


Fig. 4. Distributions of degrees

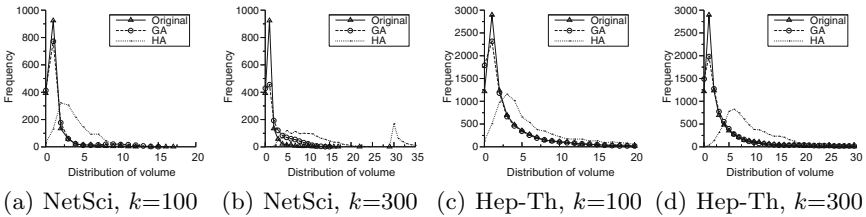


Fig. 5. Distributions of volumes

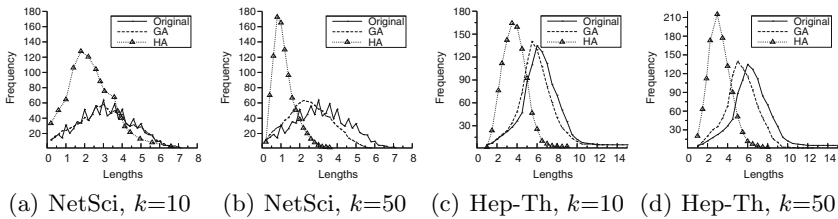


Fig. 6. Distributions of path lengths

6 Conclusion

Most of existing research work studies privacy preserving for simple graphs only, and cannot be applied to weighted graphs. In this paper, we focus on protecting privacy in weighted graphs. We propose a privacy preserving model, named

k -possible anonymity, to prevent from identity disclosure against weight-based attacks. We develop a generalization based anonymization approach (named GA) to achieve k -possible anonymity for a weighted graph, and design a number of techniques to make the anonymization process efficient while maintaining the utility. Extensive experiments on real datasets show that our method performs well in terms of protection it provides, and a wide range of weight related graph analysis can be performed accurately.

References

1. Li, Y., Shen, H.: Anonymizing graphs against weight-based attacks. In: International Conference on Data Mining Workshops (ICDMW 2010), pp. 491–498 (2010)
2. Backstrom, L., Dwork, C., Kleinberg, J.: Wherefore art thou R3579X?: Anonymized social networks, hidden patterns and structural steganography. In: International World Wide Web Conference (WWW 2007), pp. 181–190 (2007)
3. Liu, K., Terzi, E.: Towards identity anonymization on graphs. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data (SIGMOD 2008), pp. 93–106. ACM Press, New York (2008)
4. Zhou, B., Pei, J.: Preserving privacy in social networks against neighborhood attacks. In: Proceedings of the 24th IEEE International Conference on Data Engineering (ICDE 2008), pp. 506–515 (2008)
5. Cheng, J., Fu, A.W.-C., Liu, J.: K-Isomorphism: Privacy preserving network publication against structural attacks. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data (SIGMOD 2010), pp. 459–470 (2010)
6. Zou, L., Chen, L., Ozsu, M.T.: K-automorphism: A general framework for privacy preserving network publication. In: Proceedings of the 35th International Conference on Very Large Databases (VLDB 2009), vol. 2(1), pp. 946–957 (2009)
7. Wu, W., Xiao, Y., Wang, W., He, Z., Wang, Z.: K-Symmetry model for identity anonymization in social networks. In: Proceedings of the 13th International Conference on Extending Database Technology (EDBT 2010), pp. 111–122 (2010)
8. Ying, X., Wu, X.: Randomizing social networks: a spectrum preserving approach. In: Proceedings of the 2008 SIAM International Conference on Data Mining (SDM 2008), pp. 739–750 (2008)
9. Yuan, M., Chen, L., Yu, P.S.: Personalized privacy protection in social networks. Proceedings of the VLDB Endowment 4(2), 141–150 (2010)
10. Hay, M., Miklau, G., Jensen, D., Towsley, D.: Resisting structural re-identification in anonymized social networks. In: Proceedings of the 34th International Conference on Very Large Databases (VLDB 2008), pp. 102–114. ACM, New York (2008)
11. Campan, A., Truta, T.M.: A clustering approach for data and structural anonymity in social networks. In: Bonchi, F., Ferrari, E., Jiang, W., Malin, B. (eds.) PinKDD 2008. LNCS, vol. 5456, pp. 33–54. Springer, Heidelberg (2009)
12. Zheleva, E., Getoor, L.: Preserving the privacy of sensitive relationships in graph data. In: Bonchi, F., Malin, B., Saygin, Y. (eds.) PinKDD 2007. LNCS, vol. 4890, pp. 153–171. Springer, Heidelberg (2008)

13. Das, S., Egecioglu, O., Abbadi, A.E.: Anonymizing weighted social network graphs. In: The 26th International Conference on Data Engineering, ICDE 2010 (2010)
14. Liu, L., Wang, J., Liu, J., Zhang, J.: Privacy preservation in social networks with sensitive edge weights. In: 2009 SIAM International Conference on Data Mining(SDM 2009), Sparks, Nevada, pp. 954–965 (April 2009)
15. Newman, M.E.J.: Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 0605087 (2006)
16. Newman, M.E.J.: The structure of scientific collaboration networks. *Proc. Natl. Acad. Sci. USA* 98, 404–409 (2001)

Incremental Reasoning over Multiple Ontologies

Jing Lu¹, Xingzhi Sun², Linhao Xu², and Haofen Wang¹

¹ Shanghai Jiaotong University, China
{robert_lu,whfcarter}@apex.sjtu.edu.cn

² IBM China Research Lab, China
{sunxingz,xulinhao}@cn.ibm.com

Abstract. Semantic web data management on top of relational database has been regarded as a promising solution for scalable ontology storing, querying and reasoning. In order to reduce query response time, inferred results often need to be pre-computed and materialized. However, this approach faces the great challenge when data is updated, since the repositories need to perform reasoning from scratch to guarantee the consistency of the inferred results. This paper proposes a novel solution to enable the incremental data update on the context that ontology reasoning and user-defined rule reasoning are performed over multiple ontologies. We propose an effective data organization method that uniformly organizes both original ontologies and inferred results for ontology and user-defined rule reasoning, with the support of named graph. Inspired by the Rete algorithm, we design an inference network to link the ontology data, inferred results, and intermediate inferred results. As a consequence, when the ontology data update, changes can be effectively propagated in the network based on their named graphs. We implement the proposed approach on our previous ontology store and the experimental results show that our solution can significantly improve the reasoning performance when ontology data update happens.

1 Introduction

In recent years, the amount of Semantic Web data has been increasing rapidly (e.g., in Linking Open Data (LOD) project [2]). On the other side, more and more *ontologies* have been created, which provides *schema* or *meta-data* for Semantic Web data. With these ontologies, Semantic Web data can be interpreted and processed by computers, and the implicit information can be derived automatically by ontology reasoning.

Recently many systems have been developed to support storage, query and reasoning on ontologies. The majority of ontology repositories [6,12,13] are built on top of relational database (RDB) so that the mature database technology can be applied for scalable semantic web data management. The RDB-based ontology repository stores the semantic web data in a tailored schema and thus SPARQL queries [9] are eventually translated into SQLs for data retrieval. Given a query that involves reasoning on ontology data, the reasoning can be preformed in two different ways. For runtime approaches, the reasoning task is performed

at query time and only on data that is relevant to the query. On the other hand, a materialization-based reasoning will pre-compute all inferred results on all ontology data and materialize them permanently. As such, answering a query with reasoning becomes as simple and efficient as data retrieval. This paper focuses on the reasoning problem on the materialization-based ontology repository.

The main challenge for materialization-based ontology repository comes from data update. To maintain the consistency of the inferred results, traditional materialization-based ontology repositories [6,13] have to re-perform the reasoning from the scratch once data update occurs. In order to reduce the reasoning cost, a natural thought is to enable an incremental reasoning. Further supporting incremental reasoning on the selected combinations of multiple ontologies requires to efficiently identify which datasets and rules will be triggered for incremental computation when ontology data is updated.

In this paper, we propose a novel solution for effectively supporting the incremental reasoning over multiple ontologies in the materialization-based ontology repository. The reasoning scope covers (1) the ontology reasoning performed based on Description Logic Programming (DLP) [5], and (2) the reasoning for user-defined rules. In general, we treat all inferred results as a special named graph whose URI is maintained by the ontology system, so that both original ontology data and inferred data can be organized uniformly. By maintaining a mapping between the original graphs and the system-generated graphs, SPARQL queries can be rewritten into SQLs that will retrieve results from both original graphs and inferred graphs. Furthermore, inspired by the classical memory-based Rete algorithm [4], we design and implement a novel inference network on top of the relational database to enable the incremental reasoning. The inference network organizes the original data, inferred data and inferred intermediate data in a network based on a set of Datalog rules. Any data update on the ontology will be propagated within the network in a batch-processing mode until the data in the network reach a stable state.

The rest of this paper is organized as follows. Section 2 formulates the problem. Our approach is presented in Section 3. Section 4 shows the experiment results. Section 5 discusses the related work. We conclude the paper in Section 6.

2 Problem Description

Semantic Web Data. An RDF graph is defined as a set of triples in the form of $\langle s, p, o \rangle$, where s is a subject, p is a predicate and o is an object. A *named graph* is an RDF graph with a graph identifier. Formally, a named graph is defined as a pair $ng = (uri_{ng}, G_{ng})$, where uri_{ng} is a URI of ng and G_{ng} is a set of RDF triples. For example, we can define a named graph $ng_1 = (\text{http://example.org/g1}, \{\langle \text{Tom}, \text{hasSister}, \text{Jane} \rangle\})$ where G_{ng_1} contains only $\langle \text{Tom}, \text{hasSister}, \text{Jane} \rangle$ and uri_{ng_1} is $\text{http://example.org/g1}$.

Simply, an ontology can be regarded as the combination of meta-data (i.e., the definition of class, property, and their relationships) and data (assertions for individuals), both of which can be represented as RDF triples. In ontology

reasoning, by applying the formal semantics of ontology language, the implicit triples can be inferred from the data and the meta-data.

Rule. Let a *term* t be an RDF resource or a variable. A *formula* l can be formed by applying either a unary predicate or a binary predicate on term(s), that is, l is in the form of either $p(t)$ or $p(t_1, t_2)$, where p indicates a predicate. Thus, a *Datalog rule* r is in the form of $h:-b_1 \wedge \dots \wedge b_n$, where h and b_1, \dots, b_n are formulas that could be either $p(t)$ or $p(t_1, t_2)$. h is the consequence of the rule, also called the *rule head*, and $b_1 \wedge \dots \wedge b_n$ is the *rule body*, giving the conditions of the rule.

Let R_{onto} be the set of Description Logic Program rules [5], i.e., a fixed, intrinsic set of rules for ontology reasoning. For any rule r in R_{onto} , $r.head$ and $l \in r.body$ are corresponding to the predicates which are defined in the ontology. For example, if in an ontology, a predicate p is defined as *transitive*, we have the corresponding ontology rule $p(x, y) :- p(x, z), p(z, y)$. On the other side, let R_{user} be the set of user-defined rules, i.e., the rules are defined on top of the ontology by users. For any rule r in R_{user} , $r.head$ must be a *non-ontology* predicate and $l \in r.body$ could be either an ontology predicate or a user-defined rule predicate. Intuitively, it means that the user-defined rules cannot change the semantics of an ontology. To comply with the RDF syntax, we also put a constraint on the formulas appearing in R_{user} whose predicates must be either unary or binary. For example, given the ontology predicates `hasFather` and `hasBrother`, user can define new binary predicate `hasUncle` via the rule `hasUncle(x, z) :- hasFather(x, y) ^ hasBrother(y, z)`.

Query. The reasoning can derive all implicit data in form of RDF triples, which makes the data complete for any later query. The queries on the ontology data are based on SPARQL [9], with the extension for supporting rules. A query Q can be modelled as a triple $Q = \langle dataset, ruleset, pattern \rangle$, where *dataset* refers to a set of named graphs, *ruleset* defines the rules for reasoning and *pattern* describes the query conditions and result terms. Note that *pattern* here follows the complete SPARQL specification, supporting `optional`, `filter`, `union`, and all other keywords. Both *dataset* and *ruleset* define that the query scope of Q is all ontology data belonging to *dataset* union with all inferred results derived by applying *ruleset* on the *dataset*. Note that if a user-defined rule predicate appears in the query, the DLP rules are applied by default. The intuition behind is, if a user-defined rule is applied in the query, the reasoning for the ontology data is required. An example SPARQL query Q is given as follows:

```
SELECT ?x ?z
FROM http://example.org/g1
WHERE {?x hasUncle ?y. ?y hasFather ?z.}
```

where `FROM` clause defines that Q will be executed on which named graph. Actually, the equivalent expression of Q in a triple form is:

$$Q = \langle \{\text{http://example.org/g1}\}, \\ \{R_{\text{onto}} \cup \{\text{hasUncle}(x, z) : \neg \text{hasFather}(x, y) \wedge \text{hasBrother}(y, z)\}\}, \\ \{(\text{?y}, \text{?z}) \mid \text{hasUncle}(\text{?x}, \text{?y}), \text{hasFather}(\text{?y}, \text{?z})\} \rangle$$

As Q includes a user-defined predicate `hasUncle`, the evaluation of the query requires both original and inferred data.

Problem Statement. Given a set of named graphs stored in an ontology store, a set of user-defined rules, and the combinations of the named graphs required for reasoning, how to compute inferred data such that (1) the inferred results derived from multiple named graphs can be effectively stored and (2) the incremental update/reasoning on the ontology data can be efficiently supported.

3 Our Proposed Approach

Data storage model. An RDB-based ontology repository is built on the relational database to store ontology data. An ontology repository D is composed of a set of tables T_i , where $1 \leq i \leq n$. In general, to support named graph, each table schema T_i in the ontology repository D is defined as $T_i = (uri, attr_{i_1}, \dots, attr_{i_j})$, where T_i is a table that consists of a mandatory attribute uri for storing named graph's URI and attributes $attr_{i_j}$ for the content of ontology data. For simplicity, the ontology repository could be regarded as a table `Triple(s,p,o,g)`, where s is subject, p is predicate, o is object and g is named graph URI.

3.1 Managing Inferred Results Uniformly

Generally, data in the ontology repository can be classified into three types: ontology data, inferred data for ontology rules, and inferred results for user-defined rules. For any type of inference results, we allocate a new named graph URI that is transparent to users. Thus, data in the ontology repository can be divided into *primitive* named graphs, which are the original ontology data and *derived* named graphs, which correspond to all inference results.

Let us first consider the inferred ontology data. Given a set of k primitive named graphs $NG = \{ng_i\}$ ($1 \leq i \leq k$), let uri_{NG} be the identifier of NG and G be the union of triples in ng_i , i.e., $G = \bigcup G_{ng_i}$. The inferred results, which are a set of implicit RDF triples, are derived by applying R_{onto} on G . We then produce a derived named graph $ng' = (uri_{ng'}, G_{ng'})$ such that (1) $G_{ng'}$ is the graph that contains all inferred triples and (2) $uri_{ng'}$ is a new generated named graph URI. Obviously, data in the derived named graph ng' are also in the form of RDF triple. Therefore, both primitive and inferred named graphs can be stored in the same set of relational tables. Note that given a primitive named graph set NG , it uniquely determines a derived named graph ng' .

For the inferred results from user-defined rules, suppose that a set user-defined rules R_{user} be applied on a set of k primitive named graphs $NG = \{ng_i\}$, where $1 \leq i \leq k$. Firstly, by default, the DLP rules R_{onto} are always applied on

$G = \bigcup G_{ng_i}$ and hence the derived named graph ng' is created for the inferred ontology data. Secondly, another new named graph $ng'' = (uri_{ng''}, G_{ng''})$ is generated such that (1) $G_{ng''}$ is the graph that includes all inferred triples derived by applying R_{user} on $G \cup G_{ng'}$ and (2) $uri_{ng''}$ is a new graph URI for labelling $G_{ng''}$. Because the user-defined rules only allow to introduce new predicates that are unary or binary, the inferred data of the user-defined rules are therefore viewed as a special named graph and managed in the same set of ontology tables. Note that, given a primitive named graph set NG and a set of user-defined rules R_{user} , the combination of them uniquely determines a derived named graph ng'' .

In our system, when the derived named graphs ng' and ng'' are generated from a set of primitive named graphs NG , two mappings $M_{ng'}(uri_{NG}, uri_{ng'})$ and $M_{ng''}(uri_{NG}, R_{user}, uri_{ng''})$ are created to record such derivation from NG (with the identifier uri_{NG}) to ng' and ng'' . Suppose that we execute a SPARQL query Q with the *pattern* on NG and R_{user} , i.e., $Q = \langle NG, R_{user}, pattern \rangle$, then we can rewrite Q into $Q' = \langle NG \cup \{ng'\} \cup \{ng''\}, \phi, pattern \rangle$. In other words, the translated query Q' will be evaluated on both primitive named graphs NG and derived named graphs ng' and ng'' , according to the mappings $M_{ng'}$ and $M_{ng''}$. Clearly, through the named graph concept, we uniformly organize original and inferred data storage and also evaluate query with ontology rules and user-defined rules.

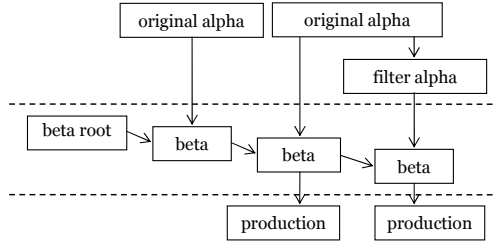


Fig. 1. The structure of the inference network

Example 1. Assume that a named graph set NG has triple $\langle \text{Tom}, \text{hasFather}, \text{Bob} \rangle$ and $\langle \text{Bob}, \text{hasBrother}, \text{Jensen} \rangle$. Given a query $Q = \langle NG, R_{user}, pattern \rangle$, where R_{user} is $\text{hasUncle}(x, z) :- \text{hasFather}(x, y) \wedge \text{hasBrother}(y, z)$ and $pattern$ is $\{ (?x, ?z) | ?x \text{ hasUncle } ?z \}$, the result $\langle \text{Tom}, \text{Jensen} \rangle$ is returned because the query is rewritten on the data $NG \cup \{ng'\} \cup \{ng''\}$ and ng'' includes the triple $\langle \text{Tom}, \text{hasUncle}, \text{Jensen} \rangle$. \square

3.2 Incremental Reasoning on an RDB-Based Inference Network

The inference network structure. Inspired by the classic Rete algorithm [4], we design an inference network on top of RDB to support incremental ontology update/reasoning in a batch-processing mode. The basic idea of the incremental

reasoning is to make the tradeoff between storage and update efficiency. Specifically, we create additional tables to store the intermediate inferred results, which are used for tracking the update of the inferred results. Given the rules for inference, a network can be built to link the original ontology data, intermediate and final inferred results together. Therefore, any update on the original ontology will be propagated along the network and the inferred results are incrementally updated accordingly, until the network reaches a stable state.

Figure 1 gives an overview of the inference network. There are three types of nodes in the network: *alpha nodes* (original alpha nodes and filter alpha nodes)¹ are bound with the original ontology data, *beta nodes* are bound with the intermediate inferred results, and *production nodes* produce the inferred results. Every node in the network is associated with some tuples in the ontology repository, which is omitted in this figure for simplicity.

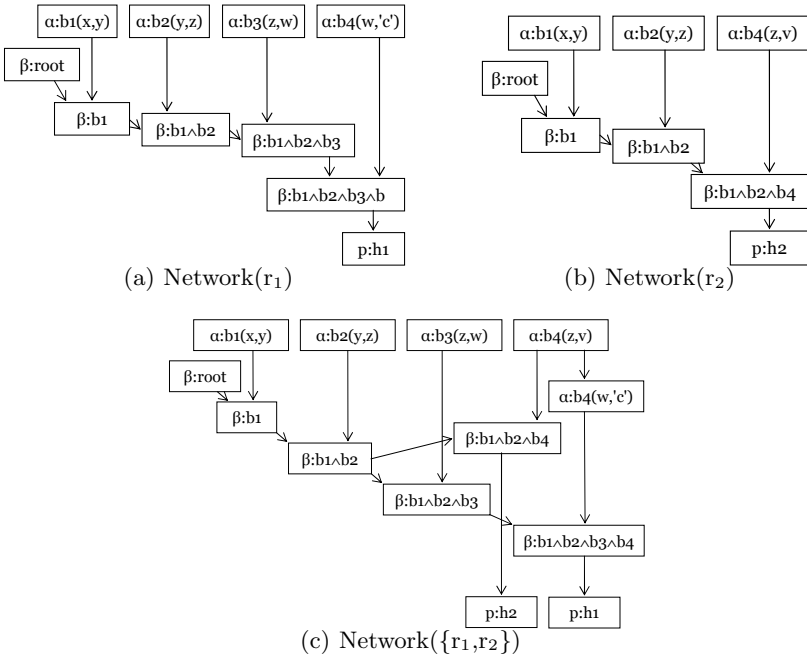


Fig. 2. An example for constructing inference network

Example 2. Given two rules $r_1 = h_1(x, y):-b1(x, y) \wedge b2(y, z) \wedge b3(z, w) \wedge b4(w, 'c')$ and $r_2 = h_2(x, y):-b1(x, y) \wedge b2(y, z) \wedge b4(z, v)$, let r_1 's and r_2 's inference networks be $Network(r_1)$ and $Network(r_2)$, which are shown in Figure 2(a) and 2(b) respectively. In Figure 2(a), we can see that an alpha node " $\alpha:b_1(x, y)$ " corresponds

¹ A filter alpha node is always a child node of an original alpha node and the input data of the filter alpha node is a subset of its parent's data.

to a formula $b_1(x, y)$ in r_1 's rule body; a beta node " $\beta:b_1 \wedge b_2$ " has a beta node " $\beta:b_1$ " and an alpha node " $\alpha:b_2(y, z)$ " as its parents, which indicates the join of the formulas $b_1 \wedge b_2$ appearing in r_1 's rule body; and a production node " $p:h_1$ ", corresponding to the rule head, has a beta node " $\beta:b_1 \wedge b_2 \wedge b_3 \wedge b_4$ " as its parent. Note that the path from the top beta node to the bottom beta node follows the order of the formulas in the rule body. A more complicated example is $Network(\{r_1, r_2\})$ shown in Figure 2(c). We can see that $Network(\{r_1, r_2\})$ can be constructed by merging $Network(r_1)$ and $Network(r_2)$ together. Note that the node " $\alpha:b_4$ " becomes the filter alpha node as its formula $b_4(w, 'c')$ equals to the binding of the variable v with the constant ' c '. \square

Generally, given a set of rules $R = \{r_1, \dots, r_m\}$, R 's inference network can be acquired by sharing the nodes of $Network(r_i)$ ($1 \leq i \leq m$). Note that, for a given rule, different orders of predicates in the rule body can lead to different network structures. However, the problem for optimizing the predicate order for each rule is out of the scope of our discussion.

Linking relational data with the inference network nodes. Now we need to define the mappings that are responsible for linking the tuples in the relational table to data associated with the formulas in the rules. In other words, the mappings feed the relational data into the alpha nodes, and point out the destination tables for the inferred data in the production nodes. We call such mappings as *tuple mappings*. Given a formula f in the rule, we define f 's tuple mapping $tm_f = \langle stmt, \alpha_{map} \rangle$, where $stmt$ is a SQL selection statement that defines an updatable view and α_{map} defines a mapping between f 's term(s) and the selected attributes in $stmt$. The tuple mapping is pre-defined based on the database schema of ontology repository.

$$\begin{aligned}
 tm_{hasUncle} &= \langle \text{SELECT } s, o \text{ FROM Triple WHERE } p=\text{hasUncle AND} \\
 &\quad (g=uri_{ng''}), \{ \langle x, s \rangle, \langle y, o \rangle \} \\
 tm_{hasBrother} &= \langle \text{SELECT } s, o \text{ FROM Triple WHERE } p=\text{hasBrother AND} \\
 &\quad (g=uri_{ng_1} \text{ OR } g=uri_{ng_2}), \{ \langle x, s \rangle, \langle y, o \rangle \} \\
 tm_{hasFather} &= \langle \text{SELECT } s, o \text{ FROM Triple WHERE } p=\text{hasFather AND} \\
 &\quad (g=uri_{ng_1} \text{ OR } g=uri_{ng_2}), \{ \langle x, s \rangle, \langle y, o \rangle \}
 \end{aligned}$$

Fig. 3. An example of tuple mappings

Example 3. Suppose that both original ontology data and inferred data be stored in a triple table $\text{Triple}(s, p, o, g)$, and an inference network is built on a named graph set $NG = \{ng_1, ng_2\}$, with a user-defined rule $\text{hasUncle}(x, z) : \text{hasFather}(x, y) \wedge \text{hasBrother}(y, z)$. Let the inferred result be the derived named graph ng'' . Then, we have three tuple mappings $tm_{hasUncle}$, $tm_{hasFather}$ and $tm_{hasBrother}$ for the formulas in the rule in Figure 3. \square

² The view defined by $stmt$ is updatable because the changes on alpha node and inferred results in the production node should be written back to the base tables.

Since the reasoning is performed on a set of primitive named graphs, the named graph information is always required as a constraint for the tuple mapping. Given a tuple mapping tm , we denote the set of named graph URIs appeared in $stmt$ as U_{tm} . In Example 3, $U_{tm_{hasBrother}}$ is $\{uri_{ng1}, uri_{ng2}\}$. Apparently, by setting U_{tm} in the tuple mappings for rules, we can control that, on which named graph(s) the reasoning is performed, and to which named graph the inference result is output.

Given a set of rules, once the tuple mapping is built, we can construct the inference network (notice that the process of constructing the network also corresponds to the process of the reasoning). During the network construction, the additional tables will be created on demand. In general, each node in the network may have two tables for data and delta data (i.e., the changes caused by ontology data update) respectively. In what follows, we discuss these tables in terms of the node type (see the detailed usage of these tables in the next section).

Original Alpha Node. Because its data table is always associated with an original table in the ontology repository, we only need to create one delta table with the same schema as the original table.

Filtered Alpha Node: Both data table and delta table have the same schema, which contains one column for each term in the corresponding formula and one mandatory column for identifying named graph.

Beta Node: Both data table and delta table have the same schema, which includes one column for each distinct variable appearing in the formula list and one column for each named graph inherited from the alpha nodes.

Production Node: Both data table and delta table have the same schema, which includes one column for the term in the rule head and a column *count* for recording how many times the same inferred result could be produced. This attribute is introduced for tracking the data deletion. Each production node does not need to maintain the named graph, as the information is recorded by the tuple mapping.

Meta-data Tables are a set of tables to store the meta-data of the inference network, which includes the tuple mappings and the network structure. With these meta data, the inference network can be rebuilt once system fails.

Incremental reasoning by change propagation. Let D_α be the dataset associated with an original alpha node. When an update Δ_α (either addition $+\Delta_\alpha$ or deletion $-\Delta_\alpha$) is made on D_α , then the incremental reasoning takes 3 steps: In the first step, the interfere network will update D_α by applying Δ_α on D_α ; then, D_α is notified to all child nodes of α ; and finally, Δ_α is cleared as empty. In the second step, for each notified child node n_i , Δ_{n_i} is computed according to Δ_α and D_{n_i} by join operation; then, D_{n_i} is updated and Δ_{n_i} is passed to the child nodes of n_i , before Δ_{n_i} is cleared. As a result, the Δ on the alpha node will be propagated along the network. We shall notice that, if a rule is self-dependant or multiple rules are mutually dependent (recursion among rules), then the Δ_{pn} of a production node pn can serve as the delta input for an alpha node. In the final step, the incremental update/reasoning is finished till the network reaches a stable state.

The difference between data addition and deletion is that for deletion, when getting the Δ_{pn} , we need to check whether the records need to be deleted from the original data table. This is because a result inferred by the rule head could be derived from multiple paths/sources.

3.3 Reasoning over Multiple Named Graphs

For each set of primitive named graphs $NG = \{ng_1, \dots, ng_k\}$ that requires ontology reasoning, we build an inference network, denoted as $InfNetwork(NG)$. As the DLP rules in R_{onto} are fixed, all inference networks have the same network structure. The only difference between the inference networks lies in the named graph in the tuple mappings. Recall that the named graph for a tuple mapping tm is represented as U_{tm} , which is a set of named graph URIs appeared in the selection statement. In general, given a named graph sets $NG = \{ng_1, \dots, ng_k\}$ that requires ontology reasoning, the corresponding inference network $InfNetwork(NG)$ is configured by setting the named graph information in the tuple mappings as below: according to the mapping $M_{ng'}(uri_{NG}, uri_{ng'})$, for all production nodes that correspond to the formulas appearing in R_{onto} 's heads, we set their U_{tm} as $\{uri_{ng'}\}$ for their tuple mapping. It indicates that all inferred results will be assigned to the derived named graph $uri_{ng'}$. For all alpha nodes that correspond to R_{onto} 's body tuples, we set their U_{tm} as $\{uri_{ng_1}, \dots, uri_{ng_k}, uri_{ng'}\}$ for the tuple mapping. It means that the source ontologies for reasoning are from the set $G_{ng_1} \cup \dots \cup G_{ng_k} \cup G_{ng'}$. Notice that $uri_{ng'}$ is included in U_{tm} because recursive ontology reasoning may be required, i.e., a rule is self-dependant or multiple rules are mutually dependent.

For user-defined rule reasoning, our solution can effectively manage the inference results of the different user-defined rule sets on the same primitive named graph set, by sharing the same inference network. Specifically, suppose that we have an inference network $InfNetwork(NG)$ that is built on top of a named graph set $NG = \{ng_1, \dots, ng_k\}$ based on R_{onto} . When adding two rule sets R_{user}^a and R_{user}^b to $InfNetwork(NG)$, we obtain two derived named graphs ng_a'' and ng_b''

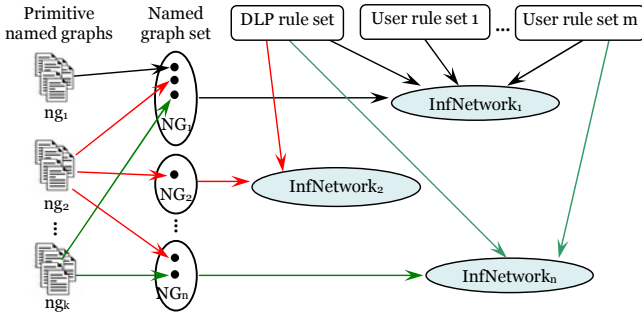


Fig. 4. An example of reasoning over multiple named graphs

which are the inferred results by applying R_{user}^a and R_{user}^b to $\{ng'\} \cup NG$ respectively, where ng' is the derived named graph for ontology reasoning. Because the new formulas introduced by R_{user}^a and R_{user}^b are always with different name graphs in the tuple mappings, the rule reasoning for R_{user}^a and R_{user}^b will not be affected with each other, but share the same ontology data as the reasoning input. Obviously, such inference network sharing can efficiently save the storage space and reduce the inference time.

Example 4. In Figure 4, given k primitive named graphs ng_1, \dots, ng_k , we first assign them to n named graph sets NG_1, \dots, NG_n according the pre-defined queries. Then we can build n inference networks, each of which corresponds to a named graph set. Similarly, a set of user-defined rules can also be added into multiple inference networks. Note that if data in ng_2 changes, $InfNetwork_1$, $InfNetwork_2$ and $InfNetwork_n$ will be triggered to perform the incremental reasoning.

4 Performance Study

We implemented the SOR v2.0 with the IBM DB2 v9.7 to evaluate the reasoning performance. All experiments were performed on a PC with an Intel 2.66GHz Duo Processor, 2GB RAM and one 160GB disk. UOBM 7 is used as our benchmark dataset and our previous SOR v1.0 6 as baseline, as the extensive comparison has been done with other typical RDF systems in 6.

4.1 Incremental Reasoning Performance

We first study the efficiency of the inference engine. The experiment is conducted on the UOBM datasets shown in Table 1. In the experiment, we insert triples into or remove triples from the existing named graph, where all newly-added triples never appear in the ontology repository and all triples being removed are randomly chosen from the ontology repository.

Table 1. The detailed statistics of the UOBM datasets

	#Original Triples	#Inferred Triples	#Total Triples
UOBM5	1.19×10^6	2.41×10^6	3.60×10^6
UOBM10	2.33×10^6	4.65×10^6	6.95×10^6
UOBM15	3.64×10^6	6.88×10^6	1.02×10^7
UOBM20	4.90×10^6	9.97×10^6	1.49×10^7
UOBM30	7.41×10^6	1.53×10^7	2.28×10^7

Figure 5 shows the incremental reasoning performance on the UOBM datasets, by varying the number of the triples to be updated from 1 to 100. From both Figure 5(a) and Figure 5(b), we can observe that for the same dataset, the time of the ontology reasoning goes up slowly as the number of triples are added or removed. Note that the vibrations of the lines in the two figures are caused by the reason that different triples use different predicates whose property (e.g., symmetric or transitive) affects the complexity of the ontology reasoning.

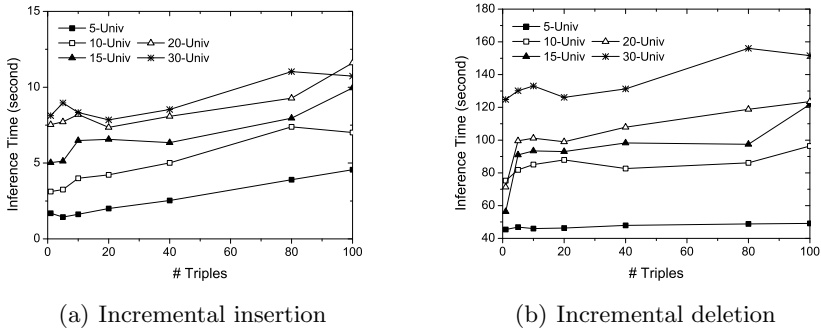


Fig. 5. Incremental reasoning performance

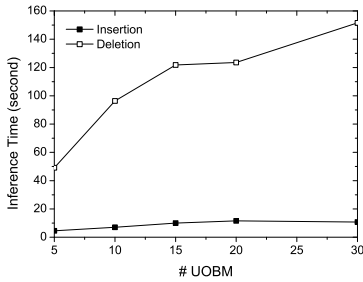


Fig. 6. Performance vs. data volume

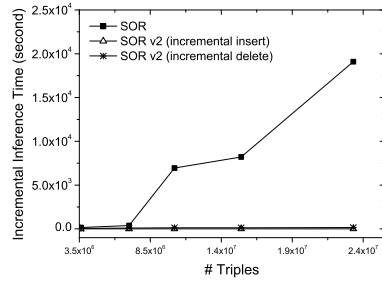


Fig. 7. Performance: SOR v2.0 vs. SOR

Figure 6 reports the performance against the data volume, by fixing the number of updated triples as 100. As the volume of the datasets increases, we find that the reasoning cost does not rise much. On the other side, as the removal of the inferred triples is more complex than the triple insertion, the reasoning cost of the ontology deletion is longer than that of the ontology insertion.

4.2 Reasoning Performance: SOR v2.0 vs. SOR

In Figure 7, we can see that the performance of the SOR v2.0 greatly outperforms that of our old system because all inferred results must be rebuilt even only one triple is added or removed. In the real-life applications, the ontology repository loads most of triples at the beginning and then incrementally updates its content. Our approach can satisfy the above requirement as it only needs to compute the inference results incrementally. And end-users can obtain their desired inferred knowledge very efficiently, while not waiting for hours for that.

5 Related Work

Ontology store with reasoning capability. Jena [12] supports a rule-based inference engine with forward and backward chaining algorithms. While Wu et al. [13] translates the DLP rules into SQLs and then executes these SQLs to perform reasoning on top of Oracle. However, to our best knowledge, the above systems cannot support *incremental reasoning* with ontology deletion.

DLP-based reasoning techniques. An ontology can impose semantics on RDF data and so the implicit ontologies can be derived by reasoning. DLP [5] is an intersection of Description Logic (DL) [1] and Logic Programming (LP) [3]. As a subset of the OWL expression capability, the DLP reasoning can be performed by Datalog rules. In general, the combination of both ontological reasoning and user-defined rule reasoning causes an undecidable problem [8]. To gain the decidability, in our work, we impose the constraint that the user-defined rules cannot affect the semantics of the properties in the ontology data.

Rete algorithm. The Rete algorithm [4] is originally a memory-based approach for pattern matching, which can support the incremental update in the production rule reasoning. However, our approach is not a trivial implementation because (1) we make a novel design on the network structure for supporting the idea of the Rete on RDB; and (2) we propose a method to build the inference network based on the named graphs and rules.

Incremental maintenance of materialized views. Many works (e.g., [10,11]) supporting the incremental maintenance of materialized views, can enable the incremental reasoning/update by (1) rewriting the rule set R for reasoning into R' and (2) applying a rule engine to evaluate R' for computing the delta data. By analogy, our approach can gain advantage due to two reasons: (1) when computing the delta data, because our approach maintains materialized intermediate reasoning results, less computation is required; and (2) the rules for ontology reasoning have significant amount of overlap among their conditions. In such a case, the materialized intermediate reasoning results can be shared in the inference network by a number of rules, which favors the performance.

6 Conclusion

In this paper, we report the work that can resolve the data update problem for a materialization-based ontology repository. Specifically, 1) we propose an effective data organization method that uniformly stores the original ontology, inferred and intermediate results, with the support of the named graph concept. 2) We propose a novel approach to build the inference network based on both named graphs and rules, which is key to enable the incremental reasoning over multiple ontologies in a batch-processing mode when data in an ontology are added/deleted. 3) We implement our proposed approach on top of our previous ontology repository system [6] and the experimental results show the effectiveness and efficiency of our solution.

References

1. Baader, R., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)
2. Bizer, C., Heath, T., Idehen, K., Berners-Lee, T.: Linked data on the web. In: Proc. of World Wide Web, pp. 1265–1266 (2008)
3. Ceri, S., Gottlob, G., Tanca, L.: Logic programming and databases. Springer-Verlag New York, Inc., New York (1990)
4. Forgy, C.: Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence* 19, 17–37 (1982)
5. Grosz, B.N., Horrocks, I., Volz, R., Decker, S.: Description logic programs: combining logic programs with description logic. In: WWW, pp. 48–57 (2003)
6. Ma, L., Wang, C., Lu, J., Cao, F., Pan, Y., Yu, Y.: Effective and efficient semantic web data management on db2. In: Proc. of SIGMOD (2008)
7. Ma, L., Yang, Y., Qiu, Z., Xie, G., Pan, Y.: Towards a complete OWL ontology benchmark. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 125–139. Springer, Heidelberg (2006)
8. Motik, B., Sattler, U., Studer, R.: Query answering for owl-dl with rules. *Web Semantics: Science, Services and Agents on the World Wide Web* 3(1), 41–60 (2005)
9. SPARQL Query Language for RDF, <http://www.w3.org/TR/rdf-sparql-query/>
10. Staudt, M., Jarke, M.: Incremental maintenance of externally materialized views. In: VLDB, pp. 75–86 (1996)
11. Steffen, R.V., Staab, S., Motik, B.: Incremental maintenance of materialized ontologies. In: Chung, S., Schmidt, D.C. (eds.) CoopIS 2003, DOA 2003, and ODBASE 2003. LNCS, vol. 2888, pp. 707–724. Springer, Heidelberg (2003)
12. Wilkinson, K., Sayers, C., Kuno, H., Reynolds, D.: Efficient rdf storage and retrieval in jena2. In: Proc. of SWDB (2003)
13. Wu, Z., Eaden, G., Das, S., Chong, E.I., Kolovski, V., Annamalai, M., Srinivasan, J.: Implementing an inference engine for rdfs/owl constructs and user-defined rules in oracle. In: Proc. of ICDE (2008)

General-Purpose Ontology Enrichment from the WWW

Mohammed Maree¹, Mohammed Belkhatir², and Saadat M. Alhashmi¹

¹ Monash University, Sunway Campus

² University of Lyon & CNRS, France

{mohammed.maree, alhashmi}@monash.edu,
mohammed.belkhatir@iut.univ-lyon1.fr

Abstract. Ontology enrichment is required when the knowledge captured by the ontology is out of date or unable to capture the specified user requirements in a specific domain. In this paper we present an automatic statistical/semantic framework for enriching general-purpose ontologies from the World Wide Web (WWW). Using the massive amount of information encoded in texts on the web as a corpus, missing background knowledge such as concepts, instances and relations can be discovered and exploited to enrich general-purpose ontologies. The benefits of our approach are: (i) enabling ontology enrichment with missing background knowledge, and thus, enabling the reuse of such knowledge in future. (ii) saving time and effort required to manually enrich and update the ontologies. Experimental results indicate that the techniques used to enrich ontologies are both effective and efficient.

Keywords: ontology, knowledge acquisition, enrichment, semantic relatedness, experimental validation.

1 Introduction

General-purpose ontologies such as WordNet [1] and Cyc[2] are structured networks of concepts that are interconnected by different types of sub-sumption and semantic relations from multiple knowledge domains. These ontologies are developed to provide explicit specifications of general-purpose domains in a machine readable and understandable format. However, they have drawbacks at both construction and enrichment levels. On the one hand, manual construction of generic ontologies is a complex and time-consuming task, as it requires experts in different domains to manually construct and maintain them. On the other hand, such ontologies don't completely cover the specified user requirements in a particular domain, due to the lack of background knowledge such as concepts or instances defined in them. For example, concepts such as "Corporate Body" or instances such as "Monash University" are missing from WordNet. To overcome these drawbacks, we propose an automatic statistical/semantic framework to enrich general-purpose ontologies with missing background knowledge from the WWW. Our source of ontology enrichment is the massive amount of information encoded in texts on the web. First, we exploit statistical techniques based on the Normalized Retrieval Distance (NRD) function to measure the semantic relatedness between concepts that are defined in the ontology

and other concepts or instances that are missing i.e. not defined in it. Then, based on the obtained semantic relatedness measures, a semantic-based ontology enrichment technique is employed to enrich the ontology according to a list of pre-defined lexico-syntactic patterns. The main contributions of our work are:

- Combining semantic relatedness measures and pattern acquisition techniques for information extraction.
- Automatic enrichment of general-purpose ontologies from the WWW.

The rest of this paper is organized as follows. Section 2 presents an overview of previous research in the ontology enrichment area. A general overview of the proposed framework is given in Section 3. Section 4 describes the details of the proposed framework. Section 5 discusses experiments with enriching a generic ontology. Section 6 presents the conclusions and outlines the future work.

2 Related Work

Several approaches dealing with automatic knowledge acquisition and ontology enrichment from textual data have been proposed [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]. For instance, the authors of [3] propose to automatically acquire hyponyms from large text corpora. Their approach is similar to the approach described in this paper in the way it defines a list of lexico-syntactic patterns to derive hyponymy relation between terms. However, in our work we defined more patterns to derive other types of relations such as synonymy and meronymy relations. In addition, we use statistical techniques to measure the semantic relatedness between the concepts and instances of the ontology and the newly discovered concepts and instances from the textual documents on the WWW. Agirre et al. [4] propose to enrich existing ontologies using the web. For concepts in WordNet they build topic signatures consisting of a list of topically related words. To identify the sense of a new word, they compare the new word's context to the topic signatures for each sense branch and choose the highest ranking branch. Maedache et al. [5] employ tree-descending and tree-ascending clustering algorithms to determine the concept of a new word. The authors of [6] use concept clustering where concepts are grouped according to the semantic distance between each other to make up hierarchies of the ontology. In [10] the authors propose a semi-automatic approach to enrich an ontology by mining the WWW. They utilized statistical information of word usage to propose new concepts to enrich the ontology. The authors of [13] employ Formal Concept Analysis (FCA) to learn concept hierarchies from text corpora in a specific domain.

3 General Overview

Figure 1 depicts the proposed framework's architecture, which is discussed below. The framework consists of the following modules:

- Natural Language Processing (NLP) and Entity Recognition (ER): First, an entity recognizer is employed to obtain named-entities from the text. Next, n-gram text

tokenization function is utilized where unigram, bigram and trigram tokens are obtained from the text and checked whether they are defined in the ontology or not. This module is further explained in section 4.1.

- **Missing Background Knowledge Handler:** A statistical technique that measures the semantic relatedness between concepts and instances that are defined in the ontology and those that are missing from it. This technique is based on the NRD and Token Hits (TH) functions that are further explained in section 4.2.
- **Semantic-Based Ontology Enrichment technique:** Enriches the ontology with knowledge triples (entity-relation-concept) based on the semantic relatedness measures obtained from the previous step and according to a list of pre-defined lexico-syntactic patterns. An entity in this context refers to a concept or an instance of a concept. This technique is detailed in section 4.3.

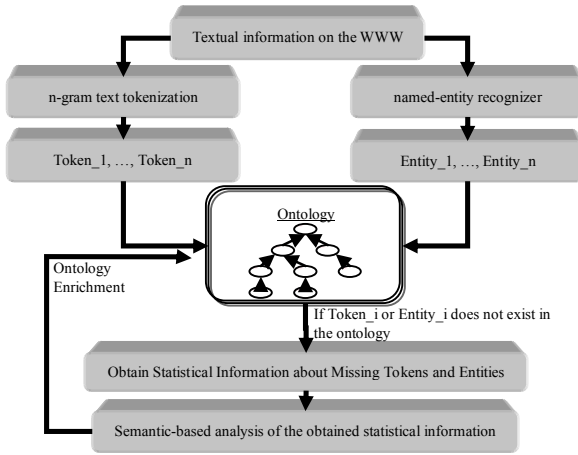


Fig. 1. Architecture of the Proposed Ontology Enrichment Framework

4 Detailed Steps of the Proposed Framework

Before we detail the steps of the proposed framework, we formalize the use of the terms “Ontology”, “Ontology Enrichment”, “Normalized Retrieval Distance (NRD)” and “Token Hits (TH)” functions.

Definition 1: Ontology: An ontology Ω is *quintuple*, $\Omega = \langle C, P, I, V, A \rangle$ where:

- C : is the set of ontology concepts. The concept hierarchy of Ω is a pair (C, \leq) , where \leq is an order relation on $C \times C$. We call $c \in C$ the set of concepts, and \leq the sub-concept relation.
- P : is the set of properties.
- I : is the set of instances or individuals
- V : is the set of property values
- A : is the set of axioms such as constraints

Definition 2: Ontology Enrichment: An ontology enrichment algorithm takes a given text corpus ζ and a given concept hierarchy from Ω as input and produces for each $c \in C$ a set of $S(c) \subseteq T(\zeta)$ where:

- $S(c)$ is the set of suggested enrichment candidates for c . A suggested candidate $t \in T(\zeta)$ is a word or compound word from ζ
- $T(\zeta)$ is the set of words from ζ

The set of suggested candidates $S(c)$ can be obtained using the NRD function and based on a threshold value ν using equation 1.

$$S(c, \nu) := \{t \in T(\zeta) \mid \text{NRD}(c, t) \geq \nu\} \quad (1)$$

Definition 3: Normalized Retrieval Distance (NRD): is an adopted form of the Normalized Google Distance (NGD) [14] function that measure the semantic relatedness between pairs of entities (such as concepts or instances): Given two entities E_{mis} and E_{in} , the Normalized Retrieval Distance between E_{mis} and E_{in} can be obtained as follows:

$$\text{NRD}(E_{\text{mis}}, E_{\text{in}}) = \frac{\max\{\log f(E_{\text{mis}}), \log f(E_{\text{in}})\} - \log f(E_{\text{mis}}, E_{\text{in}})}{\log M - \min\{\log f(E_{\text{mis}}), \log f(E_{\text{in}})\}} \quad (2)$$

where,

- E_{mis} is an entity that is not defined in the ontology
- E_{in} is an entity that exists in the ontology
- $f(E_{\text{mis}})$ is the number of hits for the search entity E_{mis}
- $f(E_{\text{in}})$ is the number of hits for the search entity E_{in}
- $f(E_{\text{mis}}, E_{\text{in}})$ is the number of hits for the search entities E_{mis} and E_{in}
- M is the number of web pages indexed by the search engine

Definition 4: Token Hits (TH) function: is employed to improve the NRD function at both performance and accuracy levels. Given a set S of n -gram tokens ($n= 1-3$), the TH function returns the number of search hits for each of the tokens in S as follows:

$$\text{TH}(n\text{-gram}) = Q("n\text{-gram}") \quad (3)$$

where,

- " n -gram" is the search token
- $Q("n$ -gram") is the number of hits for the search token

4.1 Natural Language Processing and Entity Recognition

The first step towards information extraction is Entity Recognition (ER). To do this, we employ GATE [15], which is a syntactical pattern matching entity recognizer enriched with gazetteers. Although the coverage of GATE is limited to a certain number of named-entities, additional rules can be defined in order to expand its coverage. However, the process of manual enrichment of GATE's rules can be difficult and time consuming task. Therefore, we exploit the ontology itself as part of

the named-entity recognition step. In this context, entities that are not defined in GATE are submitted to the ontology to find whether they are defined in it or not. Then, we apply the following NLP steps on the extracted texts from the WWW:

1. **Stopword removal:** Some of the words, for example, a, an, the, of,...etc occur frequently in the textual information without semantic significance. Therefore, those words are removed based on a pre-defined list that includes them.
2. **n-gram text tokenization:** First, the input text is spilt into tokens of lengths from 1 to 3. Then, each token is submitted to the ontology to find whether it exists in it or not. Tokens that do not exist in the ontology are considered as missing background knowledge and are furtherer processed by the statistical techniques.
3. **Part-of-speech tagging:** Each token that exist in the ontology is tagged by the grammatical category that it belongs to such as (Noun, Verb, etc). For instance, the token “book” belongs to two different categories (Nouns and Verbs).

The following example illustrates the abovementioned steps. In this example we exploit WordNet as a supplementary source of ER.

Example1: Part of an article on Java Island taken from Wikipedia encyclopedia.

“Java (Indonesian: *Jawa*) is an island of Indonesia and the site of its capital city, Jakarta. Once the centre of powerful Hindu-Buddhist kingdoms, Islamic sultanates, and the core of colonial Dutch East Indies, Java now plays a dominant role in the economic and political life of Indonesia.”

First, the stopword removal function removes stopwords from the text. For example, the words (the, an, a) and characters (,) , : , are removed. Then, the n-gram tokenization algorithm tokenizes the input text into unigram, bigram and trigram tokens. After this step, each token is submitted to the ontology to find whether it is already defined in it or not. The tokens and entities that are not recognized by GATE or not defined in the ontology are considered as missing background knowledge and further processed by the statistical techniques.

4.2 Statistical Information about the Missing Background Knowledge

At this step, statistical functions are utilized to suggest a set of ontology enrichment candidates. To do this, we first utilize the Token Hits (TH) function. In this context, exact match queries in the form of $Q = \text{“n-gram token”}$ are submitted to retrieve the number of Google hits for each token. For example, to retrieve the number of Google hits for the bigram token “Hindu-Buddhist Kingdoms”, the query $Q = \text{“Hindu-Buddhist Kingdoms”}$ is submitted to Google search engine. The output of this step is a set of n-gram tokens each associated with the number of its Google hits > 0 . This set is suggested as the set of candidate inputs to the NRD function. Algorithm 1 demonstrates the process of acquiring the number of Google hits for each token.

Algorithm 1. Token Hits Function

Input: n-gram tokens
Output: tokens with the number of Google hits > 0

```

1:   String [] Results;
2:   int value;
3:   for each token in Tokens
4:     value=makeQuery("token", "Google");
5:     if(value>0)
6:       Results.add("token");
7:     end if
8:   end loop

```

For each suggested token, the NRD function measures its semantic relatedness to other entities that were recognized by both the named-entity recognizer and the ontology. Since the NRD function utilizes more than one search engine, it returns different semantic relatedness measures according to these search engines. Therefore, in our approach, we sum up all NRD measures for each candidate token and only consider the highest value returned. Algorithm 2 explains the procedure of acquiring statistical information about the tokens that are not defined in the ontology.

Algorithm 2. Normalized Retrieval Distance Function

Input: suggested tokens from the TH function, $T_{missing}$, tokens recognized by GATE and the ontology, T_{in}
Output: semantic relatedness measures between $T_{missing}$ and T_{in}

```

1:   int[] result;
2:   for each  $t_{missing} \in T_{missing}$ 
3:     for each  $t_{in} \in T_{in}$ 
4:       result=NRD( $t_{missing}$ ,  $t_{in}$ )
5:     end for
6:   end for

```

The NRD function takes as input pairs of tokens and produces as output the measures of semantic relatedness between them. For example, Table 1 shows the semantic relatedness measures between the token “jawa” and the set of tokens: {Java, Island, Indonesia}.

Table 1. Semantic Relatedness Measures for the Token “Jawa”

<i>Token</i>	<i>Java</i>	<i>island</i>	<i>Indonesia</i>
jawa	0.72	0.56	0.69

4.3 Semantic-Based Analysis of the Obtained Statistical Information

The purpose of this step is to derive the semantic information from the obtained results from the NRD function. To do this, we defined a list of lexico-syntactic patterns to derive synonymy, hypernymy, hyponymy, meronymy and holonymy relations. These types of relations can be automatically obtained by utilizing the Semantic Relation Extractor (SRE) function. For each pair of semantically related tokens, the SRE function returns the number of their hits by submitting each of the patterns to the search engines. Then, relations between the token pairs are suggested

based on the highest values returned. Algorithm 3 shows the steps of the SRE function. It takes as input pairs of tokens with strong semantic relatedness measures and produces as output the suggested semantic relation between them based on the lexico-syntactic patterns. For each pattern, the `makeQuery` function (Line 7) submits exact match queries including both tokens. We considered both singular and plural forms of the patterns and excluded patterns that include negation operators such as “No $T_{missing}$ is a(n) T_{in} ”.

Algorithm 3. Semantic Relation Extractor Function

Input: Semantically related tokens, (missing tokens, $T_{missing}$, tokens exist in the ontology, T_{in})

Output: suggested relations between tokens

```

1:  String [] suggestedRelations
2:  String [] Patterns;
3:  int[] value;
4:    for each  $t_{missing} \in T_{missing}$ 
5:      for each  $t_{in} \in T_{in}$ 
6:        for each pattern  $p$  in Patterns
7:          value.add(makeQuery( $t_{missing}$ ,  $p$ ,  $t_{in}$ ));
8:        end for
9:        suggestedRelations.add(max(value));
10:     end for
11:  end for

```

Example 2: A corporate body is an organization or group of persons that is identified by a particular name, and that acts, or may act, as an entity¹.

From Example 2 we find that the token “Corporate Body” is missing from WordNet. However, based on the results obtained by the statistical technique we are able to find that this token is semantically related to “Organization” which is defined in WordNet. But, at this step we don’t know what type of semantic relation(s) may exist between them. Therefore, we utilize the SRE function by submitting patterns in the form of exact match queries Q_i such as:

- Q_1 = “Corporate Body is an organization”, which outputs 80,700 hits result
- Q_2 = “Corporate Body is a kind of Organization”, which outputs 0 hits result
- Q_3 = “Corporate Body is a part of Organization”, which outputs 0 hits result
- Q_4 = “Corporate Body is same as Organization”, which outputs 0 hits result

Based on the number of hits returned by the search engine for the queries Q_i , relations defined in the patterns are suggested to be used to enrich the ontology with “Corporate Body”. When applying this step we notice that for some of the semantically related tokens, no results are returned for any of the patterns. Therefore, in this case, we define the “Related To” relation between such tokens and use it to enrich the ontology. For instance, in Example 1 in section 4.1, we notice that that the token “Hindu-Buddhist” is missing from WordNet. Based on the NRD function, the strongest semantically related token to “Hindu-Buddhist” is “Indonesia”. However,

¹ http://www.itsmarc.com/crsbeta/mergedProjects/scmshelf/scmshelf/g_220_corporate_bodies_shelf.htm

when applying the SRE function the returned results were 0 for all of the patterns. Therefore, although none of the used patterns represented the type of semantic relation that holds between “Hindu-Buddhist” and “Indonesia”, we are still able to enrich the ontology with this token by defining the “Related To” relation between it and “Indonesia”.

4.4 General-Purpose Ontology Enrichment

Enriching the ontology with new concepts or instances requires finding the appropriate branch in the hierarchy of the ontology for locating them i.e. we need to find the concept in the hierarchy of the ontology that best approximates the meaning of the new concept or instance. Below we summarize the different cases that we consider for ontology enrichment.

Case 1: Concept or instance C is missing from the ontology, and C is semantically related to the concept X which already exists in the ontology. By utilizing the SRE function, we find that C is related to the concept X by one of the semantic relations in the defined patterns. The concept X has only one sense in the ontology, i.e. there is only one semantic path that originates from X . In this case, we update the hierarchy of the semantic path by attaching the missing concept or instance C to the concept X using the derived semantic relation between them. For example, the concept “Concept” in WordNet has only one sense. Therefore, any term considered as a sub-concept of this concept will be directly located under it.

Case 2: Concept or instance C is missing from the ontology, and C is semantically related to the concept X which is already defined in the ontology. By utilizing the SRE function, we find that C is related to the concept X by one of the semantic relations in the defined patterns. At the same time, the concept X has more than one sense in the ontology, i.e. there is more than one semantic path originating from X . In this case, we compute the similarity between the list of concepts in the semantic paths that originate from the concept X and the list of tokens in the tokenized text. Then, for the most similar semantic path(s), we update its/their hierarchy by attaching the missing concept or instance C to the concept X using the derived semantic relation between them. For Example, in Example 2 in section 4.3 we found that “*Corporate Body*” is semantically related to “*Organization*”. However, the concept “*Organization*” has 7 different senses in WordNet. This means that we need to find the most related sense of the concept “*Organization*” to “*Corporate Body*”. To do this, we compare the concepts in the semantic paths that originate from “*Organization*” to the tokens that are extracted from the text. Based on this comparison, the new token “*Corporate Body*” is attached to the appropriate semantic paths as shown in Figure 2.

Case 3: Concept or instance C is missing from the ontology, and C is semantically related to more than one Concept $\{X, Y, Z\}$ in the ontology. By utilizing the SRE function, we find that C is related to each of the concepts $\{X, Y, Z\}$ by one of the semantic relations in the defined patterns. In this case, based on the number of senses of each of the concepts $\{X, Y, Z\}$, we apply either Case 1 or Case 2.

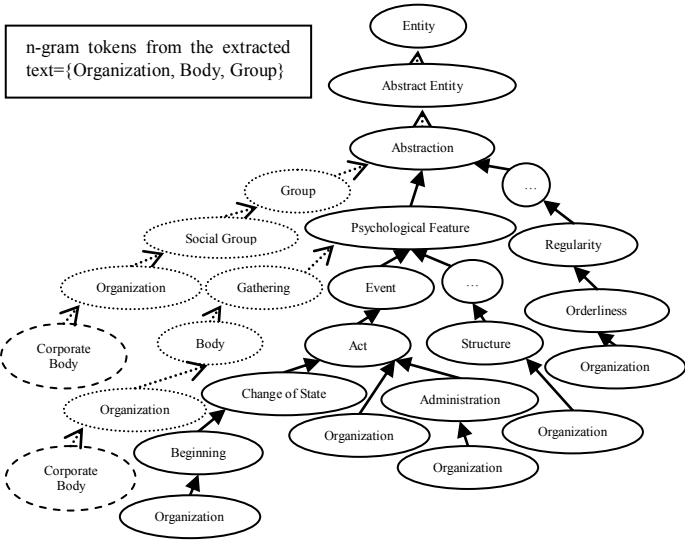


Fig. 2. Case 2, Enriching the Ontology with a Missing Concept with Multiple Senses

5 Experimental Results

This section describes the experiments that are conducted to evaluate the effectiveness of the proposed statistical/semantic ontology enrichment techniques. All solutions are implemented in Java and experiments are performed on a PC with dual-core CPU (3000GHz) and 2 GB RAM. The operating system is OpenSuse 11.1. We carried out experiments using WordNet ontology and a manually collected corpus that consists of 500 articles from different domains such as (Sport, Food kinds, Programming Languages, Countries and Cities, and Universities). For each domain, text in each article is analyzed through employing GATE and the NLP steps detailed in section 4.1. Then, we utilized both of the semantic relatedness measurement and automatic pattern acquisition techniques to automatically enrich the ontology with missing background knowledge.

In order to provide a ground for evaluating the quality of our results, we manually identified all tokens (concepts and instances) that are missing in the ontology. We built *expert enrichments* based on our knowledge and experience in the same fashion as highlighted in [16] and [17]. Then, we compared the expert enrichments to the enrichments produced by our system. We used the precision indicator in order to measure the quality of our results. This measure is defined as follows:

Precision (P): Is the percentage of the correctly enriched tokens in all enriched tokens. In this context, correctly enriched tokens are the concepts and instances that are located in the right semantic path(s) in the ontology.

Table 2 shows the number of tokens that were extracted from the articles of different domains. At this step, all of the non-recognized tokens are considered as missing background knowledge from WordNet ontology.

Table 2. Recognized and Non-Recognized Tokens Extracted from the Texts

<i>Domain Articles</i>	<i># of Recognized Tokens</i>	<i># of Non-Recognized Tokens</i>
Articles on Animals	744	6019
Articles on Programming Languages	328	3409
Articles on Countries & Cities	488	4012
Articles on Food Kinds	250	1647
Articles on Universities	454	3340
Articles on Science	476	4153
Articles on Sports	325	3224

As shown in Table 2, the number of tokens that are recognized by GATE, the NLP, and the ontology (WordNet) is much smaller than the number of tokens that were not recognized. This is due to several reasons. First, GATE and WordNet have limited domain coverage. Second, when utilizing the text tokenization and NLP steps, most of the bigram and trigram tokens are either meaningless tokens or not covered by the ontology. Therefore, we filtered the set of non-recognized tokens by eliminating meaningless ones. To do this, we employ the Token Hits (TH) function detailed in section 4. To evaluate this function, we manually eliminated the set of non-recognized tokens that has no meaning. For example, we manually eliminated the following tokens from the set of non-recognized tokens in the “Animals domain”: {“*animals generally diurnal*”, “*Both plant animals*”, “*Bears aided excellent*”, “*areas most*”, “*caused large, etc*}. In the next part of experiments we compute the precision of the Token Hits (TH) function. The precision of this function is defined as follows:

Precision (P): Is the percentage of the correctly eliminated tokens in all eliminated tokens. Eliminated tokens in this context are the tokens that are not recognized by the entity recognition techniques.

Tables 3 and 4 show the precision of the TH function in both eliminating and retaining non-recognized tokens.

Table 3. Precision of the Token Hits Function in Eliminating Non-Recognized Tokens

Domain Articles	# of “manually” Eliminated Tokens	# of “Token Hits –Based” Eliminated Tokens	Precision (P)
Articles on Animals	4989	4221	0.84
Articles on Programming Languages	2532	2108	0.83
Articles on Countries & Cities	2891	2632	0.91
Articles on Food Kinds	956	783	0.81
Articles on Universities	2578	2305	0.89
Articles on Science	3015	2789	0.92
Articles on Sports	3011	2901	0.96

As shown in Tables 3 and 4, the number of manually eliminated tokens is very big compared to the set of non-recognized tokens. The reason is because most of the bigram and mainly trigram tokens have no meanings. Therefore, we manually eliminated all of such tokens. Figure 3 shows the percentage of error rate in the TH function in eliminating non-recognized tokens.

Table 4. Precision of the Token Hits Function in Retaining Non-Recognized Tokens

Domain Articles	# of “manually” retained Tokens	# of “Token Hits – Based” retained Tokens	Precision (P)
Articles on Animals	1030	1798	0.57
Articles on Programming Languages	877	1301	0.67
Articles on Countries & Cities	1121	1380	0.81
Articles on Food Kinds	691	864	0.79
Articles on Universities	762	1035	0.73
Articles on Science	1138	1364	0.83
Articles on Sports	213	323	0.65

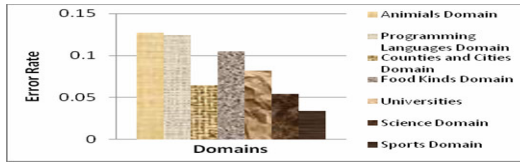


Fig. 3. Error Rate of Term Hits Function in Eliminating Non-Recognized Tokens

Although the Token Hits function has high error rate in eliminating non-recognized tokens, we still get higher performance and accuracy results when using this function as a pre-requisite to the Normalized Retrieval Distance (NRD) function. Therefore, instead of applying the NRD function on the complete set of non-recognized tokens, we only apply it on a filtered set suggested by the TH function. We call this set as the set of “meaningful entities”. At this phase of experiments, we are interested in this set as it represents the set of suggested ontology enrichment candidates. To enrich the ontology with this set, we first employ the NRD function explained in section 4. Then, we utilize the automatic pattern acquisition techniques explained in section 4.3. The purpose of this step is to derive the semantic relations that may hold between the pairs of semantically related entities. As discussed in section 4.3, at this step, we notice that for some of the strongly related entity pairs no results are returned. This means that none of the defined patterns could represent the semantic relation(s) that may exist between them. However, since such entity pairs have very strong semantic relatedness, we use the proposed “Related To” relation to enrich the ontology. The final phase of evaluation is to evaluate the precision of the ontology enrichment process. To do this, we compared between the sets of manually and automatically enriched entities for each domain. The results of this step are shown in Table 5 below:

Table 5. Results of Enriching WordNet Ontology

Domain	Precision (%)
Articles on Animals	81%
Articles on Programming Languages	84%
Articles on Countries & Cities	78%
Articles on Food Kinds	69%
Articles on Universities	73%

Although our system showed good precision in enriching the ontology, the number of entities that it was able to enrich the ontology with was small. The reason behind this is because of the limited number of used patterns. In many cases the system suggested to attach the new entity to the ontology based on the “Related To” relation. However, the “Related To” relation does not really capture the semantics of the actual relation between them. Therefore, this leads us to think of using another option instead of using this relation only. Our idea here is to submit both terms using the query “Entity_1 * Entity_2” to Google search engine. The (*) in this query will be replaced by one word or none in a webpage. We will attempt to analyze the returned results in order to extract the lexical patterns between those entities. Thus, further experiments will be carried out for performing this analysis.

6 Conclusion and Future Works

In this paper we present a general framework for enriching general-purpose ontologies with missing background knowledge. In our approach, we combined semantic relatedness measures and pattern acquisition techniques to extract information from textual data on the WWW. In addition, multiple named-entity recognition algorithms such as GATE, NLP and using the ontology itself are utilized. Initial experiments using 500 articles on different domains showed promising precision results. We plan to test the proposed framework using additional automatically extracted patterns to derive other types of semantic relations. We also plan to exploit other sources of knowledge such as domain specific ontologies in order to be used to enrich the ontologies. Furthermore, we plan to do more experimentation on enriching domain specific ontologies by using multiple sources of information such as domain-specific ontologies and domain articles on the WWW.

References

1. Miller, G.A.: WordNet: A Lexical Database for English. *Communications of the ACM* 38(1), 39–41 (1995)
2. Lenat, D.B.: Cyc: a large-scale investment in knowledge infrastructure. *Communications of the ACM* 38(11), 33–38 (1995)
3. Hearst, M.A.: Automatic acquisition of hyponyms from large text corpora. In: *Proceeding of COLING 1992, Nantes, France*, pp. 539–545 (1992)
4. Eneko Agirre, E.H., Ansa, O., Martinez, D.: Enriching Very Large Ontologies Using the WWW. In: *Proc. ECAI Workshop on Ontology Learning* (2000)
5. Maedche, A., Staab, S.: Ontology Learning for the Semantic Web. *IEEE Intelligent Systems* 16(2), 72–79 (2001)
6. Faure, F., Poibeau, T.: First Experiments of using semantic knowledge learned by ASIUM for information extraction task using INTEX. In: *Proceedings of ECAI Workshop on ontology Learning*, pp. 7–12 (2000)
7. Clerkin, P., Cunningham, P., Hayes, C.: Ontology Discovery for the Semantic Web Using Hierarchical Clustering. In: *Proc. of (ECML/PKDD 2001)*, pp. 1–12 (2001)
8. Maedche, V., Pekar, A., Staab, S.: *Ontology Learning Part One- On Discovering Taxonomic Relations from the Web*, pp. 301–322. Springer, Heidelberg (2002)

9. Xu, F., Kurz, D., Piskorski, J., Schmeier, S.: A Domain Adaptive Approach to Automatic Acquisition of Domain Relevant Terms and Their Relations with Bootstrapping. In: Proc. Third Int'l Conf. Language Resources and Evaluation, LREC 2002 (2002)
10. Faatz, A., Steinmetz, R.: Ontology Enrichment with Texts from the WWW. In: Semantic Web Mining, WS 2002 (2002)
11. Khan, L., Luo, F.: Ontology Construction for Information Selection. In: Proc. of 14th IEEE Int'l Conf. Tools with Artificial Intelligence., pp. 122–127 (2002)
12. Navigli, R., Velardi, P., Gangemi, A.: Ontology Learning and Its Applications to Automated Terminology Translation. *IEEE Intelligent Systems* 18(1), 22–31 (2003)
13. Cimiano, P., Huth, A., Staab, S.: Learning concept hierarchies from text corpora using formal concept analysis. *JAIR* 24, 305–339 (2005)
14. Cilibrasi, R., Vitanyi, P.: The Google Similarity Distance. *IEEE Transactions on Knowledge and Data Engineering* 19(3), 370–383 (2007)
15. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: GATE: a framework and graphical development environment for robust NLP tools and applications. In: Proc. of the 40th Anniversary Meeting of the Association for Computational Linguistics, Phil., USA (2002)
16. Giunchiglia, F., et al.: S-Match: an Algorithm and an Implementation of Semantic Matching. In: Proc. of ESW, pp. 61–75 (2004)
17. Trojahn, C., et al.: A Cooperative Approach for Composite Ontology Mapping. In: Spaccapetra, S. (ed.) *Journal on Data Semantics X*. LNCS, vol. 4900, pp. 237–263. Springer, Heidelberg (2008)

QuerySem: Deriving Query Semantics Based on Multiple Ontologies

Mohammed Maree¹, Saadat M. Alhashmi¹, and Mohammed Belkhatir²

¹ Monash University, Sunway Campus

² University of Lyon & CNRS, France

{mohammed.maree, alhashmi}@monash.edu

mohammed.belkhatir@iut.univ-lyon1.fr

Abstract. Internet search engines are indispensable tools that assist users to find information on the World Wide Web (WWW). These search engines use different keyword-based indexing techniques to index Web Pages. Although this approach assist users in finding information on the Web, many of the returned results are irrelevant to the user's information needs. This is because of the "semantic-gap" between the meanings of the keywords that are used to index Web Pages and the meanings of the terms used by the user to formulate his query. In this paper, we introduce an approach to employ knowledge represented by multiple large-scale general-purpose ontologies to derive the semantic aspects of the user's query. In addition, we utilize statistical-based semantic relatedness measures to compensate for missing background knowledge in the exploited ontologies. Experimental instantiation of the proposed system validates our proposal.

Keywords: query, semantics, large-scale ontology, missing background knowledge.

1 Introduction

Current Internet search engines still suffer from low precision/recall ratio [1]. One of the main reasons is because these search engines use keyword-based indexing techniques to index Web Pages. Although this approach assist users in finding information on the Web, many of the returned results are irrelevant to the user's information needs. This is due to the "semantic-gap" between the meanings of the keywords that are used to index Web Pages and the meanings of the terms used by the user to formulate his query. This paper presents an approach that combines knowledge represented by multiple large-scale general-purpose ontologies and statistical-based semantic relatedness measures for deriving the semantic aspects of the user's query. The exploited ontologies play an important role in deriving the query's semantics by constructing semantic networks that define and relate the query's terms by different types of taxonomic and semantic relations. Since we are exploiting multiple ontologies, multiple heterogeneous semantic networks may be produced. To address this problem, we employ semantic networks merging techniques to combine the produced networks into a single coherent network. This represents a cooperative decision made by multiple ontologies on the query's semantics.

Although using multiple ontologies ensures broader and deeper domain coverage, we may still have missing background knowledge such as concepts, relations or instances in the exploited ontologies. To compensate for this knowledge, we utilize statistical-based semantic relatedness measures. These measures are employed to enrich the query's semantics with additional concepts or instances that don't exist in any of the used ontologies. The presented approach has two major contributions.

- Introducing a fully-automatic semantic/statistical based technique for deriving the semantic aspects of the user's query.
- Employing a domain-independent query processing mechanism that allows users to search for information in generic domains.

The rest of this paper is organized as follows. Section 2 presents a discussion on the role of ontologies in bridging the semantic gap between users' queries and keywords used to index Web Pages. In Section 3, we describe the architecture of the proposed system. Section 4 details the algorithms and techniques used in the proposed system. Section 5 presents the experimental results. In Section 6 we discuss the conclusions and outline future work.

2 Related Work

Ontologies play a key role in deriving the semantic aspects of users' queries. They provide machine-processable and explicit specifications of conceptualizations that allow for interpreting the explicit and implicit semantics of natural language queries. In recent years, several Ontology-Based Query Processing (OBQP) approaches have been proposed, but all of them either use a single ontology or multiple ontologies for a specific domain. For instance, the system proposed by the authors of [2] maps the query's keywords to corresponding WordNet synsets. Although this system is able to discover relations between keywords, these relations are subjected to the limitations of WordNet, namely, limited number of semantic and taxonomic relations and limited domain coverage. Lei et al. propose to interpret keywords as instances, concepts or properties, respectively [3]. The assumption is that keywords represent entities in the ontology and can be connected through direct relations defined in it. While the authors claim to be able to discover relations and handle simple and complex queries, it is not clear how this is achieved, especially given the fact that existing ontologies suffer from missing background knowledge problem. A similar approach has been proposed by the authors of [4]. The main difference lays in the way queries are computed. A parameter d can be set and configured by the users to expand the queries graphical representation by considering neighboring entities. Recently, there has been attempt to use multiple ontologies in specialized domains such as, universities domain [5]. In this work, experimental results show that by using multiple ontologies precision can be improved. In our approach, our interest is not limited to a particular domain; therefore, we propose to use large-scale general-purpose ontologies that cover knowledge in multiple domains. In addition, we utilize statistical-based semantic relatedness measures to tackle the missing background knowledge problem.

3 General Overview

In this section, we present an approach for automatic query interpretation using multiple ontologies. The intuitions behind using multiple ontologies are: *i)* they make cooperative semantics-based query interpretation decisions and *ii)* using multiple ontologies provides broader and deeper domain covering and knowledge representation.

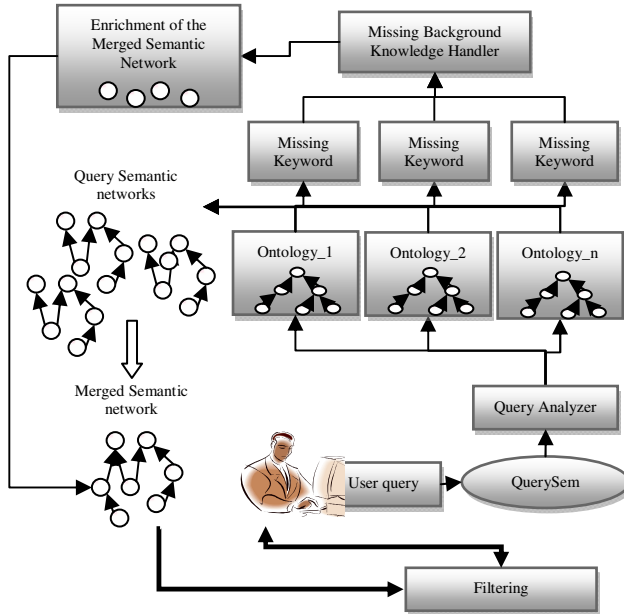


Fig. 1. General Architecture of the Proposed System (QuerySem)

As shown in Figure 1, when a user submits a query, the query analyzer first maps the query's keywords to corresponding entities in each of the exploited ontologies. It mainly checks whether each of the keywords is defined in any of the ontologies or not. From the set of keywords that are defined in the ontologies, semantic networks are automatically and incrementally constructed. At this step, an ontology may produce zero, one or more semantic networks. Therefore, a merging mechanism is required to merge the produced semantic networks into a single coherent network. The merged semantic network represents a cooperative decision made by multiple ontologies on the semantics of the user's query.

While using multiple ontologies provides broader and deeper domain coverage, we may still have some query keywords that are not defined in any of the used ontologies. In this case, the missing background knowledge handler is utilized to measure the semantic relatedness between the keywords that are missing from the ontologies and those that are defined in them. By utilizing this technique, an additional set of keywords is suggested to enrich the merged semantic network.

Accordingly, the hierarchy of the merged semantic network is updated and returned to the user. In our approach, users are provided with a decision-oriented mechanism that allows them to filter the returned semantic network by removing nodes that are not strongly related to their information needs.

4 Theoretical Basis

Ontology components (concepts, properties, and individuals) represent the reference to which the query's keywords are mapped. It is clear that the precision of the mapping process highly depends on the knowledge represented by the used ontology. Before we detail the techniques of the proposed system, we formalize the use of the terms "Ontology", "Semantic Network", "Merging Semantic Networks", "Semantic Network Enrichment", and "Normalized Retrieval Distance (NRD)":

Definition 1: Ontology: An ontology Ω is *quintuple*, $\Omega := \langle C, P, I, V, A \rangle$ where:

- C is the set of concepts of the ontology. The concept hierarchy of Ω is a pair (C, \leq) , where \leq is an order relation on $C \times C$. We call $c \in C$ the set of concepts, and \leq the sub-concept relation.
- P is the set of properties.
- I is the set of individuals
- V is the set of property values
- A is the set of axioms (such as constraints)

The multiple ontology-based query processing module takes a given user's query Q_U as input and produces a set of semantic networks $S_\zeta = \{\zeta_1, \zeta_2, \zeta_3, \zeta_n\}$ as output. These networks are automatically constructed based on the set of exploited ontologies $S_\Omega = \{\Omega_1, \Omega_2, \Omega_3, \Omega_n\}$. A semantic network can be formally presented as follows:

Definition 2: Semantic Network: A semantic network $\zeta := \langle K, R, A \rangle$ where:

- K is the set of keywords in the semantic network. These keywords are the query's keywords that are defined in the ontologies.
- R is the set of relations between K . These relations are derived from the ontologies.
- A is the set of axioms defined on K and R according to S_Ω .

As we are exploiting more than one ontology, the probability that the produced semantic networks are semantically heterogeneous (i.e. there are differences in the conceptual and terminological representations of the produced semantic networks) is very high. Therefore, we need to merge them into a single coherent semantic network. The merging algorithm can be defined as follows:

Definition 3: Merging Semantic Networks: A semantic network merging algorithm takes a given set of heterogeneous semantic networks $S_\zeta = \{\zeta_1, \zeta_2, \zeta_3, \zeta_n\}$ as input and produces a single merged semantic network ζ_{merged} as output.

As we discussed in section 3, although we are using multiple ontologies, we may still face the missing background knowledge problem. This means that a set of the query's keywords $S_{missing}$ will not be considered within the hierarchy of ζ_{merged} . However, we believe that ζ_{merged} can be further enriched with some keywords from $S_{missing}$. This can be automatically achieved by measuring the strength of the semantic relatedness between the keywords of both $S_{missing}$ and ζ_{merged} . Formally, this can be presented as follows:

Definition 4: Semantic Network Enrichment: A semantic network enrichment algorithm takes a given set of query's keywords that are not defined in the ontology $S_{missing} = \{w_1, w_2, w_3, w_n\}$ and the merged semantic network ζ_{merged} as input and produces for each $k \in K$ in ζ_{merged} a set of $S(k) \subseteq S_{missing}$ as output. where,

- $S(k)$ is the set of suggested enrichment candidates for k . A suggested candidate $w \in S_{missing}$ is a word or compound word from $S_{missing}$.

The set of suggested enrichment candidates $S(k)$ can be obtained using the Normalized Retrieval Distance (NRD) function and based on a threshold value v using Equation 1.

$$S(k,v) := \{ w \in S_{missing} \mid \text{NRD}(k,w) \geq v \} \quad (1)$$

Definition 5: Normalized Retrieval Distance (NRD): is an adopted form of the Normalized Google Distance (NGD) [6] function that measure the semantic relatedness between pairs of terms: Given two terms T_{mis} and T_{in} the Normalized Retrieval Distance between T_{mis} and T_{in} can be obtained as follows:

$$\text{NRD}(T_{mis}, T_{in}) = \frac{\max\{\log f(T_{mis}), \log f(T_{in})\} - \log f(T_{mis}, T_{in})}{\log M - \min\{\log f(T_{mis}), \log f(T_{in})\}} \quad (2)$$

where:

- T_{mis} is a term that is not defined in the ontology
- T_{in} is a term that exists in the ontology
- $f(T_{mis})$ is the number of hits for the search term T_{mis}
- $f(T_{in})$ is the number of hits for the search term T_{in}
- $f(T_{mis}, T_{in})$ is the number of hits for the search terms T_{mis}, T_{in}
- M is the number of web pages indexed by the search engine

Because the NRD function only measures the strength of the semantic relatedness, further processing is required to derive the actual semantic relation(s) that may hold between strongly related terms. This step is detailed in section 4.1.3.

4.1 From Keywords to Semantic Networks

In the following sections, we present the details of the techniques that we employ in our system.

4.1.1 Multiple Ontology-Based Query Analysis

First, we apply the following Natural Language Processing (NLP) steps on the user's query:

1. Stopword removal: some of the words, for example, a, an, the, of,... occur frequently in queries without semantic significance. Those words are removed based on a pre-defined list that includes them.
2. n-gram query tokenization: the query is split into tokens of lengths from 1-3.
3. Part-of-speech tagging: each token is tagged by the grammatical category that it belongs to such as noun, verb, etc.

After the NLP step, query tokens are submitted to each of the ontologies to check whether they are defined in any of them or not. Tokens that are defined in the ontologies are considered as meaningful query keywords and thus, semantic networks that represent these keywords and the semantic relations that hold between them are automatically constructed. As a consequence of this step, different number of semantic networks may be produced according to different ontologies. Therefore, we utilize the semantic networks merging algorithm to merge these networks into a single semantic network. The next example illustrates the NLP steps and the semantic networks construction and merging techniques.

Example 1: Query = “Java or jawa the island of Indonesia”

In this example, we use WordNet [7] and OpenCyc [8] general-purpose ontologies. First, the stopword removal function removes stopwords based on a pre-defined list. For example, the words (the, an, a) and characters such as (,) , : , are removed from the query. Next, the n-gram tokenization algorithm tokenizes the query into unigram, bigram and trigram tokens. After this step, each token is submitted to each of the ontologies to find whether it is defined in it or not. For instance, when we explore the hierarchy of WordNet ontology, we will find that the query's keyword “Java” has three different senses as follows:

1. Java -- (an island in Indonesia south of Borneo; one of the world's most densely populated regions)
2. coffee, java -- (a beverage consisting of an infusion of ground coffee beans; "he ordered a cup of coffee")
3. Java -- (a simple platform-independent object-oriented programming language used for writing applets that are downloaded from the World Wide Web by a client and run on the client's machine)

Although this keyword has three different senses (i.e. meanings), only the first sense (sense 1) is included in the produced semantic network and the other senses are excluded. This is because the other senses (2 and 3) are not semantically related to the other query's keywords “Indonesia” and “Island”. Accordingly, automatic disambiguation of the query's keywords is performed, producing a semantic network that includes only the semantically related keywords. In addition, the synonyms of the disambiguated keywords are automatically included in the produced semantic

network. For instance, from Example 1, we find that the set of keywords {Java, Island, Indonesia} exist in both WordNet and OpenCyc ontologies. Form this set, semantic networks are constructed based on both ontologies as shown in Figure 2.

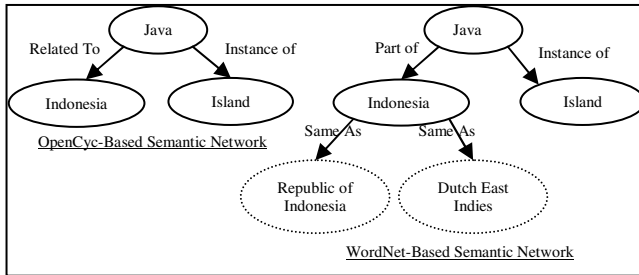


Fig. 2. Query's Keywords Represented by Semantic Networks

As we can see from Figure 2, the exploited ontologies produced heterogeneous semantic networks. These semantic networks represent different perspectives. For example, according to OpenCyc ontology, the relation between the query's keywords "Java" and "Indonesia" is "Related To". While according to WordNet ontology the relation between these keywords is "Part of". Therefore, to solve the semantic heterogeneity problem between the produced semantic networks, we utilize the merging algorithm described in section 4. In this algorithm, we use the merging techniques proposed in our previous work [9]. These techniques are prioritized according to their significance and execution into semantic, string, and statistical based merging techniques respectively. The result of merging the semantic networks is shown in Figure 3. The rest of n-gram tokens that are not defined in any of the exploited ontologies such as the token "jawa" are considered as missing background knowledge and submitted to the NRD function for further processing.

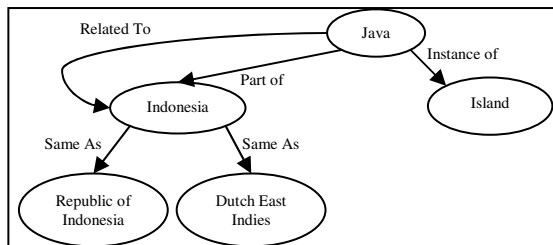


Fig. 3. Merged Query's Semantic Network

4.1.2 Statistical-Based Semantic Relatedness Measures

We utilize statistical-based semantic relatedness measures to compensate for the lack of domain coverage in the used ontologies. For query's keywords that are not defined in the used ontologies, we attempt to find whether they can be suggested as candidates to enrich the merged query's semantic network. To do this, first we utilize the NRD

function detailed in section 4. This function measures the semantic relatedness between the query's keywords that already exist in the merged semantic network and other keywords that are not defined in any of the exploited ontologies. As different semantic relatedness measures are returned according to several search engines (Google, AltaVista, Yahoo!), we sum up all NRD values for each candidate keyword. This summation represents a cooperative decision made by several search engines on the semantic relatedness measurers. Algorithm 1 explains the procedure of acquiring the semantic relatedness measures.

Algorithm 1. Normalized Retrieval Distance Function

Input: Keywords that are missing from the ontologies, $K_{missing}$, Keywords of the merged semantic network, K_{in}

Output: semantic relatedness measures between $K_{missing}$ and K_{in}

```

1:  int[] result;
2:  for each  $k_{missing} \in K_{missing}$ 
3:    for each  $k_{in} \in K_{in}$ 
4:      result=  $NRD(k_{missing}, k_{in})$ 
5:    end for
6:  end for

```

The NRD function takes as input pairs of keywords and produces as output the measures of semantic relatedness between them. Table 1 shows the semantic relatedness measures between the keyword “jawa” and the rest of the query's keywords: {Java, Island, Indonesia}.

Table 1. Semantic Relatedness Measures for the Keyword “Jawa”

<i>Keyword</i> \ <i>Keyword</i>	Java	island	Indonesia
Jawa	0.72	0.56	0.69

4.1.3 Semantic Relations Extraction

Obtaining semantic relatedness measures by utilizing the NRD function is a prior step towards deriving the actual semantic relation(s) that may hold between semantically related keywords. Therefore, in order to derive these relations, we defined a list of lexico-syntactic patterns to derive relations such as, synonymy (e.g. “X is same as Y”), hypernymy/hyponymy (e.g. “X is a(n) Y”, “Y is a(n) X”) and meronymy (e.g. “X is part of Y”). These types of relations can be automatically obtained by utilizing the Semantic Relation Extractor (SRE) function. For each pair of semantically related keywords, the SRE returns the number of their hits by submitting them including each of the patterns to several search engines. Then, relations between keyword pairs are suggested based on the highest values returned. Algorithm 2 shows the steps of the SRE function.

Algorithm 2. Semantic Relation Extractor Function**Input:** Semantically related keywords, ($K_{missing}, K_{in}$)**Output:** suggested relations between keywords

```

1:  String [] suggestedRelations
2:  String [] Patterns;
3:  int[] value;
4:  for each  $k_{missing} \in K_{missing}$ 
5:    for each  $k_{in} \in K_{in}$ 
6:      for each pattern  $p$  in Patterns
7:        value.add(makeQuery(“ $k_{missing}$ ”,  $p$ , “ $k_{in}$ ”));
8:      end for
9:      suggestedRelations.add(max(value));
10:    end for
11:  end for

```

The function takes pairs of keywords with strong semantic relatedness measures as input and produces the suggested semantic relation between them according to the lexico-semantic patterns as output. For each pattern, the makeQuery function (Line 7) submits exact match queries including both keywords. We considered both singular and plural forms of the keywords. Patterns that include negation operators such as “No $K_{missing}$ is a(n) K_{in} ” were excluded.

For instance, based on the obtained results by the NRD function, we were able to find that “jawa” is semantically related to “island”. But, we still don’t know what type of semantic relation may exist between them. Therefore, we utilize the SRE function by submitting patterns in the form of exact match queries Q_i such as:

- Q_1 =“jawa is an island”, which outputs 80,700 hits result
- Q_2 =“jawa is a part of island”, which outputs 0 hits result
- Q_3 =“jawa is same as island”, which outputs 0 hits result

Based on the number of hits returned by the search engines for the queries Q_i , relations defined in the patterns are suggested to be used to enrich the merged semantic network with the keyword “jawa”.

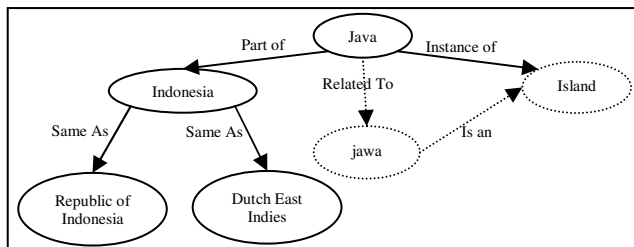


Fig. 4. Enrichment of the Merged Semantic Network

We notice that for some of the semantically related keywords, no results are returned. However, we know that they are semantically related. Therefore, in this case, we define the “Related To” relation between them and use it to enrich the merged semantic network. For instance, as shown in Figure 4, we are able to enrich the merged semantic network with the keyword “jawa” by using the “Related To” relation.

5 Experimental Results

This section describes the experiments carried out to evaluate the techniques of the proposed system. All solutions are implemented in Java and experiments are performed on a PC with dual-core CPU (3000GHz) and 2 GB RAM. The operating system is OpenSuse 11.1. We carried out experiments using WordNet [7], OpenCyc [8], and YAGO [10] ontologies. Details of these ontologies are listed below:

1. **WordNet:** it is a generic manually-created ontology that covers knowledge in multiple domains. It organizes the concepts into synsets, where each synset contains synonym concepts like {writer and author}. These synsets are connected through semantic relations such as hypernymy, hyponymy, meronymy etc. This organization of WordNet synsets is useful in discovering semantic relations between the words in the user’s query since we can map them to their corresponding senses in WordNet..
2. **Opencyc:** is the open source version of the Cyc technology, the world's largest and most complete general knowledge base and commonsense reasoning engine. The current release of OpenCyc includes the entire Cyc ontology containing hundreds of thousands of terms, along with millions of assertions relating the terms to each other, forming an ontology whose domain is all of human consensus reality. In addition to more than 100,000 "broaderTerm" assertions, added to the previous generalization (subclass) and instance (member) assertions, to capture additional relations among concepts. It also includes new links between Cyc concepts (including predicates) and the FOAF ontology. And new links between Cyc concepts and Wikipedia articles.
3. **YAGO:** is an automatically constructed ontology that knows 2 million entities such as people, cities, movies, and historical events. It also knows more than 19 million facts about these entities. The knowledge in YAGO has been extracted from Wikipedia and combined with the taxonomic structure of WordNet ontology.

In order to testify our proposal of using multiple ontologies, we selected 75 sample queries (15 per domain) from different domains. As an evaluation methodology, we used the gold standard approach, which is widely used in the Information Retrieval field. Accordingly, Precision measure was computed against a gold standard (relevance judgments on the Query-Results - Qrels) when using a single ontology vs. multiple ontologies. As shown in Table 2, higher precision is obtained when using multiple ontologies. The reason is because by employing lexical and semantic knowledge represented by multiple ontologies, query’s keywords are automatically disambiguated according to their definitions in the exploited ontologies. In addition, a

set of statistical-based semantically related keywords are suggested to enrich the query's semantics. Therefore, our approach introduces an added value with respect to traditional keyword-based indexing approaches as the search is conducted at a semantic level, and thus, leading to more precise results.

Table 2. Precision Using a Single vs. Multiple Ontologies

Queries per Domain	Precision Using a Single Ontology (WordNet)	Precision Using Multiple Ontologies (WordNet, OpenCyc, Yago)
Countries and Cities	44%	82%
Sports	40%	75%
Programming Languages	42%	71%
Universities	47%	68%
People	52%	73%

6 Conclusion and Future Work

In this paper, we proposed an approach to combine semantic and statistical relatedness measures to derive the semantic aspects of users' queries. In our approach, we exploit knowledge represented by multiple large-scale general-purpose ontologies to construct semantic networks that relate the query's keywords by different types of semantic and taxonomic relations. To merge the heterogeneous semantic networks, we utilized semantic, string, and statistical based merging techniques that we used in our previous work. To address the issue of missing background knowledge in the exploited ontologies, we employed the NRD and the SRE functions. Both functions work cooperatively to enrich the merged semantic network with additional candidate keywords. Experimental results shown that the proposed approach is effective in deriving the query semantics and that the precision of the results increased when using multiple ontologies.

In the future work, we plan to exploit additional ontologies and perform more experimentation to verify that plugging-in additional ontologies to the current system's prototype will significantly improve its precision. Other future work also includes experimenting on the users' behavior towards adopting the proposed query processing system. This step is very important because if users adopt the semantic-based search strategy, a new generation of search engines that can better understand users' queries would be developed. However, this will also require hard efforts from the ontology engineering community to build more sophisticated, up-to-date and easy-to-maintain ontologies that can be easily plugged-in to the search engines.

References

1. Tanaka, K., et al.: Improving Search and Information Creditability Analysis from Interaction between Web1.0 and Web2.0 Content. *Journal of Software* 5(2), 154–159 (2010)

2. Royo, J., Mena, E., Bernard, J., Illarramendi, A.: Searching the web: From keywords to semantic queries. In: Proceedings of the Third International Conference on Information Technology and Applications (ICITA 2005), pp. 244–249. IEEE Computer Society Press, CA (2005)
3. Lei, Y., Uren, V., Motta, E.: Semsearch: A search engine for the semantic web. In: Staab, S., Svátek, V. (eds.) EKAW 2006. LNCS (LNAI), vol. 4248, pp. 238–245. Springer, Heidelberg (2006)
4. Tran, T., Cimiano, P., Rudolph, S., Studer, R.: Ontology-Based Interpretation of Keywords for Semantic Search. 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. 523–536. Springer, Heidelberg (2007)
5. Wimalasuriya, D., Dou, D.: Using Multiple Ontologies in Information Extraction. In: CIKM 2009, Hong Kong, China, pp. 235–244 (2009)
6. Cilibrasi, R., Vitanyi, P.: The Google Similarity Distance. *IEEE Transactions on Knowledge and Data Engineering* 19(3), 370–383 (2007)
7. Miller, G.A.: WordNet: A Lexical Database for English. *Communications of the ACM*, 409–409 (1995)
8. Matuszek, C., Cabral, J., Witbrock, M., DeOliveira, J.: An Introduction to the Syntax and Content of Cyc. In: AAI Spring Symposium (2006)
9. Maree, M., Belkhatir, M.: A Coupled Statistical/Semantic Framework for Merging Heterogeneous Domain-Specific Ontologies. In: The 22nd International Conference on Tools with Artificial Intelligence (ICTAI 2010), Arras, France, pp. 159–166 (2010)
10. Fabian, M.S., Gjergji, K., Gerhard, W.: YAGO: A Core of Semantic Knowledge Unifying WordNet and Wikipedia. In: 16th International World Wide Web Conference, WWW, pp. 697–706 (2007)

Getting Critical Categories of a Data Set

Cheqing Jin, Yizhen Zhang, and Aoying Zhou

Shanghai Key Laboratory of Trustworthy Computing,
Software Engineering Institute, East China Normal University, China
{cqjin,ayzhou}@sei.ecnu.edu.cn, 51091500017@ecnu.cn

Abstract. Ranking query that is widely used in various applications is a fundamental kind of queries in the database management field. Although most of the existing work on ranking query focuses on getting top- k high-score tuples from a data set, this paper focuses on getting top- k critical categories from a data set, where each category is a data item in the nominal attribute or a combination of data items from more than one nominal attribute. To describe each category precisely, we use a data distribution that comes from the score attribute to represent each category, so that the set consisting of all categories can be treated as a probabilistic data set. In this paper, we devise a novel method to handle this issue. Analysis in theorem and experimental results show the effectiveness and efficiency of the proposed method.

Keywords: critical category, ranking query, possible world.

1 Introduction

Ranking query, which returns top- k tuples from a data set with large scores computed by a scoring function, is one of the most fundamental queries in the database management field. Ilyas et al. have summarized important studies in this area [7]. In this paper, we study how to get top- k categories from a typical data set containing tens of nominal attributes and one numeric attribute. The nominal attributes describe objects' features and the numeric attribute describes the target score. Although most of the existing work focuses on selecting top- k tuples from a data set, our focus is to find k most critical categories from a data set, where each category is a data item in the nominal attribute or a combination of data items from more than one nominal attribute. In general, one category may occur at multiple tuples. For example, consider an *employee* set with two nominal attributes, *sex* and *working-years*, and one numeric attribute, *salary*. Typical categories include: the male, the female, working less than 5 years, and working more than 5 years. It is interesting to discover which kind of employees have the highest salary. A naive way for this query is to return a category with the maximum average salary. However, since the average value is sensitive to some outlier records, it is better to use a score distribution to describe each category. Let's see an example below.

Table 1 lists the achievements of four students who attend a university competition. Each record has three nominal attributes (age, sex and university) and

a numeric attribute for his (/her) final score. It is easy to claim age-21 group behaves better than age-20 group because the score of an arbitrary student in the former group is higher than that of any student in the latter group. But it is not easy to claim whether the female students behave better or the male students behave better, since the female group has higher average score ($\frac{80+80+100}{3} > 85$), while the male group has lower average ranking value ($2 < \frac{1+3+4}{3}$). Obviously, the average score of the female students is highly influenced by the “outlier” student (the fourth record).

Table 1. The achievements of four students

User ID	Age	Sex	University	Score
1	20	Female	Stanford	80
2	21	Male	Stanford	85
3	20	Female	MIT	80
4	21	Female	MIT	100

Naturally, the data distribution is a better way to describe a category without any loss of information. The data distribution is in a format of $\{(v_i, p_i)\}$, where v_i is the value of the i th item and p_i is its probability. $\sum_i p_i = 1$. Table 2 lists the score distributions of six categories of the data set in Table 1. For example, the score distribution of c_3 is $\{(80, 0.67), (100, 0.33)\}$, because two female students are scored 80 and the other one is scored 100. In fact, each category in Table 2 can also be viewed as a probabilistic tuple, so that the whole table becomes a probabilistic data set. Our task changes to find the top- k uncertain tuples upon a probabilistic data set. There are two kinds of uncertainties in a probabilistic data set, including *existential* uncertainty and *attribute-level* uncertainty. The *existential* uncertainty uses a confidence value to describe the existence of a tuple, and the *attribute-level* uncertainty describes the imprecision of an attribute. In this paper, the probabilistic data set for the categories only has *attribute-level* uncertainty.

The possible world model is widely adopted by the probabilistic database management field. There are a huge number of possible worlds in the possible world space. In each possible world, each tuple (in this case, each tuple refers to a category) has an explicit value. The probability of each possible world is the

Table 2. The score distribution of each category

Category ID	Category Name	Score distribution
c_1	Age = 20	$\{(80, 1.0)\}$
c_2	Age = 21	$\{(85, 0.5), (100, 0.5)\}$
c_3	Sex = Female	$\{(80, 0.67), (100, 0.33)\}$
c_4	Sex = Male	$(85, 1.0)$
c_5	Univ. = Stanford	$\{(80, 0.5), (85, 0.5)\}$
c_6	Univ. = MIT	$\{(80, 0.5), (100, 0.5)\}$

Table 3. Possible worlds for Table 2

PW	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}	w_{11}	w_{12}	w_{13}	w_{14}	w_{15}	w_{16}
c_1	80	80	80	80	80	80	80	80	80	80	80	80	80	80	80	80
c_2	85	85	85	85	85	85	85	85	100	100	100	100	100	100	100	100
c_3	80	80	80	80	100	100	100	100	80	80	80	80	100	100	100	100
c_4	85	85	85	85	85	85	85	85	85	85	85	85	85	85	85	85
c_5	80	80	85	85	80	80	85	85	80	80	85	85	80	80	85	85
c_6	80	100	80	100	80	100	80	100	80	100	80	100	80	100	80	100
Prob. ($\times 10^{-2}$)	8.4	8.4	8.4	8.4	4.1	4.1	4.1	4.1	8.4	8.4	8.4	8.4	4.1	4.1	4.1	4.1

product of the occurring probability of all tuples in the world. The sum of the probabilities of all possible worlds is equal to 1. Table 3 illustrates all 16 possible worlds of the probabilistic data set in Table 2. For example, the probability of the ninth possible world, w_9 , is $0.084 (= 1.0 \times 0.5 \times 0.33 \times 1.0 \times 0.5 \times 0.5)$.

Recently, many uncertain top- k queries have been proposed to meet different application requirements, including U-Top k , U k -ranks, PT- k , ER-top k , and so on. As reported in 3, ER-Top k query, which returns k tuples with the smallest expected ranks, satisfies a number of key properties at the same time, while the rest semantics cannot. These properties include *exact- k* , *containment*, *unique-rank*, *value-invariance* and *stability*. Consequently, we use ER-top k query to rank each tuple in a probabilistic data set. For any tuple c , its expected rank, $r(c)$, refers to the expected number of tuples greater than c in all possible worlds. Let's take c_2 as an instance. Only one tuple is greater than c_2 in w_2 , w_4 , w_5 and w_7 , and merely two tuples are greater than c_2 in w_6 and w_8 . In the rest possible worlds, none is greater than c_2 . Thus, $r(c_2) = 0.25 \times 1 + 0.082 \times 2 = 0.414$.

We also study the situation where each category consists of items coming from more than one nominal attribute. Table 4 illustrates in total ten such categories based on the dataset in Table 1. Lemma 1 shows the space consumption of the probabilistic data set generated in this case.

Table 4. The score distributions of the categories with two nominal attributes

Category ID	Category Name	Score distribution
c_7	Age = 20, Sex=Female	$\{(80, 1.0)\}$
c_8	Age = 21, Sex=Male	$\{(85, 1.0)\}$
c_9	Age = 21, Sex=Female	$\{(100, 1.0)\}$
c_{10}	Age = 20, Univ.=MIT	$\{(80, 1.0)\}$
c_{11}	Age = 20, Univ.=Stanford	$\{(80, 1.0)\}$
c_{12}	Age = 21, Univ.=MIT	$\{(100, 1.0)\}$
c_{13}	Age = 21, Univ.=Stanford	$\{(85, 1.0)\}$
c_{14}	Sex = Male, Univ.=Stanford	$(85, 1.0)$
c_{15}	Sex = Female, Univ. = Stanford	$\{(80, 1.0)\}$
c_{16}	Sex = Female, Univ. =MIT	$\{(80, 0.5), (100, 0.5)\}$

Lemma 1. *Let D denote a deterministic data set that has d nominal attributes, $|D|$ the number of tuples in D , and l the number of attributes per category. The space consumption of the probabilistic data set for D is $O(\binom{d}{l} \cdot |D|)$.*

Proof. Each tuple in D is associated with $\binom{d}{l}$ different categories. In other words, the numeric value of each tuple must be stored $\binom{d}{l}$ times. Consequently, the space consumption of the probabilistic data set is $O(\binom{d}{l} \cdot |D|)$.

The top- k categories can be computed straightforwardly by expanding all of the possible worlds. However, this method is infeasible in real applications since the number of the possible worlds will grow up exponentially to the size of a probabilistic data set. In this paper, we propose a new method to handle this issue, which avoids expanding the possible world space.

We sketch our method briefly. First, we scan the deterministic data set D to generate all categories, along with all the tuples associated with each category. Second, we scan D again to construct a global function $q(v)$, which is critical to compute the expected rank for each tuple. The function $q(v)$ returns the sum of the probabilities of all tuples greater than v . Finally, we compute the expected value for each tuple, based on which top- k tuples are returned.

The executing performance can be further improved by estimating the expected rank of each category. The cost on estimating a tuple is significantly lower than computing its exact value. With the help of the estimating technique, it only requires to compute the exact expected ranks for a small part of tuples.

1.1 Contributions

The main contributions of this paper are summarized below.

1. Although most of the existing work on ranking query focuses on finding top- k tuples from a data set, our focus is to get top- k categories from a data set. Each category is described by a score distribution, not the simple average value.
2. After defining the TkCl query formally, we also propose novel methods to handle this query.
3. Experimental reports evaluate the effectiveness and efficiency of the proposed methods.

The rest of the paper is organized below. In Section 2, we define the problem formally. In Section 3, we describe our novel solution in detail. We report the experimental results in Section 4. Finally, we review the related work and conclude the paper in brief in the last two sections.

2 Problem Definition

We define the problem formally in this section. Consider a data set D containing n tuples, $t_1 \cdots, t_n$, where each tuple has d nominal attributes, A_1, \cdots, A_d ,

and one numeric target attribute A_{tar} . Each nominal attribute (say, A_i) has g_i distinct items, denoted as $a_{i,1}, \dots, a_{i,g_i}$. The A_{tar} value of t_i is $a_{i,tar}$.

Definition 1 (Category- l). A Category- l instance refers to a combination of items coming from l nominal attributes¹.

For example, there are six category-1 instances in Table 2 and ten category-2 instances in Table 4. When a category c occurs at the tuple t_i , we call $c @ t_i$. Because a category may occur at more than one tuple in D , it is natural to use a discrete probability distribution function to describe its A_{tar} attribute, as shown in Definition 2.

Definition 2 (The Data Distribution of a Category- l Instance c , $f(c)$). Let D_c denote a set containing all A_{tar} values where c occurs, i.e., $D_c = \{a_{i,tar} \mid c @ t_i, t_i \in D\}$. \hat{D}_c is a set containing all distinct items in D_c . $\forall v \in \hat{D}_c$, $Pr[f(c) = v] = h(v)/|D_c|$, where $h(v)$ is the frequency of v in D_c .

Let U denote a set containing all category- l instances when given an l . In general, the i th category, c_i , is described as $\{(v_{i,j}, p_{i,j})\}$, where it has a value of $v_{i,j}$ with probability $p_{i,j}$. For any i , $\sum_j p_{i,j} = 1$. Obviously, U is a probabilistic data set with only attribute-level uncertainty.

Definition 3 (The Top- k Category l query, TkCl). The TkCl query returns k categories from U with the smallest values of $r(c)$, which is defined in Equ. (1).

$$r(c) \triangleq \sum_{w \in \mathcal{W}} Pr[w] \cdot rank_w(c) \quad (1)$$

where \mathcal{W} is the possible world space of U , $Pr[w]$ is the probability of a possible world instance w , and $rank_w(c)$ returns the rank of t in w . In other words, if $c \in w$, $rank_w(c)$ returns the number of tuples ranked higher than c . Otherwise, it returns the number of tuples in w ($|w|$).

The definition of TkCl utilizes the ER-top k semantics in [3]. The expected rank of each tuple can be computed efficiently if a global function $q(v)$ for the whole data set U and some local functions (say, $q_i(v)$) for each tuple (say, t_i) in U are ready. For each c_i , the local function $q_i(v)$ describes the sum of probabilities of the tuple c_i greater than v , i.e., $q_i(v) = \sum_{j, v_{i,j} > v} p_{i,j}$. The global function $q(v)$ describes the sum of probabilities of all tuples in U greater than v , i.e., $q(v) = \sum_i q_i(v)$. According to the definition and linearity of expectation, The expected rank of c_i , $r(c_i)$, is computed by Equ. (2).

$$r(c_i) = \sum_j p_{i,j} (q(v_{i,j}) - q_i(v_{i,j})) \quad (2)$$

The goal of this paper is to handle the TkCl query efficiently. Our first attempt is to construct the functions $q(v)$ and $q_i(v)$ so that the expected rank can be computed through Equ. (2). Our second attempt is to estimate the expected rank of each tuple (say, t_i) without knowing the function $q_i(v)$ in advance, so that the time cost is significantly reduced.

¹ We also call a category- l instance a ‘‘category’’ if the context is explicit.

Algorithm 1. basicTkCl(D, k, l)

```

1:  $U \leftarrow \emptyset$ ;
2: for each tuple  $t_i$  in  $D$ 
3:   for each category  $c$  generated from  $t_i$ 
4:     if ( $c \notin U$ ) then  $U \leftarrow U \cup \{c\}$ ;
5:     Add  $a_{i,tar}$  to  $D_c$ , the set affiliated to  $c$ ;
6:   end for
7: end for
8: Scan  $U$  to build a probTree to describe the function  $q(v)$ ;
9: for each  $c$  in  $U$ 
10:  Compute its expected rank,  $r(c)$ , by using Equ. (2);
11: end for
12: Return top- $k$  tuples with the smallest expected ranks;

```

3 Our Solution

We introduce our method to process a TkClquery in this section. We give a basic solution in Sec. 3.1. Subsequently, an improved solution, which supports a larger data set, is introduced in Sec. 3.2.

3.1 A Basic Solution

Algorithm basicTkCl (Algorithm 1) illustrates a basic method to handle the TkCl query. It accepts three parameters, D , k and l , where D is an input deterministic data set, k and l are the parameters for a TkCl query. The set U is used to store all categories. This algorithm scans D once and U twice.

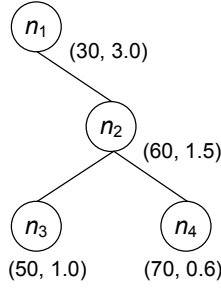
First, we scan the input database D to generate all of categories. Each t_i in D refers to $\binom{d}{l}$ different categories, where d is the number of nominal attributes in D . Any category that doesn't belong to U is inserted into U . Each category c in U also has a set D_c that stores all A_{tar} values for this category. Thus, the value $a_{i,tar}$ is added to each category's D_c set (at Lines 2-7). To reduce the processing cost, we use a binary searching tree to implement U .

Second, we scan the data set U to build a *probTree* to describe the function $q(v)$ (at Line 8). The *probTree*, a variant of binary searching tree, is efficient to describe the function $q(v)$ [8]. We review it briefly with an example. Figure 1 illustrates the states of a *probTree* for a probabilistic data set. The input data for a *probTree* contains all pairs like $(v_{i,j}, p_{i,j})$ in an uncertain data set. Each node in the *probTree* has two fields, v and p . For any pair $(v_{i,j}, p_{i,j})$, it will try to find a node e with $e.v = v_{i,j}$. A new node is created in *probTree* if no such e exists. The field p represents the sum of probabilities of all input items locating at the right-side of e 's subtree. $\forall v$, $q(v)$ can be computed efficiently. For example, $q(55) = n_2.p = 1.5$, $q(10) = n_1.p + n_6.p = 3.4$.

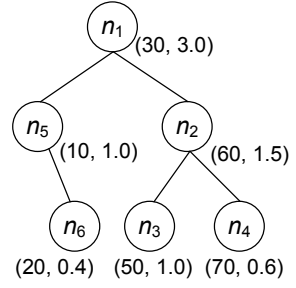
Finally, we scan the probabilistic data set U again to compute the expected ranks of all categories by using Equ. (2). For each category c , we need to generate the local function $q_i(c)$ based on its D_c set. In general, it can be done by sorting

tuple	Attribute value
t_1	$\{(30,0.5), (60, 0.5)\}$
t_2	$\{(50, 1.0)\}$
t_3	$\{(60,0.4), (70, 0.6)\}$
t_4	$\{(10,0.6), (20, 0.4)\}$

(a) a probabilistic data set



(b) a probTree for $t_1 - t_3$



(c) a probTree for $t_1 - t_4$

Fig. 1. An example of *probTree*

all items in D_c in descending order and then scan it one by one, which costs $O(|D_c| \cdot \log |D_c|)$.

Theorem 1. *The space consumption of Algorithm basicTkCl is $O(\binom{d}{l} \cdot |D|)$. The time cost is $O(\binom{d}{l} \cdot |D| \cdot \max(\log |D|, \sum_{1 \leq i_1 < \dots < i_j \leq d} \prod_{1 \leq j \leq l} g_{i_j}))$.*

Proof. The main space consumption is U , which stores all categories and score distributions. According to Lemma 1, the space consumption is $O(\binom{d}{l} \cdot |D|)$.

The time complexity of this algorithm consists of three parts. First, it scans D to generate all categories. Each tuple in D affects $\binom{d}{l}$ categories. The total number of category- l instances is $O(\sum_{1 \leq i_1 < \dots < i_j \leq d} \prod_{1 \leq j \leq l} g_{i_j})$, where g_i denotes the number of distinct items in the i th nominal attribute. Since the algorithm requires to check the existence of each category is in U , this step costs $O(\binom{d}{l} \cdot |D| \cdot \log(\sum_{1 \leq i_1 < \dots < i_j \leq d} \prod_{1 \leq j \leq l} g_{i_j}))$. Second, building the *probTree* for the global $q(v)$ function costs $O(\binom{d}{l} \cdot |D| \cdot \log |D|)$ because of the necessariness to insert all $(v_{i,j}, p_{i,j})$ pairs. Constructing the function $q_i(v)$ for all categories requires $O(\binom{d}{l} \cdot |D| \cdot \log |D|)$. Finally, building the local $q_i(v)$ functions for all categories costs $O(\binom{d}{l} \cdot |D| \cdot \log |D|)$. Consequently, the overall time complexity is $O(\binom{d}{l} \cdot |D| \cdot \max(\log |D|, \sum_{1 \leq i_1 < \dots < i_j \leq d} \prod_{1 \leq j \leq l} g_{i_j}))$.

3.2 An Advanced Solution

Algorithm basicTkCl computes the exact expected ranks of all categories by using Equ. (2) (at Line 10), which requires to know the global function $q(v)$ and local functions $q_i(v)$ for all categories. However, it is not cheap to construct all the local functions $q_i(v)$, as claimed in Theorem 1. Fortunately, we find it unnecessary to construct the local function $q_i(v)$ for all categories since the expected rank can be estimated only by using the global function $q(v)$, as shown in Equ. (3).

$$r'(c) = \sum_j p_{i,j} \cdot q(v_{i,j}) \tag{3}$$

Algorithm 2. `advancedTkCl`(D, k, l)

```

1:  $U \leftarrow \emptyset$ ;
2: for each tuple  $t_i$  in  $D$ 
3:   for each category  $c$  generated from  $t_i$ 
4:     if ( $c \notin U$ ) then  $U \leftarrow U \cup \{c\}$ ;
5:     Add  $a_{i,tar}$  to the set associated with  $c$ ;
6:   end for
7: end for
8: Scan  $U$  to build a probTree to describe the function  $q(v)$ ;
9:  $Z \leftarrow \emptyset$ ;
10: for each  $c$  in  $U$ 
11:   Compute its expected rank,  $r'(c)$ , by using Equ. (3);
12:   if ( $r'(c) \leq \xi(Z) + 1$ ) //  $\xi(Z)$  returns the  $k$ th minimum value in  $Z$ 
13:      $Z \leftarrow Z \cup \{c\}$ ;
14:     Remove categories (say,  $c'$ ) in  $Z$  satisfying  $r'(c') > \xi(Z) + 1$ ;
15:   end if
16: end for
17: for each  $c$  in  $Z$ 
18:   Compute its expected rank,  $r(c)$ , by using Equ. (2);
19: end for
20: Return top- $k$  tuples with smallest expected rank;

```

The following inequality holds because of two facts. First, $\forall v, q_i(v) \leq 1$. Second, $\sum_j p_{i,j} = 1$.

$$r'(c) - 1 \leq r(c) \leq r'(c) \quad (4)$$

Algorithm `advancedTkCl` (Algorithm 2) illustrates the detailed steps to handle the *TkCl* query avoid generating all $q_i(v)$ functions. It has four steps. The first two steps are similar to Algorithm `basicTkCl`: scanning D to generate all categories, and creating the global $q(v)$ function. In the third step, it estimates the expected ranks of all categories in U by using Equ. (3). Candidate categories are stored in a small data set Z . Here, the function $\xi(Z)$ returns the k th minimum value in Z (at Lines 9-16). Finally, it creates the local functions $q_i(v)$ for all candidate categories to compute their exact expected ranks (at Lines 17-19).

The space consumption of Algorithm `advancedTkCl` is identical to Algorithm `basicTkCl` because both of them generate a set U containing all categories. The time complexity improvement is dependent on the percentage of tuples filtered.

4 Experiments

We have conducted a series of experiments to evaluate the performance of the proposed methods. All codes are written in C++, running in a PC with Intel Core2 2.66GHz CPU and 2G memory.

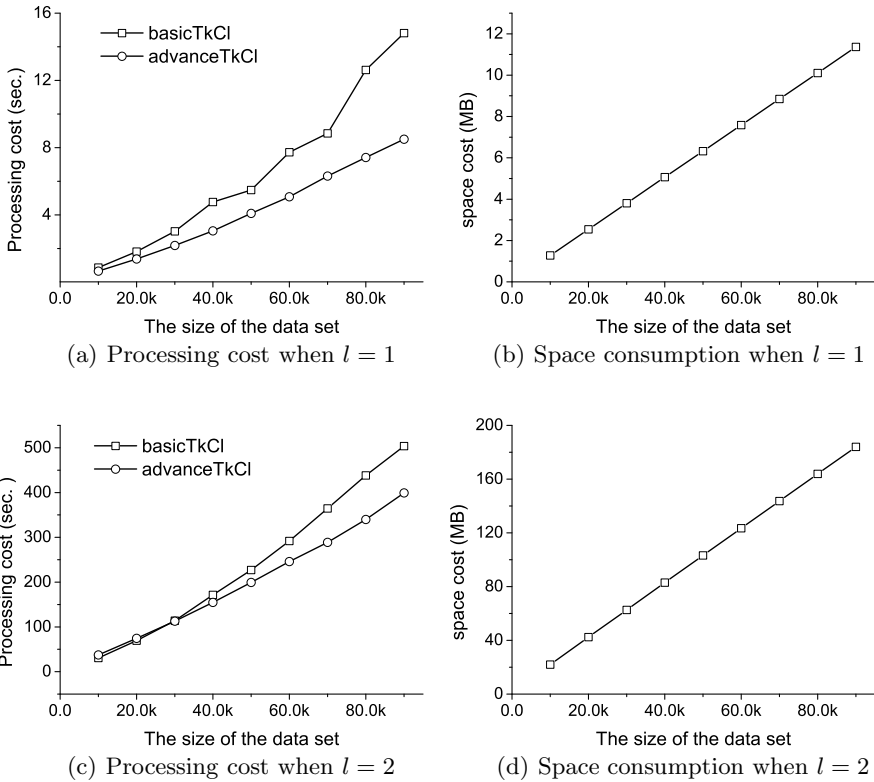


Fig. 2. The performance evaluation when varying the database size

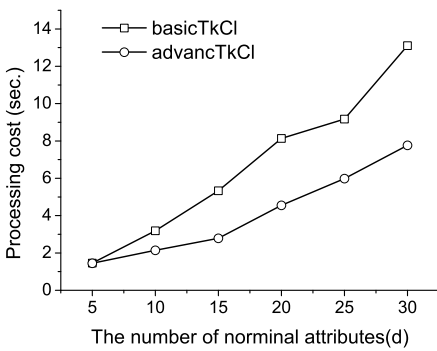
The Census-Income Database (CID)² contains weighted census data extracted from a survey conducted by the U. S. Census Bureau during the mid of 1990s to analyze the income of different people. It has 33 nominal attributes and 7 numeric attributes. Important nominal attributes include *education*, *major industry code*, *occupation code*, *hispanic origin*, etc. Typical numeric attribute include *wage per hour*, *dividends from stocks*, *num persons worked for employer* etc. There are 99,762 records in the data set. It is interesting to find which kinds of people are good at making money. Thus, we construct a new data set containing all 33 nominal attributes and 1 numeric attribute (*wage per hour*).

Figure 2 reports the performance evaluation upon the CID data set. In each test, we vary the database size from 10k to 90k. Figure 2 (a) and (c) compare the executing cost between the basicTkCI algorithm and the advancedTkCI algorithm. The executing time is almost linear to the database size. When the value of l increases, the executing time increases significantly. The advancedTkCI algorithm

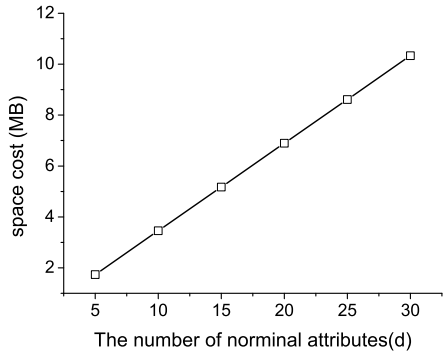
² <http://kdd.ics.uci.edu/databases/census-income/census-income.html>

runs much faster than the basicTkCI algorithm since it only needs to compute the exact expected ranks for a small number of categories. Figure 2 (b) and (d) report the space consumption of the proposed methods. Because these two methods have similar space consumption, we only draw one line in these two figures. The space consumption is almost linear to the size of the data set. When $l = 1$, the space consumption is below 12MB. When $l = 2$, the space consumption rises to about 180MB.

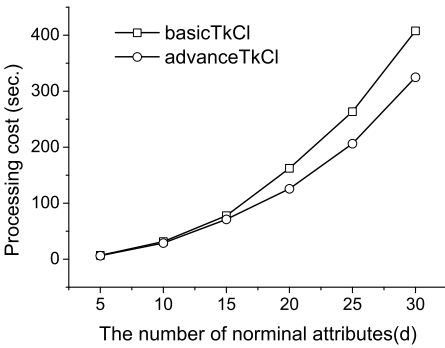
Figure 3 reports the performance evaluation when varying the number of nominal attributes. The CID data set contains 33 nominal attributes. In each test, we only choose the first d nominal attributes. We observe the performance is significantly influenced by the number of nominal attributes. When this value of d increases, it requires to consume more time and space resources to fulfill the task. When $l = 1$, the time cost and the space consumption are approximately linear to the value of d . When $l = 2$, the increasing rate of the cost is higher than that of d , because each numeric attribute value will appear at $\binom{d}{2}$ different categories.



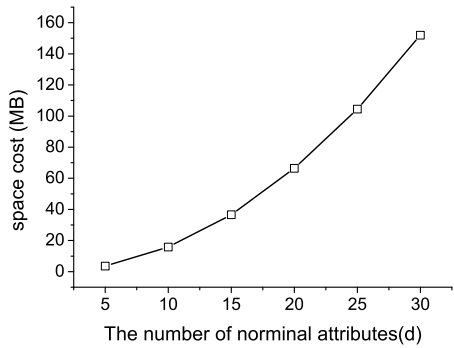
(a) Processing cost when $l = 1$



(b) Space consumption when $l = 1$



(c) Processing cost when $l = 2$



(d) Space consumption when $l = 2$

Fig. 3. The performance evaluation when varying the number of nominal attributes

5 Related Work

Ranking query, a fundamental operator in the database management field, has been well studied in the literature. [7] is a good survey of top- k query processing techniques in relational database systems. The *threshold algorithm* (TA) is a famous Top- k processing techniques [11]. This algorithm uses a parameter T to represent the running-time threshold. If tuples cannot have better scores than the threshold, they are pruned immediately. Other typical top- k processing techniques include Combined Algorithm (CA) [4], etc.

The research of top- k query has been extended to many new scenarios besides the relational database systems recently. Babcock et al. have studied how to process top- k query in the distributed environment, with a goal to minimize the communication cost among the sites [2]. Mouratidis et al. have proposed a method to monitor top- k query continuously over data streams [10]. Their work supports the sliding-window model. The work in [6] studies how to answer top- k typicality queries on large databases.

Most recently, some researchers focus on ranking queries on probabilistic data set. Probabilistic data widely exist in many applications, such as sensor network, etc [1]. There are many uncertain top- k queries proposed in recent years for the different purposes, including U-top k [13], U- k rank [13], PT- k [6], global top- k [14], ER-top k [3], ES-top k [3], UTop-Set k [12], c-Typical-Top k [5] and unified top k [9] queries.

As most of the existing work on ranking query focuses on getting top- k tuples from a data set, the focus of this paper is to get top- k categories from a data set. The set containing all categories can be treated as a probabilistic data set, so that we also use a probabilistic ranking semantic, ER-top k query, to select the top- k categories.

6 Conclusion

Although most of the existing work on ranking queries is aiming at getting top- k tuples from a data set, this paper aims at getting top- k categories from a data set. This paper has two significant features. First, the data distribution, not a single average value, is used to describe each category. Second, a typical probabilistic top k query semantics, ER-top k , is used to rank each category. We have also devised novel methods to handle the proposed method efficiently.

There still remains some work to be done in future. The first piece of work is how to process $TkCl$ query over a probabilistic data set. Recall that the two kinds of uncertainties can occur at the same time. The second piece of work is how to handle the multiple-criteria decision analysis, where a data set not only contains a number of nominal attributes, but also more than one numerical attribute. Under this condition, it is not easy to say whether a category is better than another one. Consequently, it is also interesting to find skyline categories from a data set.

Acknowledgement. The research of Cheqing Jin was supported by the Key Program of National Natural Science Foundation of China (Grant No. 60933001), National Natural Science Foundation of China (Grant No. 60803020), and Doctoral Fund of Ministry of Education of China for New Teachers (Project No. 200802511010). The research of Yizhen Zhang was supported by Shanghai Leading Academic Discipline Project (Project No. B412). The research of Aoying Zhou was supported by National Science Foundation for Distinguished Young Scholars (Grant No. 60925008), and Natural Science Foundation of China (Grant No. 61021004).

References

1. Aggarwal, C.C.: Managing and mining uncertain data. Springer, Heidelberg (2009)
2. Babcock, B., Olston, C.: Distributed top-k monitoring. In: Proc. of SIGMOD (2003)
3. Cormode, G., Li, F., Yi, K.: Semantics of ranking queries for probabilistic data and expected ranks. In: Proc. of ICDE (2009)
4. Fagin, R., Lotem, A., Naor, M.: Optimal aggregation algorithms for middleware. In: Proc. of PODS (2001)
5. Ge, T., Zdonik, S., Madden, S.: Top-k queries on uncertain data: On score distribution and typical answers. In: Proc. of SIGMOD (2009)
6. Hua, M., Pei, J., Zhang, W., Lin, X.: Ranking queries on uncertain data: A probabilistic threshold approach. In: Proc. of SIGMOD (2008)
7. Ilyas, I.F., Beskales, G., Soliman, M.A.: A survey of top-k query processing techniques in relational database systems. ACM Computing Surveys 40(4) (2008)
8. Jin, C., Gao, M., Zhou, A.: Handling er-top k query on uncertain streams. In: Yu, J.X., Kim, M.H., Unland, R. (eds.) DASFAA 2011, Part I. LNCS, vol. 6587, pp. 326–340. Springer, Heidelberg (2011)
9. Li, J., Saha, B., Deshpande, A.: A unified approach to ranking in probabilistic databases. In: Proc. of VLDB (2009)
10. Mouratidis, K., Bakiras, S., Papadias, D.: Continuous monitoring of top-k queries over sliding windows. In: Proc. of ACM SIGMOD (2006)
11. Nepal, S., Ramakrishna, M.V.: Query processing issues in image(multimedia) databases. In: Proc. of ICDE (1999)
12. Soliman, M.A., Ilyas, I.F.: Ranking with uncertain scores. In: Proc. of ICDE (2009)
13. Soliman, M.A., Ilyas, I.F., Chang, K.C.-C.: Top-k query processing in uncertain databases. In: Proc. of ICDE (2007)
14. Zhang, X., Chomicki, J.: On the semantics and evaluation of top-k queries in probabilistic databases. In: Proc. of DBRank (2008)

MFCluster: Mining Maximal Fault-Tolerant Constant Row Biclusters in Microarray Dataset

Miao Wang¹, Xuequn Shang¹, Miao Miao¹, Zhanhuai Li¹, and Wenbin Liu²

¹ School of Computer Science and Engineering, Northwestern Polytechnical University, Xi'an, China, 710072
riyushui@gmail.com, shang@nwpu.edu.cn, taluopaibing@gmail.com, lizhh@nwpu.edu.cn

² Department of Physics and Electronic information engineering, Wenzhou University, Wenzhou, Zhejiang, China, 325035
wbliu6910@126.com

Abstract. Biclustering is one of the most popular methods for microarray dataset analysis, which allows for conditions and genes clustering simultaneously. However, due to the influence of experimental noise in the microarray dataset, using traditional biclustering methods may neglect some significant biological biclusters. In order to reduce the influence of noise and find more types of biological biclusters, we propose an algorithm, *MFCluster*, to mine fault-tolerant biclusters in microarray dataset. *MFCluster* uses several novel techniques to generate fault-tolerant efficiently by merging non-relaxed biclusters. *MFCluster* generates a weighted undirected relational graph firstly. Then all the maximal fault-tolerant biclusters would be mined by using pattern-growth method in above graph. The experimental results show our algorithm is more efficiently than traditional ones.

Keywords: fault-tolerant, bicluster, microarray, constant row.

1 Introduction

Biclustering [1] is one of the popular methods for gene expression data analysis, which is a methodology for mining co-expressed genes across a subset of experimental conditions or samples simultaneously. The existing biclustering methods are classified into four categories [2]: (i) Constant value biclusters. *SAMBA* [3] is designed to find constant value biclusters. (ii) Constant row or column biclusters. *xMotifs* [4] focus on biclusters with constant columns and Cheng and Church's method [1] has good performance on constant value and constant row or column biclusters. (iii) Coherent value biclusters. The weighted multigraph has been introduced by Zaki [5] to handle above three types of biclusters. (iv) Coherent evolution biclusters. Biclusters with coherent trends of up or down regulation can be found by *OPSM* [6].

However, due to the influence of noisy character of microarray technology, traditional biclustering methods have poor scalability for mining fault-tolerant

biclusters. For example, in Table 1, RAP [7] can find many constant row biclusters, such as, $G_1G_2G_4(S_1S_2S_3S_4)$, $G_1G_2G_3G_5(S_1S_3S_5)$ and $G_1G_2G_4G_5(S_1S_2S_3)$, etc. However, if the expressed value of gene G_4 in sample S_5 is error, the significative bicluster should be $G_1G_2G_4(S_1S_2S_3S_4S_5)$, which would be neglected by traditional biclustering method.

Table 1. An Example Microarray Dataset

	S_1	S_2	S_3	S_4	S_5
G_1	0.85	1.21	0.12	1.85	2.4
G_2	0.86	1.19	0.14	1.86	2.34
G_3	0.85	1.5	0.11	2.85	2.32
G_4	0.84	1.22	0.12	1.84	4.36
G_5	0.85	1.19	0.13	2.62	2.35

Recently, some algorithms [8, 9] have been proposed for mining fault-tolerant patterns. These algorithms allow a user-defined parameter to get fault-tolerant patterns. Most of these algorithms use greedy or heuristics approach for mining fault-tolerant patterns which is not very efficient. [10] proposed to mine fault-tolerant biclusters from binary dataset, it cannot be used for mining gene expression dataset. [11] is the recent proposed novel approach for mining fault-tolerant (also called error-tolerant) biclusters in real-valued microarray dataset. It uses the Range Support [7] and greedy approach to mine constant row biclusters. Since mining fault-tolerant patterns of biclusters increases the complication of algorithm, so the performance of above algorithms is not very efficiently.

A straightforward approach for mining fault-tolerant biclusters is to mine traditional biclusters and merge these biclusters based on user-defined fault-tolerant thresholds. This approach has a major drawback that such two-step framework of mining and merging is very time consuming. However, as mentioned in above, there could not propose one algorithm for mining all types of biclusters. Therefore, there could also design an algorithm for mining all types of fault-tolerant biclusters, which motivates to investigate a general efficient algorithm to mine all types of fault-tolerant bicluster.

According to above analysis and in hopes of overcoming the limitation of traditional fault-tolerant bicluster mining method, we propose an algorithm, *MFCluster*, to efficient mine *Maximal Fault-tolerant biCluster*. In order to propose a general mining algorithm, we use two-step method to generate fault-tolerant bicluster. Each type of bicluster can be found by existing biclustering method. Then the fault-tolerant biclusters can be obtained by merging these traditional biclusters. Firstly, *MFCluster* generates a weighted undirected relational graph. Then all the maximal fault-tolerant biclusters would be mined by using pattern-growth method in above graph. Unlike to the naïve merging algorithm, we use some novel techniques for pruning non-maximal fault-tolerant biclusters. The contributions of this paper are summarized as follows. (1) We propose to mine fault-tolerant biclusters in the

weighted undirected relational graph. (2) In stead of using traditional Apriori-Like merging method to mine fault-tolerant biclusters, *MFCluster* is proposed by using depth-first and pattern-growth method to generate fault-tolerant biclusters efficiently. (3) Without using traditional candidate maintenance-and-test scheme to generate maximal fault-tolerant biclusters, *MFCluster* uses several efficient pruning techniques to generate maximal fault-tolerant biclusters.

2 Problem Definition

The microarray is presented as $D = C \times G$, where the column C denotes the different experimental conditions, and the row G denotes genes. For clarity, C and G in D are denoted as $D.Samples$ and $D.Genes$, respectively. The element value of $D_{c,g}$ is a real value which is the expression level of gene g under condition c . A bicluster B is defined as a sub-matrix of D , where $B = X \times Y$ with $X \subseteq C$, $Y \subseteq G$. A bicluster can be denoted as $Genes(Samples)$. Given M be the set of all biclusters in D , $N = P \times Q \subseteq M$ is a maximal bicluster if and only if there does not exist a bicluster $E = J \times K$ such that $P \subseteq J$ and $K \subseteq Q$.

In this paper, we will use range support [7] to generate maximal fault-tolerant constant row biclusters. The range support is defined as following.

Definition 1. Given a microarray dataset D and a self-assignment value α . Therefore, the *Range Support* of a real-valued gene set $G = \{g_1, g_2, \dots, g_k\}$ is defined as $RangeSupport(G) = \sum_{c \in C} RS(c, G)$, where $RS(c, G)$ denotes as

$$RS(c, G) = \begin{cases} \min_{g \in G} |D_{c,g}| & \text{if } [\forall g \in G, D_{c,g} > 0 \text{ or } \forall g \in G, D_{c,g} < 0] \\ & \& \left[(\max_{g \in G} D_{c,g} - \min_{g \in G} D_{c,g}) \leq \alpha \min_{g \in G} |D_{c,g}| \right] \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

A constant row bicluster is interesting if all the genes under conditions are co-expressed based on above range support definition. In fault-tolerant bicluster mining, we would relax the interesting constraint by allowing a number of (sample, gene) pairs which are not co-expressed with other co-expressed genes under conditions. A fault-tolerant bicluster (FT-Bicluster) should be satisfied the following relaxation criteria (Compared to FT-Bicluster, traditional bicluster is also called non-relaxed bicluster in this paper).

Definition 2. Given two biclusters B_1 and B_2 , if B_1 and B_2 can be merged to one fault-tolerant bicluster, the following relaxation criteria should be both satisfied:

$$\frac{|B_1.Genes \cap B_2.Genes|}{|B_1.Genes \cup B_2.Genes|} \geq \theta \text{ and } \frac{|B_1.Samples \cap B_2.Samples|}{|B_1.Samples \cup B_2.Samples|} \geq \omega.$$

Biologically speaking, mining maximal bicluster is more valuable relative to discover all biclusters. The maximal fault- tolerant bicluster is defined as following.

Definition 3. A FT-Bicluster F is a maximal one, if there cannot find another FT-Bicluster E , whose $F.Genes \subseteq E.Genes$ and $F.Samples \subseteq E.Samples$.

In our method, we mine maximal fault-tolerant biclusters by using weighted undirected relational graph (WUGraph). The definition of WUGraph is shown as following.

Definition 4. The weighted undirected relational graph is $R=\{E, V, W\}$, where each vertex V_i in the graph represent an unique bicluster, there exist an edge E_{ij} between a pair of biclusters V_i and V_j only if above biclusters satisfy the Definition 3 and weighted item set W_{ij} is the merged fault-tolerant bicluster from a pair of biclusters. For clarity, W_{ij} is denoted as $V_iV_j.Bicluster$.

3 The MFCluster Algorithm

3.1 Construct the Weighted Undirected Relational Graph

According to the definition of bicluster merging criteria, if one fault-tolerant bicluster is generated by merging several traditional biclusters, any two traditional biclusters satisfy the relaxation criteria thresholds. Such anti-monotonicity property motivates us to generate maximal FT-Bicluster by depth-first method.

One simple pattern-growth method to generate FT-Biclusters by merging non-relaxed biclusters is to merge the candidate bicluster with the current extending FT-Bicluster. For example, supposed the current candidate bicluster is $G_1G_2G_4G_5(S_1S_2S_3)$ and the current extending FT-Bicluster is $G_1G_2G_4(S_1S_2S_3S_4S_5)$ which is obtained by merging $G_1G_2(S_1S_2S_3S_4S_5)$ and $G_1G_2G_4(S_1S_2S_3S_4)$. The relaxation criteria thresholds are both to be 0.6. According to the definition of relaxation criteria (the relaxation thresholds are both to be 0.6), $G_1G_2G_4G_5(S_1S_2S_3)$ can be merged to FT-Bicluster $G_1G_2G_4(S_1S_2S_3S_4S_5)$ and the new FT-Bicluster $G_1G_2G_4G_5(S_1S_2S_3S_4S_5)$ is generated. However, such FT-Bicluster has fault relaxation information. Since non-relaxed bicluster $G_1G_2(S_1S_2S_3S_4S_5)$ and $G_1G_2G_4G_5(S_1S_2S_3)$ are not satisfied the definition of relaxation criteria. Therefore, if one bicluster can be extended to the current extending FT-Bicluster, it should be satisfied the following definition.

Definition 5. Supposed $B_i...B_j$ be the current extending FT-Bicluster, which is obtained by merging $B_i, ...,B_j$. If one bicluster B_m is a candidate bicluster, each bicluster B_n in $B_i, ...,B_j$ should be satisfied:

$$\frac{|B_n.Genes \cap B_m.Genes|}{|B_n.Genes \cup B_m.Genes|} \geq \theta \text{ and } \frac{|B_n.Samples \cap B_m.Samples|}{|B_n.Samples \cup B_m.Samples|} \geq \omega .$$

According to the Definition 5, there are more times of computing relaxation criteria between a pair of biclusters. Therefore, we construct a weighted undirected graph to

record FT-Biclusters between a pair of non-relaxed biclusters. If two non-relaxed biclusters satisfy the relaxation criteria, one edge would be generated between such biclusters and the weight of this edge is the FT-Bicluster which is generated by this pair of non-relaxed biclusters.

3.2 Mining Maximal FT-Biclusters

In this part, we will introduce how *MFCluster* algorithm finding the maximal FT-Bicluster by using vertex-growth method from the constructed the weighted undirected relational graph. Traditional efficient maximal pattern mining technique is backward checking. For example, in [12], if there is existed another priori candidate sample whose gene set is the superset of gene set in the current extended candidate sample, the current extended sample would be pruned. However, such pruning technique cannot be used for FT-Bicluster pruning. For example, supposed the current extending bicluster is $G_1G_2(S_1S_2S_3S_4S_5)$, the relaxation criteria thresholds are both to be 0.5, and the candidate non-relaxed biclusters of $G_1G_2(S_1S_2S_3S_4S_5)$ are $G_1G_2G_4(S_1S_2S_3S_4)$, $G_1G_2G_5(S_1S_2S_3S_5)$, $G_1G_2G_4G_5(S_1S_2S_3)$ and $G_1G_2G_3G_5(S_1S_3S_5)$. $G_1G_2G_4(S_1S_2S_3S_4)$ is the prior candidate bicluster of $G_1G_2G_5(S_1S_2S_3S_5)$ and $G_1G_2G_4(S_1S_2S_3S_4)$ can be extended to $G_1G_2G_5(S_1S_2S_3S_5)$. According to traditional pruning technique, $G_1G_2G_5(S_1S_2S_3S_5)$ should be pruned. However, $G_1G_2G_5(S_1S_2S_3S_5)$ can be extended to $G_1G_2G_3G_5(S_1S_3S_5)$ to generate $G_1G_2G_3G_5(S_1S_2S_3S_5)$. However, $G_1G_2G_4(S_1S_2S_3S_4)$ cannot be extended to $G_1G_2G_3G_5(S_1S_3S_5)$. Therefore, based on our observation and analysis, traditional pruning technique should be changed for mining maximal FT-Biclusters according to the following lemma.

Lemma 1. Given P be the current extending FT-Bicluster, M is the candidate set of P and N is the priori candidate set of P . Supposed the current candidate item is $M_i, M_i \in M$, and N_j is a priori candidate item where $N_j \in N$. If M_i should be pruned, it must satisfy all the following conditions. (1) $PN_j.Samples$ is the subset of $PM_i.Samples$; (2) $PN_j.Genes$ is the subset of $PM_i.Genes$; (3) PM_iN_j is FT-Bicluster; (4) For each other candidate gene M_j in M , PM_jN_j is FT-Bicluster.

According to Lemma 1, if one candidate is satisfied Lemma 1, other candidates must satisfy Lemma 1. For example, supposed the current extending bicluster is $G_1G_2G_4(S_1S_2S_3S_4)$, the relaxation criteria thresholds are both to be 0.5, and the current candidate non-relaxed biclusters are $G_1G_2G_5(S_1S_2S_3S_5)$, $G_1G_2G_4G_5(S_1S_2S_3)$. The priori candidate are $G_1G_2(S_1S_2S_3S_4S_5)$ and $G_1G_2G_3G_4G_5(S_1S_3)$. The FT-Bicluster $G_1G_2G_4G_5(S_1S_2S_3S_4S_5)$ can be obtained by combining $G_1G_2G_5(S_1S_2S_3S_5)$ to $G_1G_2G_4(S_1S_2S_3S_4)$. And the FT-Bicluster $G_1G_2G_4(S_1S_2S_3S_4S_5)$ is generated by extending $G_1G_2G_4(S_1S_2S_3S_4)$ to $G_1G_2(S_1S_2S_3S_4S_5)$. Since $G_1G_2G_4(S_1S_2S_3S_4S_5)$ is the subset of $G_1G_2G_4G_5(S_1S_2S_3S_4S_5)$ and $G_1G_2G_4(S_1S_2S_3S_4S_5)$ can be extended to $G_1G_2G_4G_5(S_1S_2S_3S_4S_5)$, according to Lemma 1, $G_1G_2G_5(S_1S_2S_3S_5)$ should be pruned. However, if the candidate bicluster only satisfies (3) in Lemma 1, it should not both be pruned and output. Since the priori candidate bicluster must extend the candidate

bicluster before. Therefore, the following lemma can guarantee not to output non-maximal FT-Bicluster.

Lemma 2. Given P be the current extending FT-Bicluster, M is the candidate set of P and N is the priori candidate set of P . Supposed the current candidate item is $M_i, M_i \in M$, and N_j is a priori candidate item where $N_j \in N$. If M_i does not satisfy Lemma 1 and PM_i can be extend to PN_j based on relaxation criteria, M_i can be extended to P , but it cannot be output.

According to above lemmas, if *MFCluster* generates FT-Biclusters from the edges which have smallest scale of FT-Bicluster in WUGraph, it would be more efficiently than extending non-smallest scale of FT-Biclusters. Since larger FT-Biclusters can be pruned in time according to Lemma 1. Therefore, *MFCluster* uses the following lemma to increase efficiency.

Lemma 3. Supposed the current extending a pair of biclusters in WUGraph is B_iB_j . If there could find another vertex B_m where $B_jB_m.Genes$ is the subset of $B_iB_j.Genes$ and $B_jB_m.Samples$ is the subset of $B_iB_j.Samples$, or $B_iB_m.Genes$ is the subset of $B_iB_j.Genes$ and $B_iB_m.Samples$ is the subset of $B_iB_j.Samples$, B_iB_j should be pruned.

Using above lemmas can prune non-maximal FT-Biclusters efficiently, but it cannot prune all the non-maximal FT-Biclusters. Since different non-relaxed maximal biclusters may generate same FT-Bicluster. For examples, the non-relaxed bicluster $ABCDE(12345)$ can be merged to $ABDE(123456)$ and $ABCD(123456)$, respectively, and get the same FT-Bicluster $ABCDE(123456)$. Therefore, we need the maximal FT-Biclusters generated by using above lemmas may be redundant. So the result needs to be checked based on maximal FT-Bicluster definition. Based on above lemmas and analysis, *MFCluster* algorithm is designed for mining maximal FT-Biclusters. The detailed *MFCluster* is illustrated in Algorithm 1.

Algorithm 1. *MFCluster* algorithm

Input: The maximal non-relaxed bicluster set: S , the minimum gene relaxation threshold: θ , the minimum sample relaxation threshold: ω , WUGraph: L , the current extending *FT-Bicluster*: P ,

Output: The complete set of maximal FT-Biclusters: F .

Initialization: $P=\emptyset$, $L=\emptyset$; Global $g=\emptyset$;

Method: *MFCluster*(S , θ , ω , L , P).

if $L=\emptyset$, Scan S to construct the WUGraph: L , g is pointed to the first edge of L ;

end if

MiningBicluster(S , θ , ω , L , P , F);

FinalOutput(F);

```

Exit;
Procedure MiningBicluster( $S, \theta, \omega, L, P, F$ ).
if  $P = \emptyset$ ,
     $P = g, g = g \rightarrow \text{next}$ ;
else
    break;
end if
Finding all the candidate FT-Bicluster set  $C$  of  $P$  and
the priori FT-Bicluster set  $E$  of  $P$ ;
if  $C$  is Null and does not satisfy Lemma 2,
Store  $P$  to  $F$ ;
end if
for each candidate FT-Bicluster  $c$  in  $C$ , do
    if  $c$  satisfies Lemma 1 or Lemma 3,
        break;
    end if
     $P.Samples = P.Samples \cup c.Samples$ ;
     $P.Genes = P.Genes \cup c.Genes$ ;
    MiningBicluster( $S, \theta, \omega, L, P$ );
end for
return;
Procedure FinalOutput( $F$ ).
for each FT-Bicluster  $F_1$  in  $F$ 
    if there could not find another FT-Biclster  $F_2$  which
        is the superset of  $F_1$ , then
        Output( $F_1$ );
    end if
end for
return;

```

4 Experiments

4.1 Dataset

AGEMAP [13] which catalogs changes in gene expression as a function of age in mice, would be used for our test dataset. It includes expression changes for 8,932 genes and 16,896 *cDNA* clones in 16 tissues as a function of age. For each tissue, there are five male and five female mice aged 1, 6, 16, 24 month. In this paper, we

only analyze one mice set (one male and one female) on three tissues, which are Hippocampus, Heart and Gonads respectively and we only focus on mining age-related co-expressed gene set between mice aged 6 months and 16 months. The detail of how these datasets are obtained can be found in [12].

4.2 Efficiency Comparison

In this section, the performance of *MFCluster* algorithm is compared with two algorithms. One is the general naïve FT-Bicluster mining algorithm (denoted as *FTMerging*) which merges the non-relaxed biclusters to generate FT-Biclusters by

Table 2. The running time and number of mining ft-biclusters comparison among algorithms in each mice dataset

Algorithm ID	Parameter Settings	Time Taken (Second)
Dataset 1 (Row Support=5, total genes=2023, total biclusters=873)		
<i>ET-bicluster 1</i>	$\xi=0.1, \alpha=0.3, RS=3$	790.412
<i>FTMerging 1</i>	$\theta=0.8, \omega=0.8$	540.782
<i>MFCluster 1</i>	$\theta=0.8, \omega=0.8$	541.821
<i>ET-bicluster 2</i>	$\xi=0.2, \alpha=0.3, RS=3$	1299.589
<i>FTMerging 2</i>	$\theta=0.7, \omega=0.7$	542.41
<i>MFCluster 2</i>	$\theta=0.7, \omega=0.7$	542.029
<i>ET-bicluster 3</i>	$\xi=0.3, \alpha=0.3, RS=3$	1002.991
<i>FTMerging 3</i>	$\theta=0.6, \omega=0.6$	549.643
<i>MFCluster 3</i>	$\theta=0.6, \omega=0.6$	543.06
Dataset 2 (Row Support=6, total genes=1128, total biclusters=1652)		
<i>ET-bicluster 4</i>	$\xi=0.1, \alpha=0.5, RS=3$	163.867
<i>FTMerging 4</i>	$\theta=0.8, \omega=0.8$	171.89
<i>MFCluster 4</i>	$\theta=0.8, \omega=0.8$	174.273
<i>ET-bicluster 5</i>	$\xi=0.2, \alpha=0.5, RS=3$	174.334
<i>FTMerging 5</i>	$\theta=0.7, \omega=0.7$	189.253
<i>MFCluster 5</i>	$\theta=0.7, \omega=0.7$	176.269
<i>ET-bicluster 6</i>	$\xi=0.3, \alpha=0.5, RS=3$	569.772
<i>FTMerging 6</i>	$\theta=0.6, \omega=0.6$	255.528
<i>MFCluster 6</i>	$\theta=0.6, \omega=0.6$	178.041
Dataset 3 (Row Support=7, total genes=532, total biclusters=2502)		
<i>ET-bicluster 7</i>	$\xi=0.1, \alpha=0.8, RS=3$	69.016
<i>FTMerging 7</i>	$\theta=0.8, \omega=0.8$	96.684
<i>MFCluster 7</i>	$\theta=0.8, \omega=0.8$	79.238
<i>ET-bicluster 8</i>	$\xi=0.2, \alpha=0.8, RS=3$	251.216
<i>FTMerging 8</i>	$\theta=0.7, \omega=0.7$	174.955
<i>MFCluster 8</i>	$\theta=0.7, \omega=0.7$	80.859
<i>ET-bicluster 9</i>	$\xi=0.3, \alpha=0.8, RS=3$	2666.407
<i>FTMerging 9</i>	$\theta=0.6, \omega=0.6$	794.714
<i>MFCluster 9</i>	$\theta=0.6, \omega=0.6$	95.503

using Apriori approach. Then the maximal FT-Biclusters would be output based on definition. The other FT-Bicluster mining algorithm is *ET-bicluster* which is implemented according to the description in [11]. It proposes a bottom-up heuristic-based mining algorithm to discover constant row error-tolerant biclusters from real-valued gene expression data. Since naïve method and *MFCluster* generate FT-Biclusters by merging non-relaxed biclusters, we need to use the recent proposed algorithm, *MRCluster* [14], to mine maximal non-relaxed biclusters firstly. Range support parameters α and RS are same to *ET-bicluster*'s. Therefore, the running times of *MFCluster* and naïve method are the process duration of maximal non-relaxed biclusters mining and FT-Bicluster mining.

Table 2 shows the running time among above three algorithms with different parameters settings in different scale microarray datasets. It is shown that *MFCluster* is faster than other algorithms in most datasets when generating almost the same number of FT-Biclusters. However, *MFCluster* can find larger scale FT-Bicluster. If the relaxation criteria parameters are set to be greater, *MFCluster* may consume more time than *FTMerging* or *ET-bicluster*. The reason is that greater parameters can get lower relaxation, which cannot use pruning techniques frequently in *MFCluster*. Therefore, *MFCluster* is suitable for mining dense and larger number of biclusters to generate FT-Biclusters. The less efficiency of *ET-bicluster* and naïve method can be attributed to its unique candidate pruning method based on the definition of maximal bicluster. And the parameter of error-tolerant lets *ET-bicluster* increase more time to compute.

5 Conclusions

In this paper, we propose an algorithm, *MFCluster*, to efficient mining maximal FT-Bicluster by merging non-relaxed biclusters. *MFCluster* can find maximal FT-Biclusters efficiently. Compared with the existing algorithms, it is shown that our algorithm is more efficiently. However, there remain several limitations and further investigations. (1) Although mining fault-tolerant biclusters in WUGraph can increase efficiency, it would cost memory for maintaining all the fault-tolerant between a pair of non-relaxed biclusters. We would trade off between efficiency and memory for mining FT-Bicluster efficiently. (2) Since the pruning techniques of *MFCluster* are based on backward checking. If the priori candidate set were huger, the pruning efficiency would not be well. In the future, we would design some efficient pruning technique for mining dense datasets.

Acknowledgement. The work is supported by the National Natural Science Foundation of China under Grant No.60703105 and No. 60970065. It is also partly supported by the Research Foundation at Northwestern Polytechnical University of China under Grant No.JC201042 and the National High-Tech Research and Development Plan of China under Grant Nos.863-317-01-04-99, 2009AA1Z134. This material is also based upon work funded by Zhejiang Provincial Natural Science Foundation of China under Grant No. R1110261.

References

- [1] Cheng, Y., Church, G.M.: Biclustering of Expression Data. In: Proc. 8th Int'l Conf. Intelligent Systems for Molecular Biology (ISMB 2000), pp. 93–103. ACM Press, New York (2000)
- [2] Madeira, S.C., Oliveira, A.L.: Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM TCBB* 1(1), 24–45 (2004)
- [3] Subramanian, A., Tamayo, P., Mootha, V., Mukherjee, S., Ebert, B., Gillette, M., Paulovich, A., Pomeroy, S., Golub, T., Lander, E.: Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. *PNAS* 102(43), 15545–15550 (2005)
- [4] Murali, T.M., Kasif, S.: Extracting conserved gene expression motifs from gene expression data. In: Proc. Pac. Symp. Biocomput., pp. 77–88 (2003)
- [5] Zhao, L., Zaki, M.: MicroCluster: Efficient deterministic biclustering of Microarray data. *IEEE Intelligent Systems* 20(6), 40–49 (2005)
- [6] Ben-Dor, A., Chor, B., Karp, R., Yakhini, Z.: Discovering local structure in gene expression data: the order-preserving submatrix problem. *J. Comput. Biol.* 10, 373–384 (2003)
- [7] Pandey, G., Atluri, G., Steinbach, M., Myers, C.L., Kumar, V.: An association analysis approach to biclustering. In: Proc. ACM Conf. on Knowledge Discovery and Data Mining, pp. 677–686 (2009)
- [8] Besson, J., Robardet, C., Boulicaut, J.F.: Mining a new fault-tolerant pattern type as an alternative to formal concept discovery. In: Schärfe, H., Hitzler, P., Øhrstrøm, P. (eds.) ICCS 2006. LNCS (LNAI), vol. 4068, pp. 144–157. Springer, Heidelberg (2006)
- [9] Poernomo, A.K., Gopalkrishnan, V.: Towards efficient mining of proportional fault-tolerant frequent itemsets. In: ACM SIGKDD, pp. 697–706. ACM, New York (2009)
- [10] Colantonio, C., Pietro, R.D., Ocello, A., Verde, N.V.: ABBA: Adaptive bicluster-based approach to impute missing values in binary matrices. In: Proceedings of the 25th ACM Symposium on Applied Computing (SAC 2010), pp. 1026–1033 (2010)
- [11] Gupta, R., Rao, N., Kumar, V.: Discovery of Error-tolerant Biclusters from Noisy Gene Expression Data. In: Proceedings of BIOKDD 2010, Washington DC, USA (2010)
- [12] Wang, M., Shang, X.Q., et al.: FDCluster: Mining Frequent Closed Discriminative Bicluster without Candidate Maintenance in Multiple Microarray Datasets. In: ICDM Workshops 2010, pp. 779–786 (2010)
- [13] Zahn, J.M., Poosala, S., et al.: AGEMAP: A gene expression database for aging in mice. *PLOS Genetics* 3(11), 2326–2337 (2007)
- [14] Miao, M., Shang, X.Q., et al.: MRCluster: Mining constant row bicluster in gene expression data. In: Proceedings of the 3rd WASE International Conference Information Engineering (WASE ICIE 2011) (in press, 2011)

Expansion Finding for Given Acronyms Using Conditional Random Fields

Jie Liu¹, Jimeng Chen¹ Tianbi Liu², and Yalou Huang²

¹ College of Information Technical Science, Nankai University, Tianjin 300071, China
² College of Software, Nankai University, Tianjin 300071, China

Abstract. There are increasingly amount of acronyms in many kinds of documents and web pages, which is a serious obstacle for the readers. This paper addresses the task of finding expansions in texts for given acronym queries. We formulate the expansion finding problem as a sequence labeling task and use Conditional Random Fields to solve it. Since it is a complex task, our method tries to enhance the performance from two aspects. First, we introduce nonlinear hidden layers to learn better representations of the input data under the framework of Conditional Random Fields. Second, simple and effective features are designed. The experimental results on real data show that our model achieves the best performance against the state-of-the-art baselines including Support Vector Machine and standard Conditional Random Fields.

Keywords: Web-mining, text-mining, named entities recognition, acronym.

1 Introduction

Abbreviations and acronyms are widely used to simplify our writing and reading and convey more information with less words. Although using acronyms makes the writing and reading more fluent, they construct obstacles for the readers who do not have the domain-specific knowledge. Thus, the need for establish a database of acronyms is getting more and more important. However, it is very difficult and costive to collect acronyms and the corresponding explanations manually.

Much research effort has been devoted to constructing acronym search. Previous work deals with manual or automatical building dictionaries of acronyms and their explanations, a.k.a expansions. Usually, there are two main steps in the existing automatically built acronym search systems. First, a system detects the occurrence of acronyms which are often not regular words. Second, the systems tries to recognize the corresponding expansions in the same sentence or context. The first step is relatively easy and can be effectively solved, while the second step which is more difficult is the key problem of the acronym search system.

The existing methods can be classified into two categories: pattern-match method and supervised learning method. There are two important previous works using pattern-match method. One is AFP (Acronym Finding Program)

[1], and the other is acronym search system TLA (Three Letter Acronym) [2] proposed by Yeates et al. Both of systems use complex patterns and heuristics to match the acronym and its expansions. Some other works [3] [4] use similar methods to design complex patterns and rules to extract acronyms and expansions in texts. The pattern-match methods often fails, because it is very difficult to design enough and precise rules or patterns to get good precision and recall. Whereas, the supervised methods that can learn the patterns automatically from large corpus have shown the advantages over the other kind of acronym-expansion finding methods. However, the existing supervised learning methods ignore the sequential information in the model.

However, the existing machine learning approaches for acronym/expansion search do not consider the structural information which is very important for this kind of tasks. In the task that we focus, the label of a token not only depends on the features of the current token or a window of tokens but also the labels of the prior tokens. In other words, the examples are not independent. So we treat the expansion finding problem as a sequence labeling task and propose a CRF model to solve it. To our best knowledge, there is no study on applying CRF to acronym search. The CRF [5] model and its related models have achieved state-of-the-art performance in many kinds of structured prediction tasks. The standard CRFs model the interactions between labels, which offers them the ability to learn the structures and dynamics of sequence data. CRF have been applied successfully to many sequence labeling tasks such as Natural Language Processing [6] [7] [8] [9], bioinformatics [10] [11], and computer vision [12] [13] etc., because they have the ability to learn the structures and dynamics of sequence data. To handle the complexity and nonlinearity of the data Lafferty et al. [14] present a kernelized version of linear CRF to enable nonlinear mapping. Our previous work [15] proposes a deep CRF which can learns high level latent features from raw input. A similar model on combining neural network and CRF [16] was proposed for the same purpose recently.

Specifically, we focus on the core problem of acronym search, that is, finding the corresponding expansions given acronym queries. We propose to use an CRF and neural network hybrid model to solve this problem. The reason we focus on this problem instead of the two-step problem as previous systems does is twofold. First, finding expansions for given acronyms is the core task of the acronym search system. Second, our proposed method can run as an independent expansion search system, which has been ignored by previous systems. The experimental results demonstrate that our model and the designed features achieved the best performance on the real dataset constructed from wikipedia.org.

2 Condition Random Fields

The Condition Random Fields (CRF) have been very widely used in many sequence labeling tasks. They are discriminative models who model conditional probability $P(\mathbf{y}|\mathbf{x})$ directly, rather than joint probability $P(\mathbf{y}, \mathbf{x})$ like Hidden

Markov Model. It is widely recognized that the discriminative models are better than generative models for supervised labeling tasks [17]. Compared with the traditional sequence model HMM, CRF models allow arbitrary, non-independent features on the observation sequence X , because the probability of a transition between labels may depend on past and future observations.

For the CRF model in the expansion finding problem, we want to learn a mapping from a token sequence $\mathbf{x} = \{x_1, x_2, \dots, x_T\}$ to a label sequence $\mathbf{y} = \{y_1, y_2, \dots, y_T\}$. A linear chain CRF defines the conditional probability of \mathbf{y} as

$$P(\mathbf{y}|\mathbf{x}; \theta) = \frac{1}{Z(\mathbf{x}; \theta)} \exp \left(\sum_t \theta^\top f(y_t, y_{t-1}, \mathbf{x}_t) \right) \quad (1)$$

where θ is a vector of linear weights, \mathbf{x}_t is the token at position t of \mathbf{x} , and $f(y_t, y_{t-1}, \mathbf{x}_t)$ computes a set of features given the node at time t .

The term $Z(\mathbf{x}; \theta)$ is a normalization factor over all the possible states of \mathbf{y}' of length $|\mathbf{x}|$ defined as

$$Z(\mathbf{x}; \theta) = \sum_{y', y''} \exp \left(\sum_t \theta^\top f(y', y'', \mathbf{x}_t) \right). \quad (2)$$

Typically, given a set of training example $\mathcal{D} = \{\mathbf{x}^{(n)}, \mathbf{y}^{(n)}\}_{n=1}^N$, where $x^{(n)} \in \mathcal{X}$ and $y^{(n)} \in \mathcal{Y}$, the linear weights can be estimated by maximizing the penalized log-likelihood with respect to the conditional probability

$$\max_{\theta} \left\{ \sum_{n=1}^N \log P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}; \theta) \right\} \quad (3)$$

There are efficient exact inference algorithms for linear chains CRFs such as Viterbi algorithm [18].

3 Expansion Identification Model Based on Conditional Random Fields

We consider the problem of expansion finding in text given acronym queries. In this section we will first analyze the characteristic of this problem. Then we describe the details of the expansion identification model which is a hybrid model of CRF and neural network.

3.1 Expansion Identification Task

The goal of expansion identification is to recognize tokens that constitute the long form of a given acronym. For example, when the system is given an acronym ‘BBC’, it tries to find a sequence of words ‘British Broadcasting Company’ in the sentence. Such a task is quite related to sequence labeling tasks.

Query:	BBC
Sentence:	<i>A new non-commercial entity called the British Broadcasting Corporation established...</i>
True labels:	O O O O O O B I I O ...

Fig. 1. An example tagging of a training sentence for the linear-chain CRF. The label ‘B’ stands for ‘Begin of an expansion’, ‘I’ stands for ‘Inside of an expansion’, and ‘O’ stands for ‘Others’.

The previous work see the task of expansion identification as a token categorization task that determines the class label y_i for each token x_i in the sentence. Even though some work includes features of the surrounding tokens for x_i , e.g. the form of x_{i+1} and/or x_{i-1} , the classifier determines the class label y_i for each token x_i independently. However, as described in Section 1, the expansions often have typical internal structure: a token would be more likely to be a member of expansion sequence if its antecedent is a member of expansion. Moreover, we would like to model the structure of a sequence rather than modeling just the label for each token. Thus, the task is more suitably formalized as a sequence labeling problem: given a acronym query q and a sentence with tokens $x = (x_1, \dots, x_n)$, determine the optimal sequence of token labels names $y = (y_1, \dots, y_n)$ of all possible sequences. With the predicted label sequence y , we can identify that whether there is a expansion for the acronym term q ; and which subsequence of the sentence is the corresponding expansion.

It is common for NLP tasks such as NP chunking to represent a chunk (e.g., NP) with two labels, the begin (e.g., B-NP) and inside (e.g., I-NP) of a chunk [19]. An example of the BIO labeling method is shown in figure 1. Specifically, we use ‘B’ for ‘Beginning of expansion’, ‘I’ for ‘Inside of expansion’, and ‘O’ for ‘Others’.

3.2 Neural Network and CRF Hybrid Model

Traditional CRF models are restricted by their linearity. In order to model the sequence of nonlinear data, we introduce a nonlinear transformation layer to compute hidden representation of input observation vectors. Here the hidden representation can be seen as powerful features leaned from input data. Consider a sequence of observations $\mathbf{x} = \{x^{(n)}\}_{i=1}^N$ and labels $\mathbf{y} = \{y^{(n)}\}_{i=1}^N$. Let $y_t \in y = \{1, \dots, C\}$, we use $c_t = [\mathbb{1}_{n=y_t}]_{n=1}^N$ to encode y_t , e.g., if $y_t = 2$ and $C = 4$, then $c_t = [0, 1, 0, 0]^T$, we define a nonlinear CRF as

$$\begin{aligned}
 P(\mathbf{y}|\mathbf{x}; \theta, \alpha) &= \frac{1}{Z(\mathbf{x}; \theta)} \exp \left\{ \sum_t \theta^\top f(y_t, y_{t-1}, \phi(\mathbf{x}_t; \alpha), t) \right\} \\
 &= \frac{1}{Z(\mathbf{x}; \theta)} \exp \left\{ \sum_t \langle \lambda, c_t \otimes c_{t-1} \rangle + \langle \mu, c_t \otimes \phi(\mathbf{x}_t; \alpha) \rangle \right\} \quad (4)
 \end{aligned}$$

where \otimes is Kronecker product, $\theta = \{\lambda, \mu\}$. In its simplest form, $\phi(\mathbf{x}_t; \alpha)$ is a $Q \times M$ vector by concatenating $\{\phi(x_i; \alpha) | x_i \in (x)_t\}$, where $\phi(x_i; \alpha)$ is a nonlinear

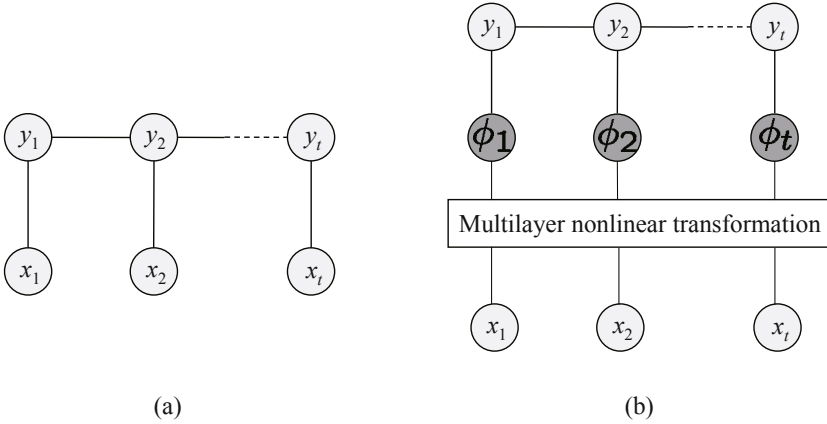


Fig. 2. CRF model (a) and NNCRF model (b)

function $x \rightarrow \mathbb{R}^M$ with parameter α . Then λ is a $C \times C$ parameter vector, and μ is a $C \times Q \times M$ parameter vector. There are many other ways to enrich the model, for example, by defining higher-order features $c_{t-1} \otimes c_t \otimes \phi(\mathbf{x}_t; \alpha)$.

The model (4) is a special nonlinear sequence model which combined the strength of multi-layer neural network (NN) and CRF. As is shown in Figure 2, a CRF is put upon $\phi(x_i; \alpha)$ that amounts to a hidden layer. This model generally optimizes the CRF classifier and hidden-layer features simultaneously. Our model, In fact, our implementation $\phi(x_i; \alpha)$ itself is an neural network

$$\begin{aligned} \phi_i(x_i; \alpha) &= \varrho \left(\sum_{k=1}^H \omega_{i,k}^\phi h_k(x) + b_i^\phi \right), \\ h_k(x) &= \varrho \left(\sum_{j=1}^D \omega_{k,j}^h x_j + b_k^h \right) \end{aligned} \quad (5)$$

where $i = 1, \dots, M$, ϱ is an non-linear (tanh) transfer function, and the parameters α include all the weights ω and bias term b . Therefore, the overall NNCRF hybrid contains two nonlinear hidden layers and one structured-output layer.

The model (4) is different with traditional CRF (1) in its learning process. It not only learns the linear parameters, i.e., λ and μ , but also optimizes the features by tuning the transformation parameter α in transformation function ϕ . This capacity of learning features is important, especially when the task if complex.

In the proposed CRF model, there are two kinds of feature functions which are state functions and transition feature functions. A state function model the dependency of a vertex and a given label. Assume ϕ_t has dimension d , we have $|\mathcal{Y}| \times d$ state functions. There are $|\mathcal{Y}| \times |\mathcal{Y}|$ transition functions and each

Table 1. Orthographical features

Feature Name	Regular expression
ALLCAPS	[A-z]+
INITCAP	^[A-Z].*
CAPMIX	.*[A-Z][a-z].*—.*[a-z][A-Z].*
HAS DASH	.*-.*
PUNCTUATION	[,;?!-+]

corresponds to a hidden state variable pair (y, y') . The transition function is defined as:

$$e_k(y_{t-1}, y_t, \phi, t) = \begin{cases} 1 & \text{if } y_{j-1} = y \text{ and } y_j = y' \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

3.3 Features

We design three kinds of features for the task of expansion recognition given an acronym.

First, we use orthographical features to describe the structure of the target token without considering the query. Such feature tells whether a token contains both upper and lower letters, whether it contains digits and whether it contains special character, etc. We list the orthographical features in table 1.

Second, token-query features are designed to describe the relation between the target token and the given acronym query. For example, whether the first character of the token occurs in the given acronym, whether the token contains an upper letter that occurs in the acronym.

Last but not least, we use context feature to provide more information by considering the neighbor tokens that are important indicators of target token's category. Taking the sentence in figure 1 as an example, we have more confidence that 'Broadcasting' is a part of the expansion of *BBC*, if we have seen the previous token 'British'. We choose three as the context window size, i.e. the target token, the token prior to the target token and the token right after token. We extract the same orthographical features and token-query features for each token in the context window. Furthermore, we also extract the context token-query features which are a composition of token-query features within a context window.

4 Experiments

4.1 Evaluation Corpus

We collect the corpus for evaluation from Wikipedia.org that is a good source for expansion finding. Wikipedia is a free online encyclopedia, representing the

outcome of a continuous collaborative effort of a large number of volunteer contributors. There are a large amount of acronyms and their corresponding definitions, a.k.a, expansions, in the web pages of Wikipedia.org. Moreover, one acronym may have multiple expansions. And it is common that the expansions often appear without their corresponding acronyms in the same sequence, because the author may want to emphasis the phrases.

We randomly select 255 acronyms and crawled the relevant pages. After some necessary processing and manually labeled, the resulted dataset contains totally 255 acronyms, 1,372 distinct expansions, 6,185 expansion sentences, and 108,830 words. Averagely, there are 5.2 distinct expansions for each acronym, 3.11 words for each expansion, and 17.6 words for each expansion sentence.

4.2 Evaluation Criteria

We evaluate our model from two aspects. One is the performance measurements per token to see how well we classify each token in the sequence. The other is the performance measurements per expansion which is a sequence of recognized tokens.

We employ precision, recall and F1 score to evaluate the performance per token. However, there are large portion of examples with label ‘O’ which make the data very unbalanced. So we do not report the average performance measurements over all categories but the precision $PrecPerToken_i$, recall $ReclPerToken_i$ and F1 score $F1PerToken_i$ on each category $y_i \in \{B, I, O\}$ instead. The definitions are:

$$PrecPerToken_i = \frac{\# \text{ of correctly labeled tokens of } y_i}{\text{total } \# \text{ of tokens labeled as } y_i} \quad (7)$$

$$ReclPerToken_i = \frac{\# \text{ of correctly labeled tokens of } y_i}{\text{total } \# \text{ of tokens of } y_i \text{ in corpus}} \quad (8)$$

$$F1PerToken_i = 2 \frac{PrecPerToken_i \cdot ReclPerToken_i}{PrecPerToken_i + ReclPerToken_i} \quad (9)$$

For the calculation of the performance per expansion, we need to first detect the sub-sequence with the predicted label sequence that begins with ‘B’ and ends right before next ‘O’ such as ”B-I-I”. A right label sequence prediction takes place only when predicted label sequence of an expansion completely equal to the true label sequence. Thus the precision, recall and F1 on per expansion are defined as follows:

$$PrecPerAcronym = \frac{\# \text{ of correct expansions found}}{\text{total } \# \text{ of expansions found}}, \quad (10)$$

$$ReclPerAcronym = \frac{\# \text{ of correctly expansions found}}{\text{total } \# \text{ of expansions in corpus}}, \quad (11)$$

$$F1PerAcronym = 2 \frac{PrecPerAcronym \cdot ReclPerAcronym}{PrecPerAcronym + ReclPerAcronym}. \quad (12)$$

4.3 Results

In the experiments, we conduct 5-fold cross-validation to compare the performance of comparison models. We selected the most representative supervised learning models as our baselines which are linear SVM, kernel SVM (RBF), and CRF. For the hyperparameters of each model are tuned on a validation set. For NNCRF model, we use one hidden layer with 15 hidden nodes which performs best during validation.

The experimental results per token are shown in the figure 3 and results on per sequence are shown in table 2.

We can see from figure 3 that label ‘O’ is predicted more accurately than label ‘B’ and ‘I’ by each methods in all evaluation measures, which is resulted by the unbalance of data. The linear SVM model and CRF model produce similar performance. The reason may be that the SVM and CRF have orthogonal ability in classification. The SVM has better generalization by maximizing the margin, while CRF can take advantage of the sequence structure information of the data. Compared to the linear models, both the nonlinear models, kernel SVM and NNCRF, performs better than the linear ones. Moreover, NNCRF which is a nonlinear sequential model achieves the best performance including kernel SVM in all evaluation measures.

The table 2 shows the performance per expansion, which is more directly and precisely reflect the actual effectiveness of each model. From the table 2 we can conclude that: (1) the linear and nonlinear sequence models, CRF and NNCRF, outperform the counterpart non-sequence models SVM and kernel SVM respectively, (2) the nonlinear sequence model NNCRF further improves the performance of CRF and achieves the best performance on the result of per acronym. Based on such experiments, we can conclude that the nonlinear and sequence model NNCRF fits for the acronym expansion recognition task better than the other state-of-the-art compared models.

Figure 4 shows an example of the prediction of the comparison models. It is obviously that the NNCRF achieves the correct prediction of the expansion for the acronym query ‘CCRC’. Although the Kernel SVM achieves one more correct tag than the CRF, it violates the implicit rule that the tag ‘I’ should

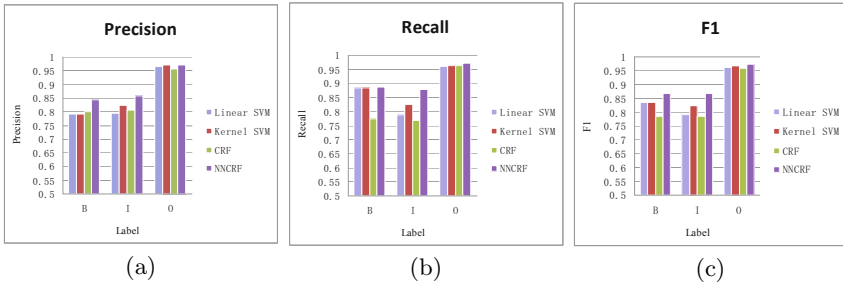


Fig. 3. The results on per label. (a)Precision. (b)Recall. (c)F1 value.

Table 2. Experimental results per acronym

Methods	Precision	Recall	F1
Linear SVM	0.6742	0.7523	0.7106
Kernel SVM	0.6971	0.7311	0.7121
CRF	0.7486	0.7114	0.7284
NNCRF	0.7902	0.8123	0.8012

Query:	CCRC									
Sentence:	<i>City of Cambridge Rowing Club, a rowing club in Cambridge, England.</i>									
True labels:	B	I	I	I	I	O	O	O	O	O
NNCRF:	B	I	I	I	I	O	O	O	O	O
CRF:	O	O	B	I	I	O	O	O	O	O
Kernel SVM:	O	I	B	I	I	O	O	O	O	O

Fig. 4. An example of expansion recognition results of each compared model

never appear right after tag ‘O’. However, the CRF model obeys such rule, which reflect the characteristic of the sequence model.

5 Conclusion

In this paper we propose an expansion finding model for given acronyms. After analyzing the characteristic of the expansion finding problem, we formulate this task as a sequence labeling task. We proposed to use a nonlinear sequence model that combines CRF and the Neural Network. Furthermore, effective features for expansion finding are described in this paper. The experimental results on real data collected from wikipedia.com show that our approach outperform the state-of-the-art algorithms a large margin.

Acknowledgements. This research is supported by the Fundamental Research Foundation for the Central Universities under Grant No.65010571 and the Ph.D. Programs Foundation of Ministry of High Education of China under Grant No.20100031110096.

References

1. Taghva, K., Gilbreth, J.: Recognizing acronyms and their definitions. IJDAR 1(4), 191–198 (1999)
2. Yeates, S.: Automatic extraction of acronyms from text. In: New Zealand Computer Science Research Students’ Conference, pp. 117–124 (1999)
3. Larkey, L.S., Ogilvie, P., Price, M.A., Tamilio, B.: Acrophile: An automated acronym extractor and server. In: Proceedings of the ACM Fifth International Conference on Digital Libraries, DL 2000, Dallas TX, pp. 205–214. ACM Press, New York (2000)

4. Roche, M., Prince, V.: Managing the acronym/expansion identification process for text-mining applications. *Int. J. Software and Informatics*, 163–179 (2008)
5. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: *ICML*, pp. 282–289 (2001)
6. Tasker, B., Pieter, A., Koller, D.: Discriminative probabilistic models for relational data. In: *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI 2002)*, pp. 485–492. Morgan Kaufmann, San Francisco (2002)
7. Peng, F., McCallum, A.: Information extraction from research papers using conditional random fields. *Information Processing & Management* 42(4), 963–979 (2006)
8. Settles, B.: Abner: an open source tool for automatically tagging genes, proteins, and other entity names in text. *Bioinformatics* (April 2005)
9. Sha, F., Pereira, F.: Shallow parsing with conditional random fields (2003)
10. Sato, K., Sakakibara, Y.: RNA secondary structural alignment with conditional random fields. *Bioinformatics* 21(suppl. 2), ii237–ii242 (2005)
11. Liu, Y., Carbonell, J., Weigele, P., Gopalakrishnan, V.: Segmentation conditional random fields (scrfs): A new approach for protein fold recognition. In: *Proc. of the 9th Ann. Intl. Conf. on Comput. Biol (RECOMB)*, pp. 14–18. ACM Press, New York (2005)
12. He, X., Zemel, R.S., Carreira-Perpinan, M.A.: Multiscale conditional random fields for image labeling, vol. 2, II-695–II-702 (2004)
13. Kumar, S., Hebert, M.: Discriminative fields for modeling spatial dependencies in natural images (2003)
14. Lafferty, J., Zhu, X., Liu, Y.: Kernel conditional random fields: representation and clique selection. In: *ICML* (2004)
15. Liu, J., Yu, K., Zhang, Y., Huang, Y.: Training conditional random fields using transfer learning for gesture recognition. In: *ICDM 2010: Proceedings of the 10th International Conference on Data Mining, Sydney, Australia* (2010)
16. Peng, J., Bo, L., Xu, J.: Conditional neural fields. In: Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C.K.I., Culotta, A. (eds.) *Advances in Neural Information Processing Systems*, vol. 22, pp. 1419–1427 (2009)
17. Vapnik, V.N.: *The Nature of Statistical Learning Theory (Information Science and Statistics)*. Springer, Heidelberg (November 1999)
18. Rabiner, L.R.: A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE* 77(2), 257–286 (1989)
19. Ramshaw, L., Marcus, M.: Text chunking using Transformation-Based learning. In: Yarovsky, D., Church, K. (eds.) *Proceedings of the Third Workshop on Very Large Corpora*, Somerset, New Jersey, pp. 82–94. Association for Computational Linguistics (1995)

Leveraging Communication Information among Readers for RFID Data Cleaning

Tao Jiang, Yingyuan Xiao*, Xiaoye Wang, and Yukun Li

Tianjin Key Laboratory of Intelligence Computing and Novel Software Technology,
Tianjin University of Technology, 300384, Tianjin, China
{jiangtaoxyy,yingyuanxiao,xiaoyewang,liyukun.tjut}@gmail.com

Abstract. Radio Frequency Identification (RFID) technologies are used in many applications for data collection. However, raw RFID readings are usually of low quality due to frequent occurrences of false negative, false positive and duplicate readings. A number of RFID data cleaning techniques are proposed to solve the problem. In this paper we explore to use communication information for RFID data cleaning and make RFID readers produce less dirty data at the early stage. First, we devise a reader communication protocol for efficiently utilizing the communication information among readers. Then, the cell event sequence tree with parameters is proposed. Finally, we present three novel RFID data cleaning methods, respectively for duplicate readings, false positive readings and data interpolating. To the best of our knowledge, this is the first work utilizing the communication information among readers in RFID data cleaning. We conduct extensive experiments, and the experimental results demonstrate the feasibility and effectiveness of our methods.

Keywords: RFID, data cleaning, communication information, cell event sequence tree.

1 Introduction

Radio Frequency Identification (RFID) is a promising technology for tracing products and human flows. One of the primary factors limiting the widespread adoption of RFID technology is the unreliability of the data streams produced by RFID readers [1]. A number of RFID data cleaning techniques are proposed to solve the problem [1-13]. However, existing works on RFID data cleaning neglect the communication capability among readers, which is very useful in devising an efficient method for RFID data cleaning. In [14], the authors propose a distributed architecture for scalable private RFID tag identification on the basis of the assumption that all readers have communication capabilities and can exchange information with other readers using a secure channel. Further, many companies have the tentative plan integrating RFID reader into mobile phone, that will make communication among readers more convenience. Motivated by such a potential practicability, in this paper, we make the same assumption like [14], i.e., all the readers have communication capabilities and

* Corresponding author: Tel.: +86-22-60216906; E-mail: yingyuanxiao@gmail.com

can exchange information with other readers. On the basis of this, we explore RFID data cleaning techniques. Specifically, our main contributions can be summarized as follows:

- 1) We explore to use communication information for RFID data cleaning. To the best of our knowledge, this is the first effort to consider it in RFID data cleaning.
- 2) We propose a cell event sequence tree with parameters by means of the reader communication protocol.
- 3) We present three novel RFID data cleaning methods, respectively for duplicate readings, false positive readings and data interpolating.
- 4) We evaluate the performance of the proposed methods through a set of simulation experiments.

The rest of the paper is organized as follows. Section 2 reviews the related work on RFID data cleaning. Section 3 describes the reader communication protocol. The cell event sequences tree and probabilistic cell events are proposed in Section 4. Three novel RFID data cleaning methods are presented in Section 5. Extensive experiments and evaluations are reported in section 6. We conclude this paper in section 7.

2 Related Work

A number of RFID data cleaning techniques have been proposed to clean the input data from readers [1-13]. The data cleaning process for such data is not easily handled by standard data warehouse-oriented techniques, which do not take into account the strong temporal and spatial components of receptor data. Based on the observation, Jeffery et al. [2] present Extensible receptor Stream Processing (ESP), an extensible framework for cleaning the sensor/RFID data streams. ESP is a declarative query processing tool with a pipelined design that is easy to setup and configure for each receptor deployment. In the literature [1], the authors propose SMURF, the first declarative, adaptive smoothing filter for RFID data cleaning. SMURF focuses on a sliding-window aggregate that interpolates for lost readings. SMURF models the unreliability of RFID readings by viewing RFID streams as a statistical sample of tags in the physical world. Considering different anomaly definitions between applications, Rao et al. [3] introduce a deferred approach for detecting and correcting RFID data anomalies, which leverages standardized SQL/OLAP functionality to implement rules specified in a declarative sequence-based language. However, the methods above do not solve the problem of noise and duplicate readings. To compensate these methods, Bai et al [4] propose several Denoising methods and Duplicate Elimination methods. For correcting erroneous RFID raw data, Khoussainova et al. [5] present StreamClean, a system for correcting input data errors automatically using global integrity constraints. However, it is unable to capture all application related prior knowledge and dependency compared with sampling methods [6]. Nevertheless, the work in [6] fails to consider the duplicate readings caused by the overlapped detection regions of RFID readers. Based on the analysis mentioned above, Chen et al. [7] propose a Bayesian inference based approach for cleaning RFID raw data, which takes full advantage of data redundancy. To capture

the likelihood, they design a state detection model, but the correlations among the monitored objects are ignored. Gu et al. [8] propose a data imputation model for RFID by efficiently maintaining and analyzing the correlations of the monitored objects. It is different from our paper which focuses on how to make RFID readers produce less dirty data at the early stage by means of the communication information among readers.

3 Preliminary

In this section, we first describe the application scenario and then formulate the concepts used in this paper. Finally, we devise a reader communication protocol for efficiently utilizing the communication information among readers.

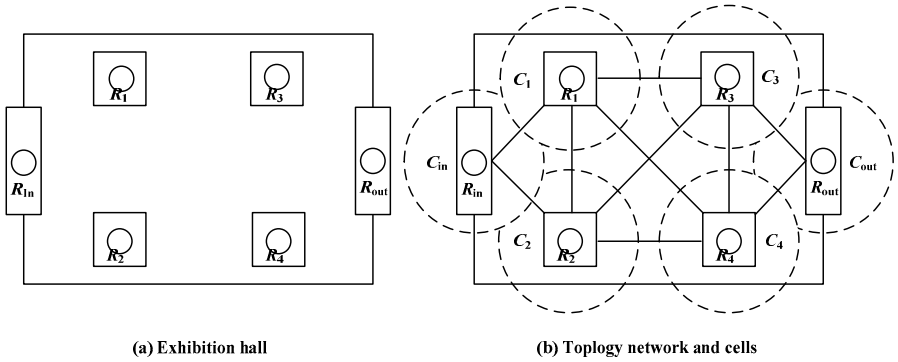


Fig. 1. Illustration of an exhibition hall

3.1 Application Scenario

We choose the exhibition hall as the application scenario. The exhibition hall is illustrated in the Fig. 1(a). The left and right rectangles represent *System Access Point (SAP)* and *System Exit Point (SEP)*, respectively. The squares indicate the exhibition tables, and the small circles denote RFID readers, notated by R_i . We use R_{in} and R_{out} to respectively denote the RFID readers located at *SAP* and *SEP*. Furthermore, the Fig. 1(b) shows the cells and topology network for connecting RFID readers. The large circles with dotted line denote the cells, notated by C_i , and the lines between cells denote the wire or wireless link. Similarly, we use C_{in} and C_{out} to respectively denote the cells covered by R_{in} and R_{out} .

3.2 Notations and Definitions

In this subsection, we introduce the following concepts.

Definition 1. Cell: A cell is defined as the geographical region covered by a reader. The size of a cell depends on the operating frequency of its reader.

Definition 2. Cell Event: We say a cell event happens when a tag enters one cell.

We use $E(ID) = C_i$ to represent the cell event that the tag ID is entering the cell C_i .

Definition 3. Cell Event Sequence: A cell event sequence is defined as the sequence of cell events which were happened by a tag in a time interval.

We use $ES(ID) = C_{i1}C_{i2}\dots C_{ik}$ to denote the cell event sequence where the tag with identifier ID orderly enters C_{i1} , C_{i2} , ..., and C_{ik} .

Definition 4. Full Cell Event Sequence: A full cell event sequence denotes the complete sequence of cell events that a tag experiences from C_{in} to C_{out} .

For example, a tag first enters C_{in} , and then passes through C_2 and C_4 to C_{out} . We say $C_{in}C_2C_4C_{out}$ is a full cell event sequence of the tag.

Definition 5. Independent Sub-Cell Event Sequence: Suppose $ES_i = C_{i1}C_{i2}\dots C_{ik}$, $ES_j = C_{j1}C_{j2}\dots C_{jm}$, and both ES_i and ES_j are sub-sequence of a full cell event sequence. If $C_{i1}=C_{in}$, $C_{jm}=C_{out}$, and $C_{iq} \neq C_{jp}$ holds for any $C_{iq} \in \{C_{i1}, C_{i2}, \dots, C_{ik}\}$, $C_{jp} \in \{C_{j1}, C_{j2}, \dots, C_{jm}\}$, then we say ES_i and ES_j is a pair of independent sub-cell event sequence.

For example, $C_{in}C_2$ and C_4C_{out} is a pair of independent sub-cell event sequence.

3.3 Communication Protocol

In this subsection, we describe the communication protocol illustrated in table 1.

Table 1. Communication Protocol for RFID Readers

Communication Protocol for RFID Readers
1. When a RFID tag enters from SAP , R_{in} is responsible for registering its ID ;
2. R_{in} broadcasts the tag ID and its cell event $E(ID) = C_{in}$ to its adjacent readers R_i to inform them that the RFID tag maybe goes to their cells.
3. When a reader R_i detects the RFID tag entering its cell, R_i expands the cell event $E(ID)$ into the cell event sequence $ES(ID) = C_{in}C_i$ and broadcasts the message $\langle ES(ID), \text{tag } ID \rangle$ to its adjacent readers.
4. When R_{out} detects a RFID tag entering its cell and exiting from SEP , R_{out} broadcasts the message "the tag ID exits" to its adjacent readers. Once the adjacent readers receive message from R_{out} , they forward the message to their adjacent readers.

Specifically, we give an example to illustrate the protocol. For example, in Fig. 1(b), a person with a RFID tag enters into the exhibition hall. Firstly, R_{in} registers the tag ID , and then sends the cell event $E(ID) = C_{in}$ to its adjacent readers R_1 and R_2 . When R_1 detects tag ID , it expands the cell event $E(ID)$ into the cell event sequence $ES(ID) = C_{in}C_1$ and broadcasts the message $\langle ES(ID), \text{tag } ID \rangle$ to its adjacent readers R_{in} , R_2 , R_3 and R_4 . Next, R_3 detects the tag ID and it expands the cell event sequence $ES(ID)$ into $ES(ID) = C_{in}C_1C_3$ and sends the message $\langle ES(ID), \text{tag } ID \rangle$ to its adjacent readers R_1 , R_2 , R_4 and R_{out} . Finally, R_{out} detects the tag ID exiting and it will send

message “ ID in cell C_{out} and exiting the hall” to its adjacent reader R_3 and R_4 , then R_3 and R_4 send the same message to their adjacent readers R_1 and R_2 , and finally to R_{in} .

4 Cell Event Sequence Tree and Probabilistic Cell Events

In this section, we firstly define the cell event sequences tree. Then, we formulate the probabilistic cell events.

4.1 Cell Event Sequence Tree

Definition 6. Cell Event Sequence Tree: In a time interval, the collection of cell event sequences of all RFID tags can be constructed as a tree which is defined as a cell event sequence tree. It is illustrated in Fig. 2.

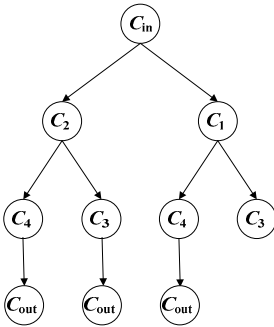


Fig. 2. Illustration of the cell event sequence tree

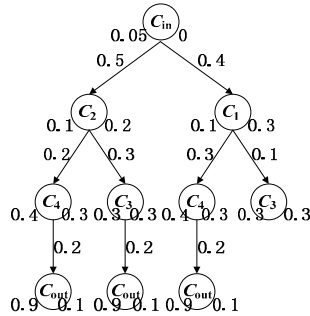


Fig. 3. Cell event sequences tree with parameters

Definition 7. Occurrence Rate of Cell Event Sequence: In time window W_i , the occurrence rate of cell event sequence is defined as the ratio between the occurrence number of a designated cell event sequence $ES(ID)$ and that of all the cell event sequences, which is represented by equation (1):

$$P_O(ES, W_i) = count(ES(ID), W_i) / count(ES(all), W_i). \tag{1}$$

Where $count(ES(ID), W_i)$ denotes the occurrence number of cell event sequence $ES(ID)$ in W_i and $count(ES(all), W_i)$ denotes the total occurrence number of all the cell event sequences in W_i .

Definition 8. Miss Rate in a Cell: In a time window W_i and a given cell C_j , miss rate is the ratio between the number of happened cell events but not be detected and that of happened cell events at the cell, notated by $P_{MC}(C_j, W_i)$.

Definition 9. Stop Rate in a Cell: In a time window W_i and a given cell C_j , stop rate is the ratio between the number of tags stopping at the cell and that of tags passing the cell, notated by $P_{SC}(C_j, W_i)$.

Definition 10. Cell Event Sequence Tree with Parameters: The cell event sequence tree with parameters is a cell event sequence tree storing the corresponding stop rate P_{SC} and miss rate P_{MC} at each cell event of the tree.

Fig. 3 shows a cell event sequence tree with parameters where the node denotes cell event and the real numbers on the left and right side of each node C_i represent the stop rate P_{SC} and the miss rate P_{MC} at cell C_i , respectively. The values on each edge between node C_i and C_j denotes the probability of going through C_i to C_j .

4.2 Probabilistic Cell Events

Suppose that ES_i and ES_j are a pair of independent sub-cell event sequence where $ES_i = C_{i1}C_{i2}\dots C_{ik}$ and $ES_j = C_{j1}C_{j2}\dots C_{jm}$. According to the definition of independent sub-cell event sequence, we can interpolate a cell event sequence ES_k to make $ES_iES_kES_j$ a full cell event sequence. Usually, ES_k satisfied the above condition is not unique. Thus, we may define the top-k probabilistic cell event sequences.

Definition 11. Top-k Probabilistic Cell Event Sequences: Suppose $ES_i = C_{i1}C_{i2}\dots C_{ik}$ and $ES_j = C_{j1}C_{j2}\dots C_{jm}$ are a pair of independent sub-cell event sequence. We define the k cell event sequences ES_1, ES_2, \dots, ES_k with the maximum occurrence of cell event sequence as Top-k probabilistic cell event sequences, where ES_q ($1 \leq q \leq k$) satisfies the condition that $ES_iES_qES_j$ is a full cell event sequence. We use $CE_{Top-k}(ES_i, ES_j)$ to denote the Top-k Probabilistic Cell Event Sequences of ES_i and ES_j .

Definition 12. Maximal Probabilistic Cell Event Sequence: We define the cell event sequence with the maximum occurrence of cell event sequence among Top-k probabilistic cell event sequences as the maximal probabilistic cell event sequence, notated by $CE_{max}(ES_i, ES_j)$.

5 RFID Data Cleaning Strategy

This section presents three new RFID data cleaning methods, which are duplicate data reducing, missing data interpolating, and positive data reducing method, respectively.

5.1 Duplicate Data Reducing Method

To reduce duplicate readings, we transform the form of row RFID readings into $(tagID, C_i, T_{start}, T_{end})$ where $tagID$ denotes the identifier of a tag, C_i denotes the cell which the tag enters into, and T_{start} and T_{end} respectively denote the time at which the tag enters and exits C_i .

In [15] Chaves et al. have an observation that the number of readings of a reader roughly follows a normal distribution $N(\mu, \sigma^2)$. Based on the observation, we employ normal distribution to calculate the time range. In RFID applications, mean value μ is the middle time while a tag is staying in one cell, and variance σ^2 is the deviation of time for readings. The calculation of mean value μ is given by equation (2):

$$\mu = \sum_{i=1}^n T_i / n. \quad (2)$$

Where T_i is the timestamp of each reading for one tag sighted by same reader, and n is the number of timestamps. The calculation of variance σ^2 is given by equation (3):

$$\sigma^2 = \sum_{i=1}^n (T_i - \bar{T})^2 / n. \quad (3)$$

Where \bar{T} is the average time of the readings for one tag. Additionally, the range of independent variable t is $[\mu - 3\sigma, \mu + 3\sigma]$.

Duplicate data reducing method is illustrated in Algorithm 1. Specifically, it first utilizes the normal distribution to extend the timestamp range. Then, it records the related information of $tagID$, C_i , T_{start} and T_{end} . Finally, it transmits the results to users.

To understand the algorithm easily, we give an example. For example, in figure 1(b), a person with a tag ID enters cell C_i at time t_1 , and exits the cell at time t_2 ($t_2 > t_1$). However, reader R_i firstly detects it at $t_1 + a$, and lastly at $t_2 - b$. With the sliding of time window, it firstly records $T_{start} = t_1 + a$, and later $T_{end} = t_2 - b$. Then, it extends the time range with normal distribution $N(\mu, \sigma^2)$. Finally, it gives the results $T_{start} = t_1 + a - \alpha$, and $T_{end} = t_2 - b + \beta$, which is more approximate to the reality. Note that α and β are real numbers correcting the timestamps.

Algorithm 1. Duplicate Data Reducing (D-DR)

Input: raw RFID data streams D_{raw} , initial size of sliding window W_0

Output: duplicate reducing data streams D_{redu}

- 1: $W_1 \leftarrow W_0, T_{start} \leftarrow \text{null}, T_{end} \leftarrow \text{null};$
 - 2: **while** (get next epoch) **do** //sliding window moves ahead a epoch
 - 3: processWindow(W_i); //adaptive processing of the size of sliding window
 - 4: **for** (all $tagID$ in D_{raw}) **do**
 - 5: **if** (first timestamp of $tagID$ in cell C_i is not record) **then**
 - 6: $T_{start} \leftarrow$ timestamp of first reading;
 - 7: **if** (there are readings of $tagID$ in cell C_i) **then**
 - 8: $T_{end} \leftarrow$ timestamp of last reading;
 - 9: $\mu \leftarrow$ equation (2), $\sigma^2 \leftarrow$ equation (3);
 - 10: normalDistribution $N(\mu, \sigma^2)$;
 - 11: $T_{start} \leftarrow$ timestamp of first readings, $T_{end} \leftarrow$ timestamp of last readings;
 - 12: **return** $D_{redu} \leftarrow R(tagID, C_i, T_{start}, T_{end});$
-

5.2 Missing Data Interpolating Method

First, we give the self-learning algorithm of occurrence rate of cell event sequences (see Algorithm 2 for details). Then, we give an example. We assume that there are 20 cell event sequences in time window W_i , and the occurrence time of cell event sequence $C_{in}C_2C_3$ is 6. Thus, the occurrence rate of $C_{in}C_2C_3$ is 0.3.

Based on the cell event sequence tree with parameters, we describe a data interpolating method, named top-k probabilistic data interpolating algorithm (see Algorithm 3 for details). To better understand it, we give an example. In time window W_i , there is a pair of independent sub-cell event sequence, $ES_1 = C_{in}C_2$ and $ES_2 = C_{out}$. To join them into one full cell event sequence, $ES_3 = C_3$ and $ES_4 = C_4$, which are top-2 candidates, are found by means of the cell event sequences tree with parameters, so $ES_1ES_3ES_2$ and $ES_1ES_4ES_2$ are returned as the results.

Algorithm 2. Self-Learning of Occurrence Rate of Cell Event Sequences (SORCES)**Input:** communication information streams IS , initial size of sliding window W_0 **Output:** occurrence rate of cell event sequences P_O

```

1:  $W_1 \leftarrow W_0$ ;
2: while (get next epoch) do //sliding window moves ahead a epoch
3:   processWindow( $W_i$ ); //adaptive processing of the size of sliding window
4:   for (all  $tagID$  in  $IS$ ) do
5:      $P_O(ES, W_i)$ ; // Equation (1)
6:     if ( $P_O(ES, W_i) \neq P_O(ES, W_{i-1})$ ) then
7:        $P_O \leftarrow P_O(ES, W_i)$ ;
8:   return  $P_O$ ;

```

Further, we propose a maximal probabilistic data interpolating algorithm (see Algorithm 4 for details) based on maximal probabilistic cell event sequence. To easily understand algorithm 4, we give an example. In W_i , there is a pair of independent sub-cell event sequence, $ES_1 = C_{in}C_2$, $ES_2 = C_{out}$. To join them into one full cell event sequence, $ES_3 = C_3$, which is the maximal probabilistic candidate, are found by means of cell event sequences tree with parameters, so $ES_1ES_3ES_2$ is returned as the result.

Algorithm 3. Top-k Probabilistic Data Interpolating (Top-kPDI)**Input:** raw RFID data streams D_{raw} , occurrence rate of cell event sequences P_O ,initial size of sliding window W_0 **Output:** interpolating data streams D_{inter}

```

1:  $W_1 \leftarrow W_0$ ;
2: while (get next epoch) do //sliding window moves ahead a epoch
3:   processWindow( $W_i$ ); //adaptive processing of the size of sliding window
4:   for (all  $tagID$  in  $D_{raw}$ ) do
5:     if ( $missDetect() = true$ ) then // detect whether there are missed readings
6:       interpolate  $ES(ID)$  which is in the  $CE_{Top-k}(ES_i, ES_j)$ ; // Notation in definition 11
7:   return  $D_{inter}$ ;

```

Algorithm 4. Maximal Probabilistic Data Interpolating (M-PDI)**Input:** raw RFID data streams D_{raw} , occurrence rate of cell event sequences P_O ,initial size of sliding window W_0 **Output:** interpolating data streams D_{inter}

```

1:  $W_1 \leftarrow W_0$ ;
2: while (get next epoch) do //sliding window moves ahead a epoch
3:   processWindow( $W_i$ ); //adaptive processing of the size of sliding window
4:   for (all  $tagID$  in  $D_{raw}$ ) do
5:     if ( $missDetect() = true$ ) then // detect whether there are missed readings
6:       interpolate  $ES(ID)$  which is in the  $CE_{max}(ES_i, ES_j)$ ; // Notation in definition 12
7:   return  $D_{inter}$ ;

```

5.3 Positive Data Reducing Method

In a given time window W_i , if a tag ID is detected in two or more cells, false positive readings occurs. Further, in the communication information streams, there are two or more readers which detect tag ID . However, staying times in the two cells are not same, so we utilize the staying time of tag ID in cells to confirm which cell is real one.

Algorithm 5. Positive Data Reducing (P-DR)

Input: raw RFID data streams D_{raw} , records $R(\text{tagID}, C_i, T_{\text{start}}, T_{\text{end}})$, initial window size W_0

Output: positive reducing data streams D_{redu}

```

1:  $W_1 \leftarrow W_0$ ;
2: while (get next epoch) do //sliding window moves ahead a epoch
3:   processWindow( $W_i$ ); //adaptive processing of the size of sliding window
4:   for (all  $\text{tagID}$  in  $D_{\text{raw}}$ ) do
5:     if ( $\text{positiveDetect}() = \text{true}$ ) then // detect the false positive readings
6:       if ( $T_{\text{end}, C_i} - T_{\text{start}, C_i} > T_{\text{end}, C_j} - T_{\text{start}, C_j}$ ) then
7:         delete the data of  $\text{tagID}$  from cell  $C_j$ ;
8:   return  $D_{\text{redu}}$ ;

```

We propose a novel positive data reducing method in Algorithm 5. To easily understand it, we give an example. In figure 1(b), when a person with a tag ID stops in cell C_i . However, it is detected by two readers R_i and R_j . Which one is right? Suppose the staying time $T_{\text{stay}, C_i} = 2\text{s}$, and $T_{\text{stay}, C_j} = 0.5\text{s}$, obviously the longer one has bigger probability to be the true one. Thus, we delete readings generated by reader R_j .

6 Experimental Evaluation

In this section, we give experimental evaluations for the proposed methods. Since the communication cost has been described in [14], we only give evaluation results on processing time and accuracy in this paper. The experiments are conducted on the PC with Pentium (R) dual CPU 1.86 GHz and 2GB memory, and the programs are implemented with C++.

6.1 Experimental Setup

We adopt simulated data to conduct the experiments. To ensure the experimental results approximate to the reality, we utilize the Netlogo system, which is a well-known simulator, to generate the simulated data. We simulate an exhibition hall, which has 20 exhibition tables and 1 to 4 RFID readers for each exhibition table. Further, three simulated data are generated in the following scenarios.

Dataset 1: There is only 1 exhibition table due to it is the initial stage, and there is 1 reader fixed at the table. Further, 300 tags go through the cell at the same time.

Dataset 2: There are 10 exhibition tables, in which there only 1 reader at each table, but the noise is different, due to different physical surroundings.

Dataset 3: All the 20 exhibition tables are opened for sightseers, and 2 to 4 readers are fixed in each table. Therefore, there are readers sighting the overlapping regions.

6.2 Evaluation Criteria

In this section, we will give the evaluation criteria for data cleaning algorithms, which are accuracy rate and duplicate reducing rate.

Definition 13. Accuracy Rate: For two given data sets, the reality set D_r and the cleaned set D_c , the accuracy rate is represented by equation (4):

$$P_A = |D_r \cap D_c| / |D_r| \tag{4}$$

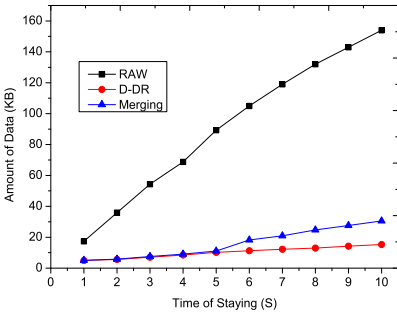
Definition 14. Duplicate Readings Reducing Rate: For two given data sets, the raw data D_{raw} and the duplicate reducing data D_{redu} , the duplicate readings reducing rate is represented by equation (5):

$$P_{redu} = (|D_{raw}| - |D_{redu}|) / |D_{raw}| \tag{5}$$

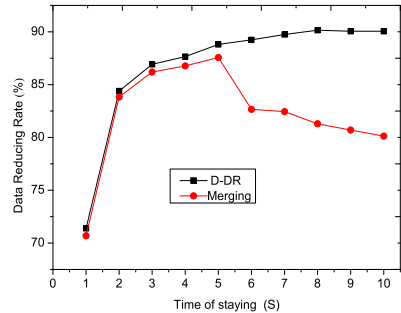
6.3 Evaluations of Duplicate Data Reducing Algorithm (D-DR)

In this test, using Datasets 1, we will compare data amount and data reducing rate of our method D-DR with existing method Merging [4].

As shown in the Fig. 4(a), we can find that the data amount grows linearly with the increasing of staying time. The slope of raw data amount rises sharply and the slope of data amount goes up gently when the algorithms of Merging and D-DR are used. The slopes of the two methods are nearly same before 5s (note that max_distance of Merging is 5s in the test). However, after 5s, the slope of D-DR algorithm still raises, and the slope of Merging algorithm drops sharply. Because Merging algorithm utilizes max_distance as the criteria, which is difficult to be determined. Furthermore, in the Fig. 4(b), we can observe the duplicate data reducing rate of D-DR is bigger than Merging algorithm after 5s. Therefore, D-DR algorithm will substantially cut down the system overhead than Merging algorithm.



(a) Data amount comparison



(b) Data reducing rate comparison

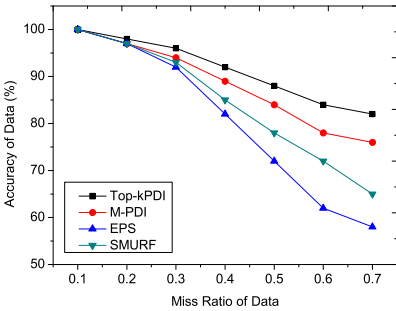
Fig. 4. Comparison of data amount and data reducing rate

6.4 Evaluations of Missing Data Interpolating Algorithm (Top-kPDI, M-PDI)

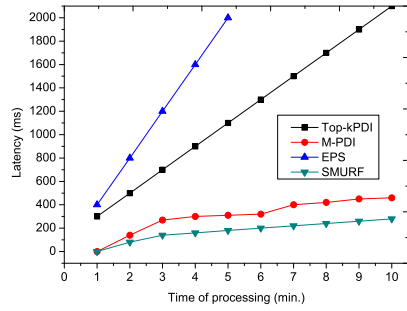
In this experiment, using Datasets 2, we will compare the accuracy and latency of our methods with existing algorithms, which include SMURF [1] and ESP [2].

Fig. 5(a) shows the accuracy of the data processed by these algorithms. When the miss ratio is less than 0.3, the accuracy of these algorithms is nearly same. However, with the increasing of miss ratio, the performance of Top-kPDI and M-PDI behaves better than the other two, because our methods are based on probabilities that learning from the communication information. In addition, Top-kPDI performs better than M-PDI as the latter only searches the maximal result rather than the top-k results.

In the Fig. 5(b), we compare our algorithms with SMURF and EPS on the latency aspect. SMURF is best in respect to latency among four methods, and ESP shows the worst real-time performance, because ESP is a pipeline which has five stages that will consume more time. The reason that M-PDI is not as well as SMURF lies in that M-PDI not only processes the size of sliding time windows but also calculates the probabilistic of interpolated candidates. Similarly, Top-kPDI consumes more time than SMURF, since it considers many cases for better accurate.

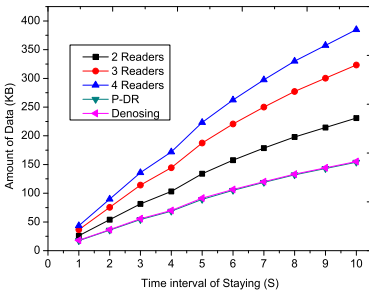


(a) Accuracy comparison

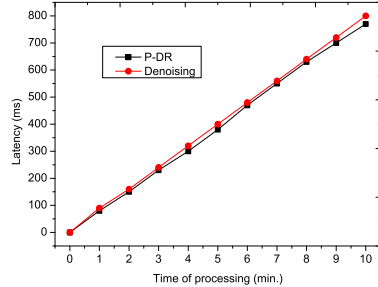


(b) Latency comparison

Fig. 5. Comparison of accuracy and latency



(a) Data amount comparison



(b) Latency comparison

Fig. 6. Comparison of data amount and latency

6.5 Evaluations of Positive Data Reducing Algorithm (P-DR)

In this experiment, using Datasets 3, we will compare the proposed P-DR with existing algorithm Denoising [4] on data amount and latency.

The Fig. 6(a) illustrates the raw data amount, which is generated from an activity that 300 tags go through a cell that installs 2, 3, 4 readers respectively.

It can be seen from Fig. 6(a) that the data amount grows proportionally with the increasing of RFID reader number, and it also rises linearly with the increasing of staying time in cells. We can also learn that the data amount goes down dramatically when processed by P-DR and Denoising. Furthermore, P-DR and Denoising nearly show the same good behaviors, because they both utilize the strategies to eliminate duplicate data, although their strategies are not same. In Figure 6(b), we compare the latency of P-DR and Denoising on the processing time. The two algorithms also show nearly the same latency.

7 Conclusion

In this paper, we address the problem of dirty readings over RFID data streams. On basis of the assumption that RFID readers can communicate with each other using wireless or wired networks in time, we devise a reader communication protocol. Then we present a cell event sequence tree with parameters. Based on them, we propose an active RFID data cleaning strategy, which includes duplicate data reducing method, missing data interpolating method and positive data reducing method. We conduct extensive experiments, and the experimental results demonstrate the feasibility and effectiveness of our methods. Our work is the first study to address the communication information among readers for RFID data cleaning. Thus, our work is complementary to the existing research on RFID data cleaning.

References

1. Jeffery, S.R., Garofalakis, M., Franklin, M.J.: Adaptive Cleaning for RFID Data Streams. In: Proceedings of VLDB, pp. 163–174. ACM Press, Seoul (2006)
2. Jeffery, S.R., Alonso, G., Franklin, M.J., Hong, W., Widom, J.: A Pipelined Framework for Online Cleaning of Sensor Data Streams. In: Proceedings of ICDE, pp. 140–142. IEEE Press, Atlanta (2006)
3. Rao, J., Doraiswamy, S., Thakkar, H., Colby, L.S.: A Deferred Cleansing Method for RFID Data Analytics. In: Proceedings of VLDB, pp. 175–186. ACM Press, Seoul (2006)
4. Bai, Y., Wang, F., Liu, P.: Efficiently Filtering RFID Data Streams. In: VLDB Workshop on CleanDB. ACM Press, Seoul (2006)
5. Khoussainova, N., et al.: Towards Correcting Input Data Errors Probabilistically Using Integrity Constraints. In: Proceedings of MobiDE, pp. 43–50. ACM Press, Chicago (2006)
6. Xie, J., Yang, J., Chen, Y., Wang, H., Yu, P.S.: A Sampling-Based Approach to Information Recovery. In: Proceedings of ICDE, pp. 476–485. IEEE Press, Cancun (2008)
7. Chen, H., Ku, W.S., et al.: Leveraging Spatio-Temporal Redundancy for RFID Data Cleansing. In: Proceedings of SIGMOD, pp. 51–62. ACM Press, New York (2010)

8. Gu, Y., et al.: Efficient RFID Data Imputation by Analyzing the Correlations of Monitored Objects. In: Zhou, X., Yokota, H., Deng, K., Liu, Q. (eds.) DASFAA 2009. LNCS, vol. 5463, pp. 186–200. Springer, Heidelberg (2009)
9. Jeffery, S.R., Alonso, G., et al.: Declarative Support for Sensor Data Cleaning. In: Fishkin, K.P., Schiele, B., Nixon, P., Quigley, A. (eds.) PERVASIVE 2006. LNCS, vol. 3968, pp. 83–100. Springer, Heidelberg (2006)
10. Kanagal, B., et al.: Online Filtering, Smoothing and Probabilistic Modeling of Streaming Data. In: Proceedings of ICDE, pp. 1160–1169. IEEE Press, Cancun (2008)
11. Gonzalez, H., Han, J., Shen, X.: Cost-Conscious Cleaning of Massive RFID Data Sets. In: Proceedings of ICDE, pp. 1268–1272. IEEE Press, Istanbul (2007)
12. Franklin, M.J., Hong, W., et al.: Design Considerations for High Fan-in Systems: The HiFi Approach. In: Proceedings of CIDR, pp. 290–304. ACM press, California (2005)
13. Cocci, R., Tran, T., et al.: Efficient Data Interpretation and Compression over RFID Streams. In: Proceedings of ICDE, pp. 1445–1447. IEEE Press, Cancun (2008)
14. Agusti, S., Josep, D.F., Antoni, M.B., Vanesa, D.: A Distributed Architecture for Scalable Private RFID Tag Identification. *Computer Networks* 51, 2268–2279 (2007)
15. Chaves, L.W.F., Buchmann, E., Böhm, K.: Finding Misplaced Items in Retail by Clustering RFID Data. In: Proceedings of EDBT, pp. 501–512. ACM Press, New York (2010)

Web Article Quality Assessment in Multi-dimensional Space*

Jingyu Han, Xiong Fu, Kejia Chen, and Chuandong Wang

School of Computer Science and Technology,
Nanjing University of Posts and Telecommunications
Nanjing, China 210003

hjysky@gmail.com, fux@njupt.edu.cn, ke.jia@gmail.com,
chdwang@njupt.edu.cn

Abstract. Nowadays user-generated content (UGC) such as Wikipedia, is emerging on the web at an explosive rate, but its data quality varies dramatically. How to effectively rate the article's quality is the focus of research and industry communities. Considering that each quality class demonstrates its specific characteristics on different quality dimensions, we propose to learn the web quality corpus by taking different quality dimensions into consideration. Each article is regarded as an aggregation of sections and each section's quality is modelled using Dynamic Bayesian Network(DBN) with reference to accuracy, completeness and consistency. Each quality class is represented by three dimension corpora, namely accuracy corpus, completeness corpus and consistency corpus. Finally we propose two schemes to compute quality ranking. Experiments show our approach performs well.

1 Introduction

Over the past decade, web 2.0 has radically changed the way of generating information. For example, in Wikipedia every internet user actively generates, modifies and publishes data according to his/her own understanding and knowledge. The online collaborative work naturally raises a question: how to control the quality of the content which is constantly being contributed by various users. It differs from traditional publication in that the contributors are usually not professional and the quality can not be guaranteed. Therefore, assessing and promoting the data quality with minimal human interpretation is an imminent concern. Here data quality means how good the data is in terms of quality dimensions such as accuracy, completeness, consistency, etc.

We ground our work on Wikipedia as it is a typical example of collaborative content repository and growing in popularity [1]. The quality of Wikipedia articles demonstrates great uncertainty. The fundamental policy of quality rating falls on human judgement. However, this kind of approach has two drawbacks.

* This work is fully supported by National Natural Science Foundation of China under grant No.61003040, 60903181.

First, given the huge size of collaborative content, the manual assessment will eventually cease to be feasible. Second, human judgement is subject to bias. To overcome the drawbacks, a possible solution is to design automatic or semi-automatic quality assessment policies [2,3,4,5,6]. But until now seldom work is given on how to quantify web article quality in terms of quality dimensions such as accuracy, completeness and consistency.

Actually Wikipedia quality classes including Featured Article(FA), Good Article(GA), B-Class(B), C-Class(C), Start-Class(ST) and Stub-Class(SU)¹ can not distinguish clearly from each other as they have some overlapped features. Therefore, to precisely predict each article's quality class, we argue that articles' quality class should be identified in their own quality subspace, namely the space constituted by the quality dimensions on which quality classes differ from each other sharply.

Based on above observations, we propose to learn Wikipedia quality corpus for each class by aggregating component sections' dimension values, which is further used to given an article's quality ranking. Our approach works as follows.

1. First, each article is divided into sections and each section is a relatively complete semantic unit. Each section's quality evolution is modelled by Dynamic Bayesian Network(DBN) in terms of commonly accepted dimensions, namely accuracy, completeness and consistency. Thus, we can get the dimension value of the last version for each section.
2. Second, by aggregating the final dimension values of its component sections, an article's quality is represented as a distribution over all dimensions. Thus, quality corpus for each class is determined over all quality dimensions.
3. Finally, an article's quality ranking is determined by comparing itself with each quality corpus. To perform the ranking, we first propose a Global Ranking(GR), which regard each dimension uniformly among all quality classes. Furthermore, a Local Adaptive Ranking(LAR) is proposed, which can distinguish quality classes in their local quality subspace more effectively.

Until now a lot of work has been done on how to assess Wikipedia article's quality. Literature [1,5,6,7] give quantitative methods to evaluate Wikipedia article's quality but they do not use editing history to assess quality. The work closely relevant to ours is using edit history to assess the trustworthiness of articles [8,9]. But they do not touch on how to assess quality in multi-dimensional quality space.

2 Problem Setting and Preliminary

Given an article P with an editing history $\langle P_0, P_1, P_2, \dots, P_i, \dots, P_n \rangle$ where

1. n refers to the total number of revisions during the whole life and version P_i results from a revision R_i applied to previous version P_{i-1} .
2. $P_i \neq P_{i+1}$ for each $i \in [0, n - 1]$.

¹ http://en.wikipedia.org/wiki/Wikipedia:Version_1.0_Editorial_Team/Assessment

The revision R_i includes three types of operations, namely insert (I), delete (D), update (U). Operations of delete, insert, and update are related to two aspects, namely *content* and *formatting* of articles. The *content* includes text, images, and links. The *formatting* includes HTML tags or CSS, and Wikipedia templates.

Our goal is to give P a quality rank, denoted as $Rank(P) = \langle l_1, l_2, \dots, l_m \rangle$, based on its editing history. Here l_i ($1 \leq i \leq m$) is a quality class.

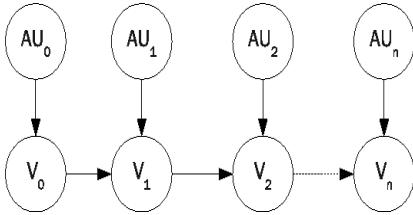


Fig. 1. Evolution Modelled by DBN

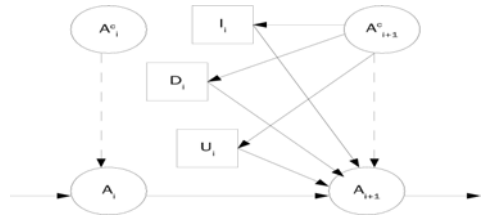


Fig. 2. Accuracy Evolution from V_i to V_{i+1}

We analyse the quality in terms of widely accepted dimensions including accuracy, completeness and consistency [10,11] and they mean as follows.

Definition 1 (Accuracy). Accuracy (denoted as A) is the extent to which data are correct, reliable and free of error.

Definition 2 (Completeness). Completeness (denoted as C) is the extent to which information is not missing and is of sufficient breadth and depth.

Definition 3 (Consistency). Consistency (denoted as CON) is the extent to which information is presented in the same format and compatible with previous data.

3 Modelling Dimension Evolution of Sections

The quality of an article P is determined by its component section $V \in P$. Actually the section’s dimension value of one version not only depends on its previous version but also on current contributor’s work and it also exhibits great uncertainty during this process. Thus, we employ Dynamic Bayesian Network (DBN) to describe quality dimension’s evolution over versions. DBN is defined by a pair (B_s, B_t) , where B_s is the graph structure of network and B_t corresponds to the set of transition functions. As illustrated by figure 1, version V_i results from operations of current contributor AU_i on its previous version V_{i-1} and thus quality dimension A_i (C_i, CON_i) is determined by A_{i-1} (C_{i-1}, CON_{i-1}) and AU_i ’s dimension factor.

To describe conditional transition probability and contributor’s dimension factor, a common approach is to assume normality and model them with Gaussian distributions. In this paper, we chose beta distributions because dimension values range from 0 to 1, where beta distributions are also defined.

$$beta(x|\alpha, \beta) = \begin{cases} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}x^{\alpha-1}(1-x)^{\beta-1} & \text{for } 0 < x < 1, \alpha, \beta > 0 \\ 0 & \text{elsewhere} \end{cases} \quad (1)$$

where $\Gamma(k)$ is a Gamma function. The mean and the variance of beta distribution are given by

$$u = \frac{\alpha}{\alpha + \beta} \text{ and } \sigma^2 = \frac{\alpha\beta}{(\alpha + \beta)^2(\alpha + \beta + 1)}. \quad (2)$$

Beta distribution takes on many different shapes depending on α and β . We make a assumption that the variance should neither be too small nor too large when u is close to 0.5. On the other hand, if u is close to 0 or 1, variance does not make much difference in the choice of α and β as it is bounded by $u(1 - u)$. Given this α is learned by sampling and β is set as

$$\beta = \frac{\alpha - \alpha u}{u}. \quad (3)$$

3.1 Contributor’s Quality Dimension Factors

Contributors are divided into four groups, namely administrator (adm), registered contributor (reg), anonymous contributor (ano) and blocked contributor (blo). Different types of contributors have their specific dimension factors with reference to accuracy, completeness and consistency. We learn the contributors’ dimension factors by heuristic methods.

Intuitively if the content generated by one contributor has not been updated or the percentage of updated content is smaller than a predefined threshold in subsequent operations, it means the contributor’s accuracy factor is high. The contributor’s completeness and consistency factors are also heuristically determined and we do not address them in detail due to space.

3.2 Modelling Accuracy Evolution

Figure 2 illustrates the accuracy transition from V_i to V_{i+1} , where A_i and A_{i+1} represent the accuracy of version V_i and V_{i+1} , A_{i+1}^c represents the accuracy factor of contributor, I_i , D_i and U_i denotes the set of insert, delete and update operations on version V_i . The arrows in the figure 2 state that the accuracy of version V_{i+1} depends not only on that of previous version V_i , but also on the operations of D_i , I_i , U_i affected by contributor’s accuracy factor A_{i+1}^c .

We seek to determine the probability density function of A_n , denoted as $f(A_n)$, which is fully determined by $f(A_0|A_0^c)$ and $f(A_{i+1}|A_i, A_{i+1}^c, I_i, D_i, U_i)$. The dependency between A_0 and A_0^c is non-deterministic due to casual factors

such as editing context and personal knowledge. We assume it takes beta distribution

$$f(A_0|A_0^c = a_0) = \text{beta}(x|\alpha_0^{acc}, \beta_0^{acc}). \quad (4)$$

We take a_0 as u_0 , thus α_0^{acc} and β_0^{acc} are determined by equation 3.

We now consider how to determine A_{i+1} given its previous version's accuracy A_i and contributor's accuracy factor A_{i+1}^c . We also assume that it is a beta distribution

$$f(A_{i+1}|A_i, A_{i+1}^c, I_i, D_i, U_i) = \text{beta}(x|\alpha_{i+1}^{acc}, \beta_{i+1}^{acc}) \quad (5)$$

Based on equation 3, α_{i+1}^{acc} and β_{i+1}^{acc} are dependent on u_{i+1}^{acc} . The following details how to estimate parameter u_{i+1}^{acc} .

Whenever a contributor edits some part of the section, he usually reviews the adjacent part [1]. Thus V_i is divided into revision portion and non-revision portion. The former captures the quality dimension in terms of contributor's authorship while the latter takes the reviewship into consideration.

Accuracy of Revision Portion. The accuracy of revision portion R_i is directly determined by I_i , D_i and U_i . The following gives how to quantify the incremental accuracy for insert, delete and update operations respectively.

- **Insert.** The incremental accuracy due to I_i is $|I_i|A_{i+1}^c$, where $|I_i|$ denotes the size of inserted content by AU_{i+1} .
- **Delete.** The incremental accuracy due to D_i is $-|D_{i+1}|A_i$, where $|D_i|$ denotes the size of deleted content.
- **Update.** This case is regarded as a deletion followed by an insertion. Thus the incremental accuracy is $|U_i|(A_{i+1}^c - A_i)$, where $|U_i|$ is the size of updated content.

Thus the incremental accuracy due to R_i is

$$R_i^{acc} = |I_i|A_{i+1}^c - |D_i|A_i + |U_i|(A_{i+1}^c - A_i) \quad (6)$$

Accuracy of non-Revision Portion. When a contributor is editing some portion of a section, the adjacent portion undergoes review. Intuitively, the more close a word $s_l \in V_i - R_i$ is to R_i , the more accurate it is. Its reviewship accuracy is denoted as $s_l.review$.

Based on above analysis, the mean accuracy u_{i+1}^{acc} is an average of revision portion's accuracy and non-revision portion's accuracy, namely

$$\frac{R_i^{acc} + \sum_{s_l \in (V_i - R_i)} |s_l.review|}{|V_{i+1}|}. \quad (7)$$

3.3 Modelling Completeness Evolution

To determine the completeness distribution of last version, denoted as $g(C_n)$, we need to determine $g(C_0|C_0^c)$ and $g(C_{i+1}|C_i, C_{i+1}^c, I_i, D_i, U_i)$, where C_i , C_{i+1}^c denote the completeness of version V_i and the completeness factor of contributor AU_{i+1} respectively.

The Initial Completeness Probability Density Function. We assume that it takes a beta distribution $g(C_0|C_0^c = c_0) = \text{beta}(x|\alpha_0^{com}, \beta_0^{com})$ where c_0 is the completeness factor of initial contributor. Similarly α_0^{com} and β_0^{com} is determined by taking c_0 as u_0 .

Compute the Conditional Density Functions. We assume that each transition $g(C_{i+1}|C_i, C_{i+1}^c, I_i, D_i, U_i)$ also takes a beta distribution, namely $\text{beta}(x|\alpha_{i+1}^{com}, \beta_{i+1}^{com})$. u_{i+1}^{com} is needed to determine α_{i+1}^{com} and β_{i+1}^{com} . The syntax completeness measure is combined with contributor’s inherent completeness factor to compute u_{i+1}^{com} . The former is the quantitative measures with reference to completeness, which include the total number of internal links (denoted as *ilinks*), the total number of internal broken Links (denoted as *blinks*) and length in words(*len*) [5]. The latter denotes the inherent completeness tendency due to user’s expertise, commitment, etc.

We believe that the earlier the portion is added to one section, the more marginal completeness it offers. This is due to the two reasons. First, the contributors tend to fill up the most obvious semantic gap facing a topic being discussed. The earlier the portion is inserted, the more semantic gap it covers. Second, the later the portion is added, the more overlap it will incur. In view of this, we use a monotonically increasing function as equation 8 to model how u_{i+1}^{com} depends on both syntax completeness measures and contributor’s completeness factors.

$$u_{i+1}^{com} = 1 - \frac{1}{B_{i+1}^\theta} \tag{8}$$

This function targets 1 but never reaches it. Here θ is a parameter ranging from 0 to 1 tuned by experiment. B_{i+1} is the completeness base which ranges from 1 to infinity. The completeness base B_{i+1} for version $i + 1$ is computed as

$$\text{pow}\left(\frac{1}{1 - C_i}, \frac{1}{\theta}\right) + I_{i+1}^{inc} + D_{i+1}^{inc} + U_{i+1}^{inc}. \tag{9}$$

Here I_{i+1}^{inc} , D_{i+1}^{inc} , U_{i+1}^{inc} are the incremental completeness bases with reference to insert, delete and update respectively. Following this, u_{i+1}^{com} is determined.

3.4 Modelling Consistency Evolution

Consistency denotes the extent to which information is presented in the same format and compatible with previous data. We regard consistency as syntax consistency adjusted by contributor’s inherent consistency factor. For wikipedia article, syntax consistency is measured by template consistency and HTML consistency [5].

Similar to accuracy and completeness, the evolution of consistency is characterized by initial distribution function and the conditional density function. We also assume that they take beta distributions. That is to say, $h(CON_0|CON_0^c) = \text{beta}(x|\alpha_0^{con}, \beta_0^{con})$ and $h(CON_{i+1}|CON_i, CON_{i+1}^c, I_i, D_i, U_i) = \text{beta}(x|\alpha_{i+1}^{con}, \beta_{i+1}^{con})$ hold. To determine $\text{beta}(x|\alpha_{i+1}^{con}, \beta_{i+1}^{con})$, we need to obtain u_{i+1}^{con} , which is determined by both the syntax consistency and the contributor’s consistency factor. Thus u_{i+1}^{con} is defined as

$$\frac{CON_i * |V_i| + CON_{i+1}^{inc-I} + CON_{i+1}^{inc-D} + CON_{i+1}^{inc-U}}{|V_{i+1}|} \tag{10}$$

where CON_{i+1}^{inc-I} , CON_{i+1}^{inc-D} , CON_{i+1}^{inc-U} are the incremental consistency for insert, delete and update respectively.

4 Ranking Articles

An article’s quality ranking is achieved through two steps. First, class corpus is constructed for each quality class. Then, article is compared with all the quality class corpora to determine its ranking.

4.1 Extracting Corpus for Each Quality Class

Definition 4 (Quality Class Corpus). *Each quality class corpus consists of three quality dimension with reference to accuracy, completeness and consistency, denoted as $\Omega^l = \{\Omega_{acc}^l, \Omega_{com}^l, \Omega_{con}^l\}$ ($l \in \{FA, GA, \dots, SU\}$).*

Suppose there are N article $\{P_1, P_2, \dots, P_N\}$ in training set for quality class l , the dimension corpus Ω_j^l is

$$\Omega_j^l = \frac{\sum_{i=1}^N cor_j(P_i)}{N} \tag{11}$$

where $j \in \{acc, com, con\}$. Note here $cor_j(P_i)$ is article’s dimension corpus for dimension j , which is defined as follows.

Definition 5 (Article Dimension Corpus). *Given an article P_i , its article dimension corpus with reference to dimension j is the average density functions over all its sections. That is to say,*

$$cor_j(P_i) = \frac{\sum_{sec \in P_i} beta^{sec}(x|\alpha_n^j, \beta_n^j)}{total_{sec}} \tag{12}$$

where $beta^{sec}(x|\alpha_n^j, \beta_n^j)$ is the beta distribution of the last version of $sec \in P_i$ with reference to dimension j and $total_{sec}$ is the total number of sections in P_i .

4.2 Classifying an Article into Quality Class

Given an article with article corpus $cor(P) = \{cor_{acc}(P), cor_{com}(P), cor_{con}(P)\}$, it is compared with all quality corpora $\{\Omega^{FA}, \Omega^{GA}, \dots, \Omega^{SU}\}$ to determine its quality rank. We propose Global Ranking (GR) and Local Adaptive Ranking (LAR) to give article ranking. They are detailed in the followings.

Global Ranking. In this approach each dimension weights the same for all quality classes. In other words, the quality rating is computed in a uniform high-dimensional space for all quality classes. The ranking proceeds as follows. First, the weight for quality dimension j , denoted as w^j , is $\frac{GAD_j}{\sum_{j=1}^3 GAD_j}$ where GAD_j is global average distance for quality dimension j . It is defined as

$$GAD_j = \frac{2 * \sum_{k=1}^m \sum_{n=k+1}^m \sqrt{\frac{1}{2} \int (\sqrt{\Omega_j^k} - \sqrt{\Omega_j^n})^2 dx}}{m(m-1)} \tag{13}$$

where m is the total number of quality classes. Second, the similarity between article P and quality class Ω^l , denoted as $sim(P, l)$, is computed. Based on the similarities, quality ranking of P is given as $rank(P) = \langle l_1, l_2, \dots, l_m \rangle$ satisfying $sim(P, l_i) \geq sim(P, l_{i+1})$ (for $1 < i < m$).

Algorithm 1. localAdaptiveRanking

Input: All quality class corpora $\mathcal{Y} = \{\Omega^1, \dots, \Omega^m\}$, an article corpus $cor(P) = \{cor_1(P), cor_2(P), cor_3(P)\}$, threshold factor r

Output: $Rank(P)$

```

1  curSet ← Y, LAD ← ∅;
2  foreach quality dimension j do
3    LADj ← computeLocalAverageDistance(curSet); LAD ← LAD ∪ LADj;
4  end
5  simSet ← ∅;
6  foreach quality class i ∈ curSet do
7    sim(P, i) ← computeSim(P, Ωi, LAD); simSet ← simSet ∪ sim(P, i);
8  end
9  Rank(P) ← computeRank(simSet);
   ambSet ← selectAmbiguousSubset(simSet, r);
10 if ambSet = ∅ then
11   return Rank(P);
12 end
13 else
14   oriSubRanking ← ranking of ambSet according to Rank(P);
15   reSubRanking ← localAdaptiveRanking(ΓambSet, cor(P), r);
16   if oriSubRanking = reSubRanking then
17     return Rank(P);
18   end
19   else
20     Rank(P) ← mergeRank(Rank(P), ambSet, reSubRanking);
21     return Rank(P);
22   end
23 end

```

Local Adaptive Ranking. To get a more precise ranking of an article, the dimensions’ weights are tuned to the subset of quality classes. In other words,

the dimension weighs different for different subset of quality classes. To achieve this, Local Average Distance (LAD) is defined as follows.

Definition 6 (Local Average Distance). *Given a subset of quality classes $L = \{\Omega^1, \dots, \Omega^w\}$, local average distance (LAD) in L with reference to dimension j , denoted as LAD_j , is*

$$\frac{\sum_{k=1}^w \sum_{n=k+1}^w \sqrt{\frac{1}{2} \int (\sqrt{\Omega_j^k} - \sqrt{\Omega_j^n})^2 dx}}{w(w-1)} \quad (14)$$

The quality ranking for an article is computed with algorithm [1](#). The algorithm works in an iterative fashion. If a subset of quality classes fall in a narrow range from the article, the weights for all dimensions are adjusted among the involved classes and the similarities are recomputed. If the adjusted ranking for subset of quality classes are different from the original ranking, it is adjusted. Otherwise, it remains. Here *selectAmbiguousSubset* is used to select the similarities which fall in a range, which is detailed in algorithm [2](#).

Algorithm 2. selectAmbiguousSubset

Input: $simset = \{sim_1, sim_2, \dots, sim_w\}$, similarity threshold factor r

Output: subset of similarities that are close to each other

```

1   $minDif \leftarrow 1, maxDif \leftarrow 0;$ 
2  for  $i \leftarrow 1$  to  $w - 1$  do
3    for  $j \leftarrow i + 1$  to  $w$  do
4      if  $|sim_i - sim_j| > maxDif$  then
5         $maxDif \leftarrow |sim_i - sim_j|$ 
6      end
7      if  $|sim_i - sim_j| < minDif$  then
8         $minDif \leftarrow |sim_i - sim_j|$ 
9      end
10   end
11 end
12  $retSet \leftarrow \emptyset;$ 
13 for  $i \leftarrow 1$  to  $w - 1$  do
14   for  $j \leftarrow i + 1$  to  $w$  do
15     if  $|sim_i - sim_j| < r * (maxDif - minDif)$  then
16        $retSet \leftarrow retSet \cup \{i, j\}$ 
17     end
18   end
19 end
20 return  $retSet;$ 

```

5 Experiment

We collected a set of English articles from computing category in July 2010. We chose this set of articles because the articles have been assigned quality

class labels according to wikipedia editorial team’s quality grading scheme. The performed experiment had two goals. First, to analyse how well our proposed approach can give an article its quality ranking. Second, to compare our approach with the state-of-the-art of the works. To measure how our ranking approaches can cover the correct rating, we borrow $p@n$ metric in information retrieval as follows.

$$p@n = \frac{\sum_{i=1}^N tag_i^n}{N} \tag{15}$$

where N is the total number of articles to be ranked. Here tag_i^n is equal to 1 if the top n ranking for article i covers the correct class. Otherwise, it is set to 0.

5.1 Effectiveness of Our Ranking Approaches

To show the performance of quality assessment in multi-dimensional space, we conducted the experiment using five quality ranking schemes. In addition to Global Ranking (GR) and Local Adaptive Ranking(LAR), we also rank quality based on single dimension, namely accuracy (denoted as ACC), completeness (denoted as COM) and consistency (denoted as CON) respectively. Figure 3 and figure 4 summarize the results in terms of $p@1$ and $p@2$ respectively.

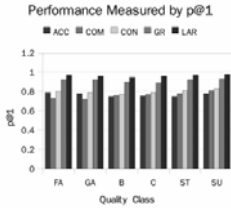


Fig. 3. Performance Measured by p@1

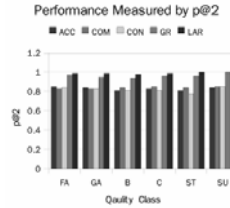


Fig. 4. Performance Measured by p@2

Based on $p@n$ measures, we can observe that GR and LAR approaches perform much better than the approaches using single dimensions. Specifically, based on $p@n$ measures GR and LAR outperform ACC, COM, CON by a percentage ranging from 20 to 32. This verifies that different quality dimensions complement each other, thus producing a better performance. We can also note that LAR approach consistently has a better performance than GR, with an improvement ranging from 5 to 9 regarding $p@n$ measures. This is due to that in local subspace a more proper weights are tuned for different dimensions, which generate more effective distinctiveness.

In addition we observed that the LAR approach is sensitive to similarity threshold factor r . Figure 5 shows the average $p@1$ and $p@2$ measures among all quality classes with reference to different values of r . On average the $p@n$ measures first increase rapidly then decrease slowly with the increase of r . If r is too small, the chance for adjusting ranking is very little. To the extreme, if r

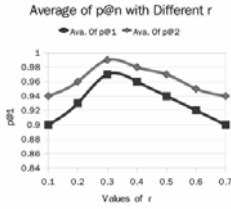


Fig. 5. Average $p@n$ with Reference to r

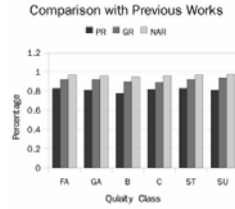


Fig. 6. Comparison Based on $p@1$

is reduced to zero, LAR works the same as GR. However, if r is too large, the weights for different dimensions tend to change little because a large proportion of classes are involved. In this case it can not effectively distinguish different quality classes either. In our experiment, r is tuned to be 0.31 to perform best.

5.2 Comparisons with Previous Works

We compare our method with the state-of-the-art work proposed in literature [1]. We compare our work with ProbReview (denoted as PR) approach as it was reported to perform best among its three kinds of approaches. PR is based on the mutual dependency between article quality and contributor authority and takes partial reviewship of contributors into consideration. We compare our GR and LAR approaches with PR approach based on $p@1$ measure. As PR gives a total order of all the articles based on $NDCG@k$ measure and a total order $FA > GA > B > C > ST > SU$ is assumed. Therefore we computed $p@1$ measure from the total ordering given by PR as follows. Suppose that the article set is composed of N articles and $\sum_{i=1}^6 N_i = N$ holds and here N_1, N_2, \dots, N_6 denotes the numbers of articles belonged to FA, GA, B, C, ST, SU class respectively. Suppose that the total order given by PR approach for N articles is $\langle P_1, P_2, \dots, P_N \rangle$, it naturally means that the first N_1 articles are regarded FA articles, the subsequent N_2 articles are regarded as GA articles, and so on. Thus the $p@1$ measure can be obtained. Figure 6 gives the comparison result based on $p@1$ measure. Apparently, GR outperforms PR by an average of 10 percentage and LAR approach outperforms PR by a percentage between 15 and 23.

6 Conclusion

In this paper, we investigate how to rate Wikipedia article quality in a multi-dimensional space. We propose to learn the quality corpus for each quality class in terms of quality dimensions including accuracy, completeness and consistency. In sum, the contribution of this paper is as follows. First, We propose to quantify web article quality in a multi-dimensional space where different dimensions can complement each other, thus producing a better ranking performance. Second, a

concrete model describing the evolution of dimensions including accuracy, completeness and consistency is detailed. Third, two quality ranking approaches, namely GR and LAR, are proposed.

As our future work, we would further consider how to include other dimensions in our modelling and how to determine the relationship among different dimensions to improve quality ranking performance, etc.

References

1. Hu, M., Lim, E.P., Sun, A.: Measuring article quality in wikipedia: Models and evaluation. In: Proc. of the sixteenth CIKM, pp. 243–252 (2007)
2. Aebi, D., Perrochon, L.: Towards improving data quality. In: Proc. of the International Conference on Information Systems and Management of Data, pp. 273–281 (1993)
3. Wang, R.Y., Kon, H.B., Madnick, S.E.: Data quality requirements analysis and modeling. In: Proc. of the Ninth International Conference on Data Engineering, pp. 670–677 (1993)
4. Pernici, B., Scannapieco, M.: Data quality in web information systems. In: Spaccapietra, S., March, S.T., Kambayashi, Y. (eds.) ER 2002. LNCS, vol. 2503, pp. 397–413. Springer, Heidelberg (2002)
5. Stvilia, B., Twidle, M.B., Smith, L.C.: Assessing information quality of a community-based encyclopedia. In: Proc. of the International Conference on Information Quality, pp. 442–454 (2005)
6. Rassbach, L., Pincock, T., Mingus, B.: Exploring the feasibility of automatically rating online article quality (2008)
7. Dalip, D.H., Cristo, M., Calado, P.: Automatic quality assessment of content created collaboratively by web communities: A case study of wikipedia. In: Proc. of JCDL 2009, pp. 295–304 (2009)
8. Zeng, H., Alhossaini, M.A., Ding, L.: Computing trust from revision history. In: Proc. of the 2006 International Conference on Privacy, Security and Trust: Bridge the Gap Between PST Technologies and Business Services (2006)
9. Zeng, H., Alhossaini, M.A., Fikes, R., McGuinness, D.L.: Mining revision history to assess trustworthiness of article fragments. In: Proc. of International Conference on Collaborative Computing: Networking, Applications and Worksharing, pp. 1–10 (2009)
10. Pipino, L.L., Lee, Y.W., Wang, R.Y.: Data quality assessment. *Communications of the ACM* 45(4), 211–218 (2002)
11. Batini, C., Cappiello, C., Francalanci, C., Maurino, A.: Methodologies for data quality assessment and improvement. *ACM Computing Surveys* 41(3), 1–52 (2009)

DRScribe: An Improved Topic-Based Publish-Subscribe System with Dynamic Routing

Guohui Li and Sheng Gao*

School of Computer Science and Technology, Huazhong University of Science and Technology,
Wuhan, P.R. China
guohuili@mail.hust.edu.cn, gaosheng@smail.hust.edu.cn

Abstract. As information sharing and news dissemination flourish, Scribe, a classic topic-based publish-subscribe system, gains popularity recently. In Scribe, events are delivered to subscribers through distributed multicast trees, which should be continuously maintained to guarantee that no subscribers miss events. However, redundant event deliveries and overstaffed multicast trees are introduced. In this paper, we proposed a new topic-based publish-subscribe system based on Scribe, namely DRScribe, to reduce the costs of event dissemination and multicast tree maintenance. DRScribe adopts Bloom filters to check the subscriptions of the neighbor nodes, and dynamically routes the next hop by means of the neighbors' subscriptions. The maintenance interval can be tuned according to the level of a node in the multicast tree. A series of experiments is conducted to demonstrate that DRScribe greatly reduces redundant event deliveries and helper nodes of multicast trees, and the dissemination and maintenance costs decrease significantly.

Keywords: Topic-based pub/sub systems, Scribe, Multicast tree, Bloom filter, Dynamic routing.

1 Introduction

How to select up-to-date and interesting data from the ocean of increasing information is a major issue in peer-to-peer networks. Publish/subscribe interaction schema (pub/sub, for short) is one of the most popular solutions. Pub/sub is a messaging paradigm in which publishers (data/events producers) disseminate messages (also called as events) to subscribers (data/events consumers) in large-scale distributed networks. With the development of information dissemination services, pub/sub systems based on peer-to-peer networks become an active research area.

In the pub/sub system, a publisher specifies a class or values of a set of attributes for an event, and then disseminates it to other nodes in peer-to-peer networks. Note that, publishers are not predetermined to send the events to specific subscribers. On the other hand, subscribers express their interests into classes or attributes, and only receive

* Corresponding author: School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, Hubei, 430074, PR China. Tel.: +86 13971479933.

events of interest. Both publishers and subscribers are unaware of the existences and the behaviors of each other. The decoupling and decentralization of pub/sub systems make the network topology scalable and dynamic.

At present, there are two major kinds of pub/sub systems: the topic-based pub/sub systems [1-3], such as Scribe [1], Bayeux [2], Content Addressable Network (CAN) Multicast [4]; and the content-based pub/sub systems [5-7], such as MEDYM [6], Siena [7]. Note that, our paper mainly focuses on the topic-based pub/sub system.

1.1 Topic-Based Pub/Sub Systems

In the topic-based system, messages/events are published to topic groups or logical channels. Subscribers subscribe the topics/channels that they are interested in, and receive all events that belong to the subscribed topics. In many modern topic-based pub/sub systems like Scribe, Bayeux, the events in each topic are delivered to the subscribers via a maintained and distributed data structure (i.e., a multicast tree) over DHT (Distributed Hash Table) overlays [8-10]. Another representative kind of topic-based system like CAN Multicast, which is built on top of CAN [10], adopts routing tables of CAN to flood the published events to all nodes in a CAN overlay network, and creates a separate CAN overlay for each topic.

Because the peers may join and leave frequently in the peer-to-peer network, the multicast tree should be continuously updated; otherwise subscribers will miss some events. Hence, the dissemination and delivery of events incurs the maintenance cost of the multicast tree besides the actual dissemination cost (which depends on the event frequency). Note that, the maintenance costs, which are the additional overheads for the pub/sub system, can be significant especially when a pub/sub system supports a large number of topics and subscribers with a low event frequency.

1.2 Motivation

Topic-based pub/sub systems like Scribe and Bayeux maintain a multicast tree for each topic over DHT overlays. Although the pub/sub system with multicast trees has few redundant events and little delivery latency, the major problem is the high overheads to maintain the multicast trees. As nodes join or leave freely, and subscribers subscribe or unsubscribe in the peer-to-peer environment, the multicast trees will be disrupted without continuously maintenance. To reduce the maintenance overheads of multicast trees, the reference [11] clusters the topics with the similar sets of subscribers into a topic-cluster, which is associated with a new multicast tree. A cost-benefit analysis is adopted to dynamically merge or split multicast groups. However, the cost-benefit analysis is nontrivial to compute although the cost function is simplified in the cost-benefit model. Meanwhile, the reference [12] proposes a probabilistic solution to deliver events to subscribers without maintaining multicast trees, however, variant of random walks for event delivery introduce redundant events and high delivery latency.

In this paper, we propose a new topic-based publish-subscribe system based on Scribe, namely DRScribe (Dynamic Routing Scribe), to reduce the costs of event dissemination and multicast tree maintenance. The main contributions of DRScribe are

as follows: (1) A space-efficient probabilistic data structure, Bloom filter, is introduced to maintain the subscriptions of the successors for each DRSubscribe node. (2) Dynamic routing is adopted to choose the best next hop according to the subscriptions of the successors, and then the helper nodes of the multicast tree are minimized. (3) The maintenance interval is tuned according to the level of a node in the multicast tree. Specifically, the maintenance interval grows as the level of a node increases. (4) During each runtime of maintenance, the multicast tree is dynamically adjusted according to the new subscriptions of successors, and the multicast tree of DRSubscribe has less helper nodes than that of Scribe. Hence, the maintenance cost of multicast trees and the redundant event delivery to helper nodes will be both reduced.

In the rest of the paper, Section 2 briefly describes the background. Section 3 presents the design and implementation of DRSubscribe. Section 4 gives the performance evaluation and experimental results. Conclusions are given in Section 5.

2 Background

In this section, we first present a representative distributed hash table (DHT), namely Chord [8], and then introduce a classic topic-based pub/sub system, i.e., Scribe, which works on the DHT overlay.

Chord provides the basic lookup/route operation *lookup (key)*, which maps a *key* to the nodes responsible for the key. Firstly, each node and key are assigned an m -bit identifier arranged from 0 to 2^m-1 using a consistent-hash function, denoted as *hash*, where m depends on the chosen consistent-hash function and the scale of peer-to-peer networks. Typically, m is set to be 64, 128 or 160. Suppose K is the m -bit identifier of *key*, i.e., $K=hash(key)$. In the Chord circle, the nodes whose identifiers immediately follow K in a clockwise direction, namely successors of K , are responsible for K and the number of successors is generally set to be 16. Meanwhile, each node has one predecessor node, whose identifier immediately precedes the node. Each node, say n , maintains a routing table consisting of a finger table and a successor list. The finger table is composed of the IDs and IP addresses (and port numbers) of nodes that follow the node n at power-of-two distances in the identifier circle (i.e., $2^i \mid 0 \leq i < m$). The successor list contains all the successors of the node n .

Algorithm *lookup (key)* works as follows: when a node n issues a lookup with a key k , the successor list of n is checked firstly. If the key k is in the interval $[n.ID, n.lastsuccessor.ID]$, there must exist a successor s_i responsible for k . Otherwise, n looks up its finger table, and finds out a node f_j whose ID most immediately precedes k , then asks f_j for the node whose ID is closest to k . By repeating this process $O(\log N)$ times, the node n finally knows which node is closest to k and responsible for k .

Scribe manages the operative layer using the APIs of the DHT overlay, such as Pastry [9]. Scribe uses Pastry to locate an active node that serves as the rendezvous node (also called the channel peer) with the key of the topic, and all the relevant nodes subscribe the topic to the rendezvous node. Subscribing a topic or joining a group will form a path from the subscriber to the rendezvous node. As a result, a distributed multicast tree rooted at the rendezvous node is formed, which consists of all the routing paths from the

subscribers to the rendezvous node. Besides the rendezvous node and subscribers, there exist helper nodes in the multicast tree. Helper nodes reside in the routing paths from the subscribers to the rendezvous node and assist in disseminating the topic events.

In Scribe, multicast trees are created by the reverse path forwarding approach, and the heartbeat detection method is adopted to maintain the completeness of the multicast trees. The basic idea is as follows: each parent node periodically sends a heartbeat message to its children. If an active child node n detects its parent is faulty (i.e., n doesn't receive the heartbeat message), n will get a new active parent by anew subscribing the topic. Thus, the broken multicast tree is repaired.

3 System Design

In this section, we mainly focus on the design and implementation of DRScribe. As mentioned above, event dissemination, subscription management and multicast tree maintenance are the critical issues in Scribe. In order to reduce the maintenance overheads and eliminate redundant event deliveries as far as possible, our solution takes advantage of dynamic routing and variable maintenance interval strategies.

3.1 Subscription Installation and Management

In this section, how a participant subscribes a topic is detailed. Firstly, we introduce Bloom filter to store the subscriptions of successors. Bloom filter is a space-efficient probabilistic data structure, which is used to check whether an element is a member of a set with a probabilistic misjudgment, i.e., *false positive*. A bloom filter is a fixed-size bit vector with a set of independent hash functions.

In our proposed dynamic routing strategy, each node maintains a Bloom filter for local subscription and l Bloom filters for l successors' subscriptions, where l is the number of the successors, since each node needs to know about the subscriptions of its successors. In the following, Algorithm 1 and 2 detail the process of subscribing a topic (i.e., joining a group). For the presentation convenience, table 1 lists the items that each DRScribe node maintains, and table 2 lists the types of messages (not the published events) used in DRScribe system.

Table 1. Items that a DRScribe node maintains

Item Name	Type (in C++ STL)	Description
subscriptions	<i>Set<Topic></i>	The set of local subscriptions.
localBloomfilter	<i>Bloomfilter</i>	The bloom filter that contains local subscriptions
successorBFs	<i>Vector<Bloomfilter></i>	the subscription records for each successor in the form of Bloom filter
children	<i>Map<Topic,Set<IP>></i>	The map of the topic to the IP Address set of the corresponding topic's children.
parents	<i>Map<Topic,IP></i>	The map of the topic to the IP Address of its parent. i.e., cached IP addresses of the next hop.
latestHeartbeat	<i>Map<Topic,Time></i>	The timestamp of the latest received heartbeat for each topic

Table 2. The description of the message types in the algorithms

Message Type	Description
SUBSCRIBE	Request for joining the group/channel of the topic
UNSUBSCRIBE	Request for leaving the group/channel of the topic
SUBSCRIBENOTIFY	The subscriber notified its predecessors with new subscription
PARENTADJUST	A parent moves its children to one of its successors
HEARTBEAT	A parent periodically sends heartbeat messages to the children of related channel.
EVENTDELIVERY	A parent forwards an event to the children of related channel.

Algorithm 1 details how a participant p_1 subscribes a topic t (i.e., p_1 joins group t). Firstly, p_1 asks the underlay Chord to look up the rendezvous node of topic t , and then sends a SUBSCRIBE message to the first node p_2 along the route created by the *lookup()* operation (i.e., $p_2 = \text{nexthop}(t)$, see line 1 – line 5), and then p_1 notifies its predecessor node p_3 to update the corresponding Bloom filter of p_3 (by sending a SUBSCRIBENOTIFY message to p_3)(see line 6 – line 8). At last, local records of p_1 are updated (see line 9 – line 11).

```

Algorithm 1. subscribe (Topic t)
1.     next= nexthop(t);
2.     msg.sender = me.ip;
3.     msg.type = SUBSCRIBE;
4.     msg.topic = t;
5.     send( msg, next.ip);
6.     msg.type = SUBSCRIBENOTIFY;
7.     msg.forwards = 0;
8.     send(msg, predecessor.ip );
9.     subscriptions.insert( t );
10.    parents[t] = next.ip;
11.    localBloomfilter.insert( t );

```

Once p_2 (the next hop of p_1) receives a SUBSCRIBE message or p_3 (the predecessor of p_1) receives a SUBSCRIBENOTIFY message from p_1 , Algorithm 2 is invoked by p_2 or p_3 respectively. In the following, we focus on p_2 firstly. When p_2 receives the SUBSCRIBE message from p_1 , p_2 checks whether p_2 is currently a forwarder or not (see line 3 – line 7); if so, p_1 is inserted into the corresponding children list of p_2 . Otherwise, p_2 queries the Bloom filters of its successors to check whether a successor has subscribed the topic or not. If there exists such a successor, p_2 sends a PARENTADJUST message to the successor (see line 8 – line 14). Otherwise, p_2 creates a children entry for this topic and inserts p_1 into this entry, and then sends a SUBSCRIBE message to the next hop and builds a link to the multicast tree (see line 15 – line 20).

Secondly, when p_3 (the predecessor of p_1) receives a SUBSCRIBENOTIFY from p_1 , p_3 updates the corresponding Bloom filter and then forwards the SUBSCRIBENOTIFY message to its predecessor. This process repeats as above until all the nodes whose successors contain p_1 get the SUBSCRIBENOTIFY message (see line 24 – line 28).

If a node p_i receives a PARENTADJUST message from a node p_j (such as p_2 above), p_i will replace p_j as the new parent of the original child of p_j , and notify the child to update its local records (see line 21 – line 23).

The subscription strategy of DRScribe will decrease the size of multicast tree by avoiding helper nodes as far as possible, although our DRScribe strategy introduces some extra costs of the subscription installation. However, the extra costs are far less than the reduced costs of event dissemination and multicast tree maintenance, which is demonstrated in the section of experimental evaluation.

Algorithm 2. messagehandler(Message m)

```

1.      switch (m.type){
2.          case SUBSCRIBE:
3.              if ( children.contains(m.topic) ||
4.                  subscriptions.contains(m.topic) ) {
5.                  children[m.topic].insert( m.sender );
6.                  exit(0);
7.              }
8.              foreach ( Bloom filter f in successorBFs ){
9.                  if ( f.contains( m.topic ) ) {
10.                     m.type = PARENTADJUST;
11.                     send(m, ip of corresponding successor of f );
12.                     exit(0);
13.                 }
14.             }
15.             next = nexthop(m.topic);
16.             m.sender = me.ip;
17.             send(m,next.ip );
18.             children.insert( m.topic);
19.             children[m.topic].insert( m.sender )
20.             parents[m.topic] = next.ip;
21.         case PARENTADJUST:
22.             children[m.topic].insert( m.sender );
23.             Notify the child m.sender to update its parent;
24.         case SUBSCRIBENOTIFY:
25.             successorBFs[m.sender].insert( m.topic );
26.             if ( ++ m.fowards < CHORD_PARAM_SUCCESSORS ){
27.                 send(m, predecessor.ip );
28.             }
29.     }
```

Once a node has joined the peer-to-peer network, it immediately pulls successors' local Bloom filters over the successor links, and takes less cost than that in directly pulling successors' subscriptions records. That is to say, Bloom filter is not only efficient in local storage but also efficient in bandwidth costs. A node updates the bloom filter of its successor after it receives a SUBSCRIBENOTIFY message (see line

24 – line 28 in Algorithm 2). If a node unsubscribes a topic, it should not only send a UNSUBSCRIBE to its corresponding parent (similar to that of Scribe), but also notify its predecessors to refresh their corresponding successor Bloom filters.

3.2 Event Dissemination

The event dissemination of DRSubscribe is similar to that of Scribe. To notify the subscribers on a topic update, the publisher node routes the topic to the rendezvous node of the topic's multicast tree, and asks it to return its IP address. The publisher node caches this IP address to avoid repeated routings through Chord. If the rendezvous node changes or fails, the publisher node will find a new rendezvous node and refresh the cached IP address.

The publisher sends the event to the rendezvous node. This event is then disseminated through the multicast tree. There is a single multicast tree for each topic and all multicast trees use the above way to multicast events to all subscribers [1].

3.3 Multicast Tree Maintenance

The multicast tree maintenance has great influence on the overall cost of Scribe, especially when the event frequency is very low. Meanwhile, the cost of multicast tree maintenance depends on the adopted maintenance interval. For each topic, the maintenance for its underlying tree (i.e., multicast tree) is performed periodically in every Mt minutes, which is termed as the maintenance interval of the Scribe system. A selection of the maintenance interval with the best possible performance can be an intricate task. A short maintenance interval will induce a high maintenance cost of a multicast tree, while a long maintenance interval will result in that some failed nodes are not be repaired timely and their children might miss the following events.

1) Variable Maintenance Interval

In Scribe, FMI (fixed maintenance interval) model is adopted, where the maintenance interval is a fixed value for each node in the multicast tree. However, the invalid nodes in different levels have different effects on the event coverage. See the left part of Fig. 1, if the node B fails, the nodes E, F, J, K, M, O and Q (notes that G, I and N are helper nodes) will miss the following events before repairing the tree next time. Meanwhile, if the node I fails, only the node M misses the following events. Obviously, the failed node on the upper level of the tree has more impact than that on the lower level does. Hence, we introduce Variable Maintenance Interval (VMI for short) model, where Mt , a fixed interval in FMI, is replaced by a function $Mt(l)$, where l is the level of the node in the tree (Note that the level of the rendezvous node is 1), and $Mt(l)$ increases with the growth of l . Let d denote the depth of the multicast tree. Suppose that $Mt(1) < Mt$, the inequality $Mt(l) > Mt$ holds only if $l > l_b$ ($1 \leq l_b < d$), and $Mt(1), Mt(2), \dots, Mt(d)$ is a linear or logarithmic growth progression. Here l_b is a turning point, Note that, the value of l_b depends on the scale of the network and the growth rate of the function $Mt(l)$ depends on the event frequency of the topic. Because the invalid node in the upper level has more impact on the event coverage, and its maintenance interval is less than Mt , the

invalid node in the upper level will be repaired sooner. Therefore, the event coverage of VMI model is better than that of FMI model under the same scenarios.

In the FMI model, the maintenance cost is directly proportional to the number of non-leaf nodes, and inversely proportional to the changeless Mt . However, in VMI model, the maintenance interval of the node increases with the growth of the level of the node. If the level of a node is upper than or equal to l_b , the maintenance interval (i.e., $Mt(l)$) will be less than Mt (the fixed interval in FMI model) and the maintenance cost will be larger than that of FMI model. On the other hand, if the level of a node is lower than l_b , the maintenance interval will be larger than Mt and the maintenance cost will be less than that of FMI model. Meanwhile, it is obvious that the lower the level, the more the non-leaf nodes at the level. Hence, the maintenance costs occurring at lower levels have more impact on the overall maintenance costs than that at the upper levels. The overall maintenance costs in VMI model are less than that in FMI model since the lower levels in VMI model have less maintenance costs than that in FMI model does.

A node might reside in several multicast trees, and it adopts a separate maintenance interval for each tree according to its level in that tree. If a child has the same parent in different trees, the parent just need to send heartbeat to the child in one tree, not each of the trees. Such combination could reduce unnecessary costs.

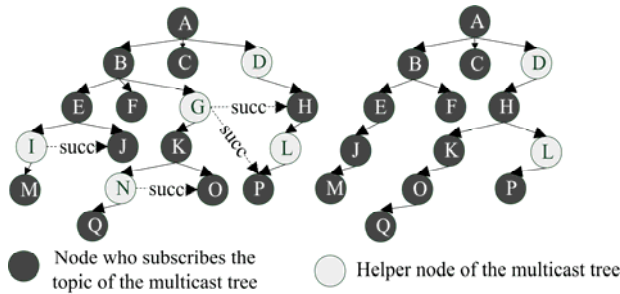


Fig. 1. The maintenance of a simple multicast tree of Scribe (left) and DRScribe(right)

2) Dynamic adjustment of multicast tree

During the maintenance of a multicast tree, new nodes might subscribe the topic and join the multicast tree at any time. As shown in Fig.1, suppose that the nodes J, O, H and P (L is a helper node) subscribe the topic in the last maintenance cycle, and J, O, H (and P) are the successors of I, N, G respectively, the multicast tree of the Scribe system is similar to the left of Fig.1. However, in DRScribe, the helper node I could find that its successor node J subscribes the topic by scanning its self-maintained Bloom filter, and make sure that I and J are at the same level of the multicast tree, and then I will notify the node M to re-subscribe to J . At last, I will quit the multicast tree, and the node N and G will quit the multicast tree similarly. Comparing Scribe (left in Fig.1) and DRScribe (right in Fig.1), the optimized multicast tree of DRScribe has less helper nodes than that of Scribe, and has less maintenance cost and reduces the event dissemination cost, since redundant event deliveries to helper nodes are minimized.

3.4 Additional Explanations

At last, we emphasize the following three advantages of DRSubscribe: (1) DRSubscribe does not break the load balance of the multicast tree during the maintenance, since the adjustment only happens at the same level (e.g., in Fig.1, $A \rightarrow B \rightarrow G \rightarrow K$ is adjusted to $A \rightarrow D \rightarrow H \rightarrow K$). That is, the depth and structure of the multicast tree in DRSubscribe have not been destroyed, and the balance will not be destroyed. On the contrary, if the node G notifies K to re-subscribe to P , not H (P and H are both the successors of G), the levels of the nodes K , O , Q and the depth of the tree will increase by 2, which might break the balance and increase the latency of the events delivered from the rendezvous node to the nodes K , O and Q . (2). The number of the children of a node can not grow unboundedly. We introduce a threshold to control the node congestion. (3) At last, the proposed DRSubscribe is adaptable to other DHTs, such as Pastry, which is essentially a prefix-based routing protocol. The main process of extending DRSubscribe into Pastry is as follows: the neighborhood set and the leaf set of a Pastry node will take the place of the successor nodes of a Chord, and assist DRSubscribe to route dynamically. The optimization of multicast trees is the same with that of Chord.

4 Experimental Evaluation

The proposed DRSubscribe approach was evaluated and compared with the Scribe in terms of bandwidth cost, and delivery success rate. We implemented the above two approaches and conducted extensive experiments. In the following, we summarized the configurations of experiments, and then gave experimental results.

4.1 Experimental Setup

We implemented Scribe and DRSubscribe approaches on top of *sgaosim*, which is a discrete-event packet/message level simulator developed by us. The simulator *sgaosim* consists of underlay layer, overlay layer and application layer. The underlay layer represents the network layer, where nodes manager, events manager, global observers are all located. The overlay layer adopts Chord protocol. Chord has a configuration named Proximity Neighbor Selection (*PNS*) to reduce the lookup latency, and the number of samples of the *PNS* fingers was fixed at 16. For the application layer, two Pub/Sub approaches, Scribe and DRSubscribe, were implemented respectively in our simulator. We adopted the E2EGraph as the network topology and the simulated network consisted of 1740 nodes, which were derived from the kingdata¹. The average latency between each two nodes was 90 milliseconds.

DRSubscribe was evaluated and compared with Scribe in terms of the following performance metrics: the bandwidth cost and the delivery success rate. The bandwidth cost is the network transmission cost to deliver events to all of their subscribers, including the bandwidth cost to subscriptions management and multicast trees

¹ The kingdata is available on <http://pdos.csail.mit.edu/p2psim/kingdata/mking-t>

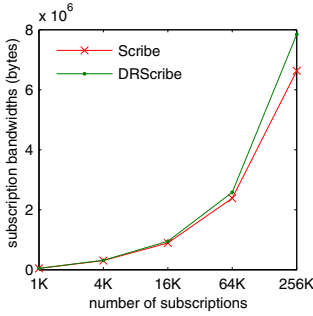


Fig. 2. Subscription costs

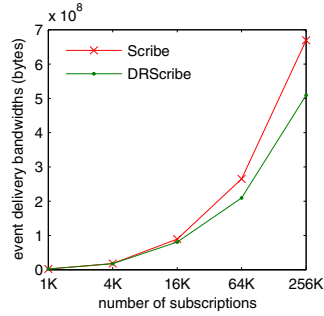


Fig. 3. Event delivery costs

maintenance. The delivery success rate is the rate of event coverage, which could be represented as the ratio of the number of the received events to the number of events expected to be received.

4.2 Experimental Results

In our experiments, we measured DRScribe and compared with Scribe under different scenarios in terms of the subscription management cost, event delivery cost and multicast tree maintenance cost and then evaluated the delivery success rate.

Table 3. Configurations of the experiments

Item	Value	Item	Value
the finger table base in Chord	2	the total number of topics	240
the number of successors in Chord	16	the size in bytes of the event (same size with a Twitter message)	140
The maintenance interval in Scribe	1min.	the size in bytes of the subscription installation message	16
total simulation time	2 hours	the size in bytes of the heartbeat message	16
node crash rate	25%	node unsubscribe rate	20%

Firstly, we evaluated DRScribe and Scribe in terms of bandwidth cost under different number of subscriptions (1K, 4K, 16K, 64K, or 256K) when the number of published events was fixed at 4K. Fig. 2 shows that the total cost of the subscription installation increased exponentially with the exponential growth of subscriptions. As mentioned in section 3.4, DRScribe incurred an extra cost of subscription installation compared with Scribe. However, the extra cost (about 10^6 bytes) was far less than the reduced costs (about 10^8 bytes) of event disseminations and multicast tree maintenances as shown in Fig. 3 and Fig. 4.

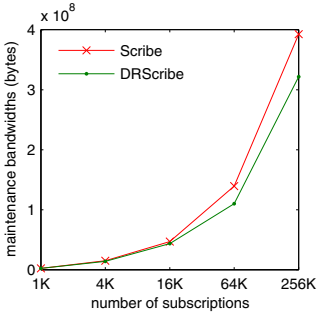


Fig. 4. Maintenance costs

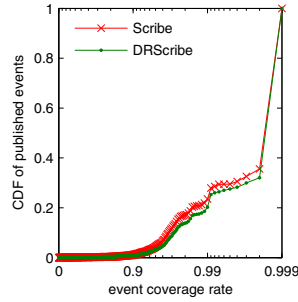


Fig. 5. CDF of events

As shown in Fig. 3 and Fig. 4, the total event delivery cost and the total maintenance cost of multicast trees increased with the growth of subscriptions. When the number of subscriptions was larger than 16K, DRscribe had a lower event delivery cost and maintenance cost than those of Scribe. That is to say, the probability of a node successfully finding a subscriber from its successors list increased with the growth of subscriptions. Therefore, the multicast tree could be optimized soon, and the number of helper nodes of the multicast tree was minimized. Hence, the number of redundant event deliveries to helper nodes was reduced.

Fig. 5 shows CDF (cumulative distribution function) of published events with respect to the success delivery rate. Most events (above 99.95%) had a success rate [0.999, 1], and more than 35% of events had a success rate of greater than 0.998. Meanwhile, the event coverage rates under DRscribe and Scribe were very similar, that is, DRscribe did not decrease the event coverage rates while reducing the bandwidth.

5 Conclusions and Future Work

In this paper, we propose DRscribe to reduce the maintenance overheads and eliminate redundant event deliveries as far as possible. DRscribe adopts Bloom filters to check the subscriptions of the neighbor nodes, and dynamically routes the next hop by means of the neighbors' subscriptions. Our preliminary experimental results are encouraging, which show that DRscribe has lower cost than Scribe. For future works, we shall further focus on the quantitative calculation of Variable Maintenance Interval and implemented DRscribe over other DHT overlays (such as Pastry).

Acknowledgement. This work was supported by National Natural Science Fund of China under grant 60873030 and Doctoral Fund of Ministry of Education of China under grant 20090142110023.

References

1. Castro, M., Druschel, P., Kermarrec, A., Rowstron, A.: SCRIBE: A Large-scale and Decentralized Application-level Multicast Infrastructure. *IEEE Journal on Selected Areas in communications (JSAC)*, 1489–1499 (2002)

2. Zhuang, S.Q., Zhao, B.Y., Joseph, A.D., Katz, R.H., Kubiawicz, J.: Bayeux: An Architecture for Scalable and Fault-Tolerant Wide-Area Data Dissemination. In: Proc. 11th International Workshop Network and Operating System Support for Digital Audio and Video, NOSSDAV 2001 (2001)
3. Ramasubramanian, V., Peterson, R., Sireer, E.G.: Corona: A High Performance Publish-subscribe System for the World Wide Web. In: Proc. of Networked System Design and Implementation, San Jose, California (2006)
4. Ratnasamy, S., Handley, M., Karp, R., Shenker, S.: Application-level Multicast Using Content-addressable Networks. In: Crowcroft, J., Hofmann, M. (eds.) NGC 2001. LNCS, vol. 2233, pp. 14–29. Springer, Heidelberg (2001)
5. Fabret, F., Jacobsen, H.A., Llibat, F., Pereira, J., Ross, K.A., Shasha, D.: Filtering Algorithms and Implementation for Very Fast Publish/Subscribe Systems. In: Proc. ACM SIGMOD, pp. 115–126 (2001)
6. Cao, F., Singh, J.P.: MEDYM: An Architecture for Content-Based Publish-Subscribe Networks. In: Proc. ACM SIGCOMM (August 2004)
7. Carzaniga, A., Rosenblum, D.S., Wolf, A.L.: Design and Evaluation of a Wide-Area Event Notification Service. ACM Trans. Computer Systems 19(3), 332–383 (2001)
8. Stoica, I., Morris, R., Karger, D., Kaashoek, M., Balakrishnan, H.: Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In: Proc. ACM SIGCOMM, pp. 149–160 (2001)
9. Rowstron, A., Druschel, P.: Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In: Liu, H. (ed.) Middleware 2001. LNCS, vol. 2218, pp. 329–350. Springer, Heidelberg (2001)
10. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A Scalable Content-Addressable Network. In: Proc. ACM SIGCOMM, pp. 161–172 (2001)
11. Milo, T., Zur, T., Verbin, E.: Boosting Topic-Based Publish-Subscribe Systems with Dynamic Clustering. In: Proc. of the 2007 ACM SIGMOD, Beijing, China (2007)
12. Wong, B., Guha, S.: Quasar: A Probabilistic Publish-Subscribe System for Social Networks. In: Proc. of the 7th International Workshop on Peer-to-Peer Systems (IPTPS 2008), Tampa Bay, USA (2008)

An Efficient Quad-Tree Based Index Structure for Cloud Data Management

Linlin Ding^{1,2}, Baiyou Qiao^{1,2}, Guoren Wang^{1,2}, and Chen Chen²

¹ Key Laboratory of Medical Image Computing (NEU), Ministry of Education

² College of Information Science & Engineering, Northeastern University, China
linlin.neu@gmail.com, qiaobaiyou@ise.neu.edu.cn, wanggr@mail.neu.edu.cn

Abstract. Recently, as a new computing infrastructure, cloud computing is getting more and more attention. How to improve the data management of cloud computing is becoming a research hot. Current cloud computing systems only support key-value insert and lookup operations. However, they can not effectively support complex queries and the management of multi-dimensional data due to lack of efficient index structures. Therefore, a scalable and reliable index structure is generally needed. In this paper, a novel quad-tree based multi-dimensional index structure is proposed for efficient data management and query processing in cloud computing systems. A local quad-tree index is built on each compute node to manage the data residing on the node. Then, the compute nodes are organized in a Chord-based overlay network. A portion of local indexes is selected from each compute node as a global index and published based on the overlay routing protocol. The global index with low maintenance cost can dramatically enhance the performance of query processing in cloud computing systems. Experiments show that the proposed index structure is scalable, efficient and reliable.

1 Introduction

With the rapid development of computer and Internet technology, cloud computing emerges as a new computing platform and draws more and more attention. Recently, there has been an increasing interest in developing cloud computing systems to provide scalable and reliable services for numerous end users. Examples of such systems contain Google's MapReduce [1], Amazon's Dynamo [2], and so on. These cloud computing systems fully realize the power of cloud computing: scalability, availability, reliability and easy maintenance. They store data with easy key-value model and provide services according to the needs of end users. However, current cloud computing systems only support key-value insert and lookup operations, and they can not effectively support complex queries and management of multi-dimensional data. For example, suppose a conventional web shop built on top of a cloud computing system. In such a system, we need to provide product search via keywords. Specifically, each product is described as $\{c_0, c_1, \dots, c_m, s_0, s_1, \dots, s_n\}$, where c_i represents its company information (e.g. place, brand, scale, etc) and s_i describes the surface characteristics

of the product (e.g. color, texture, style, etc). Typical queries include searching products with specific keywords in a given place or finding the company information of a product about a specific surface characteristic. In order to support the above queries, building more efficient index structure, e.g. multi-dimensional index structure, is a pressing demand.

This paper presents an efficient quad-tree based multi-dimensional index structure, called QT-Chord, which integrates Chord-based [3] routing mechanism and quad-tree based index scheme to support efficient multi-dimensional query processing in a cloud computing system. Although the method of using a master node to maintain the global index and send queries to relative slave nodes is popular and useful, the bottleneck of the master node may happen by increasing the number of compute nodes. Therefore, we use distributed index across the compute nodes to release the workload of the master node. There are two levels of our QT-Chord index structure, global index level and local index level. Numerous compute nodes are organized in a cloud computing system to provide their services to end users. The data of user are divided into data chunks and then stored on different compute nodes. To realize efficient local multi-dimensional data management, each compute node builds its local index by an improved MX-CIF [4] quad-tree index structure, named IMX-CIF quad-tree. The global index is built on top of the local indexes. Each compute node shares a part of storage space for maintaining the global index. The global index is a set of index entries composed of portions of IMX-CIF quad-trees from different compute nodes. To distribute the global index, the compute nodes are organized in an overlay network. We choose Chord as the overlay network only for routing purpose. Other overlay networks, such as CAN [5], can also be used for our research.

The main problem of designing QT-Chord index structure is the distribution of global index to the compute nodes. In IMX-CIF quad-tree, our distributed index structure is responsible for regions of space to the compute nodes in the system. Each multi-dimensional data is indexed by IMX-CIF quad-tree structure. Each quad-tree block is uniquely identified by its code upon reaching the terminal condition. Multi-dimensional data are expressed by their codes. We pass the codes as DHT keys hashed to the compute nodes organized in Chord overlay network. A mapping function is built to map a set of local IMX-CIF quad-tree nodes to compute nodes. To process a query, the compute nodes first locate the relative nodes by their global index. Then, the relative nodes process the query in parallel and return the results to the clients.

The contributions of this paper are the followings:

- An improved MX-CIF quad-tree structure (IMX-CIF) is proposed to effectively index the multi-dimensional data on each compute node.
- A multi-dimensional index structure, QT-Chord, is proposed to effectively process complex queries about multi-dimensional data. Query processing algorithms, such as point, range and KNN query processing, are presented.
- A series of experiments on several machine nodes with numerous data objects demonstrate that QT-Chord is quite efficient and scalable.

The rest of this paper is organized as follows: Section 2 introduces related work. Section 3 proposes QT-Chord index structure. Section 4 presents the query processing algorithms, including point, range and KNN query processing. Section 5 shows the experiment evaluation and the conclusions are given in Section 6.

2 Related Work

2.1 Data Management of Cloud Computing

The researches of cloud computing systems are developing at an astonishing speed, such as Google's GFS [6], Bigtable [7] and Amazon's Dynamo which store data by a key-value model. They can effectively process data insert and keyword queries without supporting complex queries, such as range queries, KNN queries. A general index framework of cloud computing systems is proposed in [8]. The machine nodes are organized in a structured overlay network. Each machine node builds its local index to accelerate data access. A global index is constructed by selecting and publishing a portion of local indexes in the overlay network. The framework of our research is based on [8]. An efficient approach to build multi-dimensional index structure for cloud computing systems (EMINC) is proposed in [9]. This index structure uses the combination of R-tree [10] and KD-tree [11] to organize data, reaching quick query processing and efficient index maintenance. However, in the relative nodes locating phase, EMINC chooses all the slave nodes in the cluster as the candidates of the query since it doesn't have the information about the data distribution on each slave node, and thus chooses all the possible slave nodes as the candidate set. RT-CAN is proposed in [12], which introduces an efficient multi-dimensional index structure to support complex queries in a cloud system based on the routing protocol of CAN and R-tree indexing scheme. But, contrary to R-tree, the superiority of quad-tree is that the decomposition is implicitly known by all the compute nodes in the system without any communications. In QT-Chord, we propose a new quad-tree structure to manage the multi-dimensional data which can enhance the query performance with low maintenance cost.

2.2 MX-CIF Quad-Tree

It is known that MX-CIF quad-tree structure is capable of facilitating spatial data objects and queries. It starts with a big rectangle which contains all the data objects. The rectangle is the root of MX-CIF quad-tree and is divided into four congruent subrectangles by dividing each of its sides into two. The four congruent rectangles are the children of the original rectangle. In turn, each of these rectangles is divided into four until it reaches the terminal condition of dividing course. For each data object o , the dividing course stops encountering a block b such that o overlaps at least two child blocks of b or upon reaching a maximum level (d_{max}) of dividing course. We use an improved quad-tree structure of MX-CIF (IMX-CIF) to index the multi-dimensional data of the compute nodes.

3 QT-Chord Index

3.1 System Overview

Figure 1 shows the index structure and framework of QT-Chord in a cloud computing system. The index structure of QT-Chord is shown in Figure 1(a), which has two levels: global index level and local index level. Each compute node builds a local index to manage its data and processes query requests. The global index is built on top of the local indexes. Each compute node shares a portion of its storage space and publishes a portion of its local index as the global index to enhance the system performance. To distribute the global index, the compute nodes are organized in an overlay network. The framework of QT-Chord is presented in Figure 1(b). The master node is only responsible for receiving and sending query requests. The compute nodes are organized in a Chord overlay structure designed for peer-to-peer systems. The overlay is set as a logical network for partitioning data and routing queries. We consider the compute nodes remaining online unless the hardware fails.

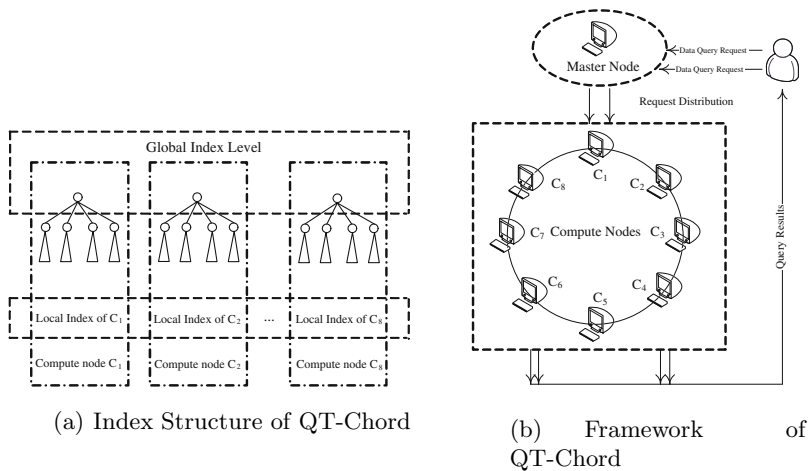


Fig. 1. Index Structure and Framework of QT-Chord

In QT-Chord index structure, each compute node builds a local index using our IMX-CIF quad-tree structure, which will be introduced in Section 3.2 to effectively manage its residing multi-dimensional data and process complex queries. In IMX-CIF quad-tree, the codes of quad-tree blocks are used to express the multi-dimensional data and set as the DHT keys hashed to the compute nodes organized in the overlay network. We choose Chord as the overlay network. Chord has high performance of scalability and availability to satisfy the demands of cloud computing systems. Although CAN is capable of processing multi-dimensional data, the average hops are higher than Chord when there are thousands of compute nodes. The query processing has been divided into two steps. In the first step, the query receiver finds all the relative nodes to the

query by searching the global index and routes the query to them. In the second step, those compute nodes relative to the query process the query in parallel and return the results to the client at last.

3.2 IMX-CIF Quad-Tree Structure

In order to effectively express the multi-dimensional data, IMX-CIF quad-tree structure is proposed. We recursively partition the data space until reaching the terminal condition which is the same as MX-CIF quad-tree. However, a single point of failure occurs if all the quad-tree operations begin at the root. In addition, in distributed index structure, it is unnecessary to start the operations at the root of quad-tree. Therefore, a concept of the minimum dividing level (d_{min}) is proposed, where all data are stored and all query processing occur at levels $d_{max} \geq l \geq d_{min}$. That is to say, there is no data storing at levels $0 \leq l < d_{min}$. Figure 2 shows the structure of a local IMX-CIF quad-tree of a compute node. As shown in Figure 2(a), the two-dimensional space is recursively divided to index seven multi-dimensional data objects A to G . This dividing course is known globally, which is essential to support distributed query processing. The IMX-CIF quad-tree is shown in Figure 2(b). One multi-dimensional data is distributed to several quad-tree blocks. Each internal node has four children. Each tree node is labeled with a code. The root code is “0”. Each tree edge is also labeled with the principle as follows. In each dividing course, the top right block and its edge have the code “0”, the top left, lower left and lower right blocks with the code “1”, “2” and “3” respectively. Then, the code of each internal node or leaf node can be obtained by concatenating all codes on the path from the root to the node itself. For example, data object A overlaps two child blocks of its parent, so the codes of A are “00” and “03”. Data object F is divided into the third level with the code from root node to itself is “0”, “2”, “1” and “0”, so the code of data F is “0210”.

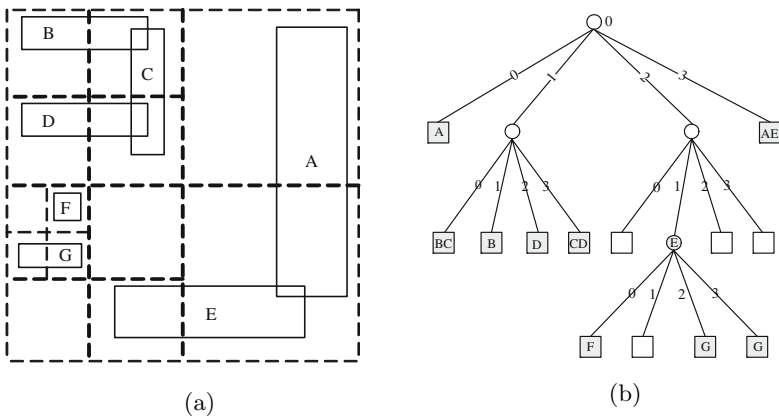


Fig. 2. The Naming Function of QT-Chord

3.3 The Mapping and Publishing Scheme of QT-Chord

In our cloud computing system, we choose Chord as the underlying structure. In IMX-CIF quad-tree structure, each quad-tree block can be uniquely identified by its codes. The size of local index is proportional to its data size. Therefore, to reduce the index maintenance cost, we can not publish all the local index nodes to the global index. A mapping scheme is necessary designed for mapping a set of local index nodes to the compute nodes.

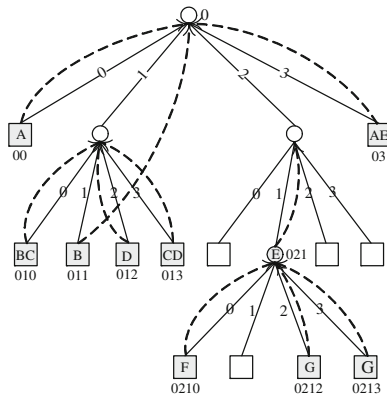


Fig. 3. Mapping Function of QT-Chord

We can randomly choose a set of local index nodes to publish as the global index, which however destroys data locality and impedes effective range query processing. Instead, in QT-Chord, a new method is proposed to generate DHT keys by the codes that preserves locality. Each compute node builds a local IMX-CIF quad-tree index to manage the multi-dimensional data. A multi-dimensional data is partitioned by its IMX-CIF quad-tree structure obtaining its codes. Simultaneously, with node labeling strategy, each leaf node obtains its code, where the code of its ancestor is a prefix of its code. The meta-data of a published IMX-CIF quad-tree node is that: $(code, ip)$, where $code$ is the code of the quad-tree node and ip is the IP address of the corresponding compute node. The remaining problem is how to map the quad-tree nodes to the compute nodes by a naming function to generate DHT keys, so that the ancestor nodes can be easily located in query processing and the maintenance cost is reduced. For example, each quad-tree node in Figure 3 has a code, where the IMX-CIF quad-tree is the same as Figure 2. Figure 3 shows the naming function of a local IMX-CIF quad-tree in QT-Chord. The naming function is recursively described as follows:

$$f_2(b_1b_2...b_{i-1}b_i) = \begin{cases} f_2(b_1b_2...b_{i-1}) & \text{if } b_i = b_{i-1} \\ b_1b_2...b_{i-1} & \text{otherwise} \end{cases}$$

We take two-dimensional space as an example. Specifically, given a compute node code or a query code, the function checks the last bit and the second

last bit. If they are the same, the last bit is abandoned and this procedure is repeated. Otherwise, the procedure is stopped after abandoning the last bit. The dotted arrows in Figure 3 illustrate the corresponding relationships of the quad-tree nodes by the naming function. For example, the node with code “011” is mapped to the root node. The node with code “0212” is mapped to the node with “021”. Similar naming method is used in [13], but it is different from us that it names the space KD-trees for all the data and imports the virtual root concept. We only use the naming function to generate DHT keys of storing data of the quad-tree nodes. The DHT keys are published as the global index of the compute nodes organized in Chord overlay network. In this way, a compute node not only preserves its local index but also the global index consisting of the meta-data of the nodes located in its region.

4 Query Processing

4.1 Point Query Processing

A point query is denoted as $Q(point)$, where $point = (v_1, \dots, v_d)$, indicating a d -dimensional point. In this paper, we set d as 2 to illustrate this situation. When a user initiates a point query, the compute node which receives the query first calculates the code of quad-tree block with the point for the query at the dividing level d_{min} . Then it generates the DHT key of the query by the naming function. The compute node looks up its part of global index and finds the compute nodes which have the same prefix of the query DHT key. The query messages are forward via Chord’s routing mechanism in parallel. Algorithm 1 shows the idea of point query processing. We first calculate the code c of the query and its DHT key (line 1-2) and then the DHT key nodes route the query by global index (line 3). From line 4 to 9, we filter the results set and return the final results.

Algorithm 1. Point Query Processing

Input: $Q(key)$
Output: $R(S_i, ID)$

- 1: calculate c ;
- 2: $DHTkey = f_2(c)$;
- 3: add the DHT key nodes to S_i by global index;
- 4: **for** $\forall s_i \in S_i$ **do**
- 5: **if** s_i does not have the DHT key **then**
- 6: $S_i = S_i - s_i$;
- 7: **end if**
- 8: **end for**
- 9: **return** S_i ;

4.2 Range Query Processing

A range query is described as $Q(range)$, where $range = [l_1, u_1], \dots, [l_d, u_d]$ is a multi-dimensional hypercube. We take two-dimensional space as an example.

The main idea of range query is the same as point query. Range query takes queries or data objects as range space. When a user initiates a range query, the compute node which receives the query first identifies the code list of the query and then finds the DHT keys of the query by the naming function. After that, the compute node routes the query messages to the relative nodes by the global index. The range query algorithm is described in Algorithm 2. From line 1 to 7, we find the DHT keys of the query and route them by the global index in line 8. From line 9 to 13, the results set is filtered and returned.

Algorithm 2. Range query Processing

```

Input: Query( $Q$ )
Output:  $R(S_i, ID)$ 
1: code list  $G = \emptyset$ ;
2: DHT key list  $K = \emptyset$ ;
3: add  $c$  to  $G$ ;
4: for  $\forall c_i \in G$  do
5:   DHT key  $k_i = f_2(c_i)$ ;
6:   add  $k_i$  to  $K$ ;
7: end for
8: add  $K$  to  $S_i$  by global index;
9: for  $\forall s_i \in S_i$  do
10:  if  $s_i$  does not have the DHT key then
11:    $S_i = S_i - s_i$ ;
12:  end if
13: end for

```

4.3 KNN Query Processing

A KNN query is denoted as $Q(key, k)$, which returns the k nearest data to the key . To effectively process the KNN query, the idea is that the query starts at a small range and enlarges the search space accordingly. In each search space, we find the k nearest data to key and replace the temporary data with a nearer data at all search space. The whole course stops when the IMX-CIF quad-tree reaches the depth of d_{min} . The idea of KNN query processing is shown in Algorithm 3. In each dividing course, a query region is generated with key as the center and distance between key and the dividing center as radius to obtain the nearest k results of the query.

5 Experiment Evaluation

We evaluate QT-Chord index structure in a small cloud computing system. Nine machines are connected together to construct the cloud computing system. Each machine has Intel Duo Core 4.00Hz CPU, 1TB memory, 250G Disk. One machine is the master node and the others are organized in a Chord overlay network. The eight compute nodes can provide 128 virtual nodes by the technology of

Algorithm 3. KNN Query Processing

Input: $Q(key, k)$
Output: $R(S_i, ID)$
 d_i : the distance between key and the i th dividing center;
 l_i : the dividing depth of key;
 c_i : the circle region with key as the center and d_i as the radius;

- 1: $S_i = \emptyset$;
- 2: DHT key list $K = \emptyset$;
- 3: **for** $d_i = l_i, d_i \geq d_{min, i} - -$ **do**
- 4: get the k nearest distance DHT keys in c_i ;
- 5: $S_i =$ the k nodes of these DHT keys;
- 6: **end for**
- 7: **return** S_i ;

virtual machine. Each of the eight machines simulates 16 virtual nodes. Each time 32 more virtual nodes are considered to be added into the cloud computing system. The master node controls the behaviors of all the compute nodes, such as partitioning the data of users and distributing queries continuously. A compute node retrieves a new query from the master node after finishing its current query.

Two types of data sets are used to evaluate the performance of QT-Chord: the uniform distribution data set and the skewed distribution data set. In the uniform distribution data set, each node generates 500K data objects. The data object has 2 to 5 attributes. These data objects follow the uniform distribution. So the nodes hold nearly the same number of data objects. In the skewed distribution data set, we use the *zipf-like* distribution to generate the skewed distribution data set with the distribution parameter α as 0.8. Each compute node generates 100K data objects. QT-Chord is compared with the multi-dimensional index for cloud data management EMINC and RT-CAN. EMINC index structure builds an R-tree on the master nodes and a KD-tree on each slave node. RT-CAN index scheme combines CAN based routing protocol and R-tree based index structure to process multi-dimensional query requests in a cloud computing system.

5.1 Performance of Point Queries

Figure 4 shows the performance of the number of processed queries per-second of point queries. When increasing the number of compute nodes, the number of processed queries per-second of point queries increases almost linearly. The performance of QT-Chord is better than RT-CAN and EMINC. QT-Chord chooses the relative nodes to the query by the global index, but EMINC chooses all the slave nodes in the system as the query candidate set. In addition, the master nodes in EMINC build an R-tree index to increase the maintenance cost and reduce query performance. In RT-CAN, each compute node uses an R-tree like index structure to manage the local data by storing the range information of R-tree, but our quad-tree decomposition is implicitly known by all the compute nodes in the system.

5.2 Performance of Range Queries

Figure 5 shows the performance of the number of processed queries per-second of range queries by two kinds of data sets. By increasing the number of compute nodes, the number of processed queries per-second of range queries also increases. In the relative nodes locating phase of EMINC, although it builds a cube for each compute node, it chooses all the slave nodes in the system as candidates of the queries. Therefore, the performance of EMINC is lower than RT-CAN and QT-Chord. In RT-CAN, the compute nodes are organized in a variant of CAN overlay network. When processing a range query, the query is recursively forwarded to all neighbors overlapped with the query, so the maintenance and communication cost increases by increasing the number of nodes. However, QT-Chord index structure effectively transforms the multi-dimensional range queries into several code information then hashed to nodes of Chord. So, as shown in Figure 5, the performance of QT-Chord is more efficient than RT-CAN and EMINC. The performance of changing the *zipf-like* parameter is shown in Figure 6. The performance decreases by enlarging the *zipf-like* parameter.

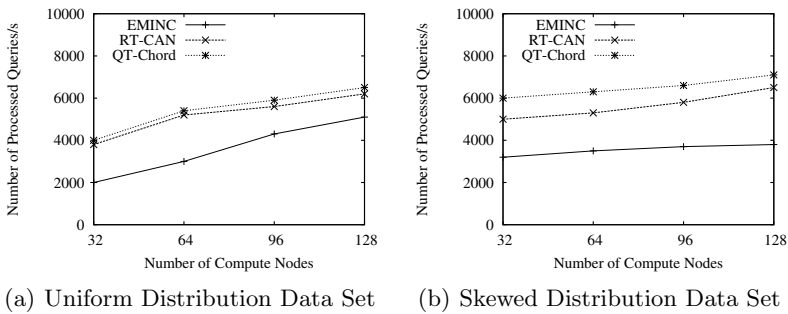


Fig. 4. The Performance of Point Queries

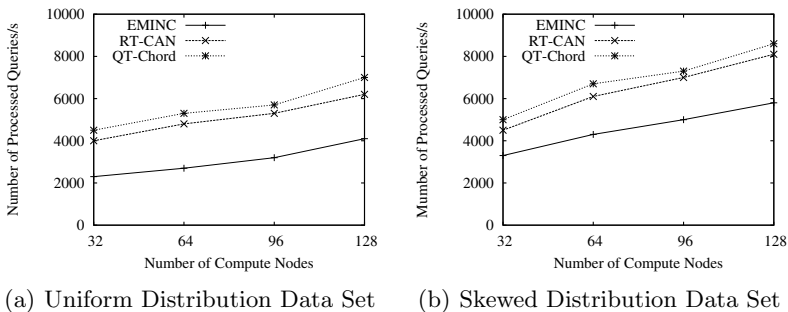


Fig. 5. The Performance of Range Queries

5.3 Performance of KNN Queries

Figure 7 shows the KNN queries performance with k from 1 to 16 respectively with uniform distribution data set and skewed distribution data set. Each compute node has the number of data 100K. As shown in Figure 7, the KNN queries performance decreases by the reason that the query space increases with k and more index entries about the query. The performance of QT-Chord is better than RT-CAN. EMINC does not give the KNN query algorithm.

5.4 Effect of Dimensionality

This experiment shows the performance of changing the dimensions in the uniform distribution data set. The skewed distribution data set is set to two-dimensions. Figure 8 shows the performance of QT-Chord index structure in different dimensions. The number of processed queries per-second decreases by increasing the number of dimensions and the best one is two-dimensions. The reason is that the range query overlapped with more index items by increasing the dimensions.

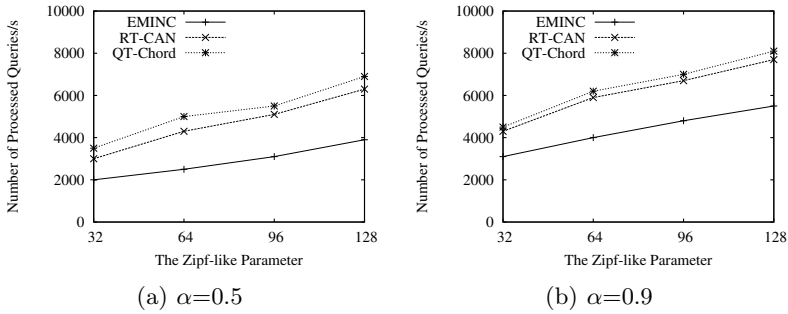


Fig. 6. The Performance of Range Queries by Changing of The Zipf-like Parameter

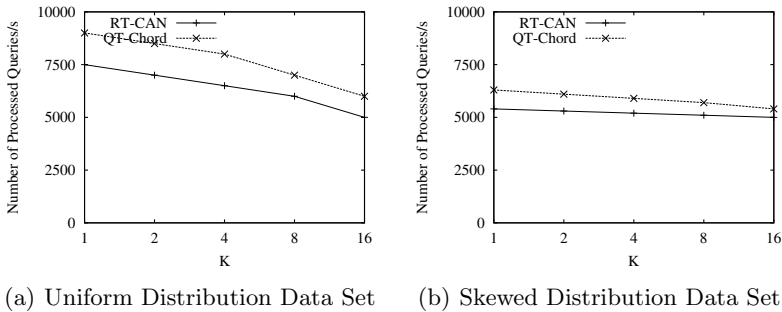


Fig. 7. The Performance of KNN Queries

5.5 Effect of Changing d_{min}

We change the value of d_{min} to test the performance of QT-Chord. As shown in Figure 9, the average number of messages increases by increasing d_{min} . This phenomenon is due to increasing the initial d_{min} level messages per query and hence due to losing the pruning capability of the distributed quad-tree index structure when it becomes more and more like a regular grid structure.

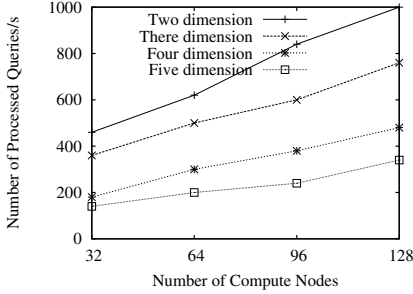


Fig. 8. Effect of Dimensionality

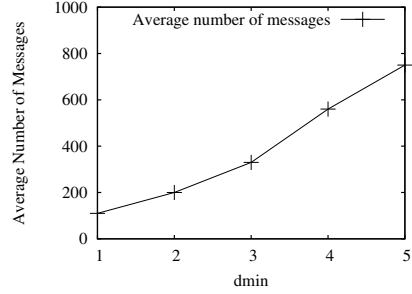


Fig. 9. Average Number of Messages Per Query by Changing of d_{min}

6 Conclusions

In this paper, we present QT-Chord, a novel multi-dimensional index structure for cloud computing systems which has two levels: global index level and local index level. Each compute node builds a local index to manage its data and process query requests. Each compute node publishes a portion of its local index as the global index to reduce the workload of master nodes and enhance the system performance. The compute nodes are organized in a Chord overlay network. The algorithms of point, range and KNN query processing, are proposed in this paper. Experiments show that QT-Chord is effective, scalable and available.

Acknowledgement. This work was partially supported by the National Natural Science Foundation of China (Grant No. 61073063, 71001018, 60933001), the Fundamental Research Funds For The Central Universities (Grant No. N090404012), and Ocean Public Benefit Industry Special Research (Grant No. 201105033).

References

1. Dean, J., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. In: Proc. of OSDI, pp. 137–150 (2004)
2. 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: Proc. of SOSP, pp. 205–220 (2007)
3. Stoica, I., Morris, R., Karger, D.R., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: Proc. of SIGCOMM, pp. 149–160 (2001)

4. Kedem, G.: The Quad-CIF Tree: A Data Structure for Hierarchical On-Line Algorithms. In: Proc. of the 19th Design Automation Conference, pp. 352–357 (1982)
5. Ratnasamy, S., Francis, P., Handley, M., Karp, R.M., Shenker, S.: A scalable content-addressable network. In: Proc. of SIGCOMM, pp. 161–172 (2001)
6. Ghemawat, S., Gobioff, H., Leung, S.-T.: The Google file System. In: Proc. of SOSP, pp. 29–43 (2003)
7. Chang, F., Dean, J., Ghemawat, S., Hsieh, W.C., Wallach, D.A., Burrows, M., Chandra, T., Fikes, A., Gruber, R.E.: Bigtable: A Distributed Storage System for Structured Data. In: Proc. of OSDI, pp. 205–218 (2006)
8. Wu, S., Wu, K.-L.: An Indexing Framework for Efficient Retrieval on the Cloud. *IEEE Data Eng. Bull. (DEBU)* 32(1), 75–82 (2009)
9. Zhang, X., Ai, J., Wang, Z., Lu, J., Meng, X.: An Efficient Multi-Dimensional Index for Cloud Data Management. In: Proc. of CloudDb, pp. 17–24 (2009)
10. Guttman, A.: R-Trees: A Dynamic Index Structure for Spatial Searching In: Proc. of the ACM SIGMOD, pp. 47–57 (1984)
11. Bentley, J.L.: Multidimensional Binary Search Trees Used for Associative Searching. *Commun. ACM* 18(9), 509–517 (1975)
12. Wang, J., Wu, S., Gao, H., Li, J., Ooi, B.C.: Indexing Multi-dimensional Data in a Cloud System. In: Proc. of SIGMOD, pp. 591–602 (2010)
13. Tang, Y., Zhou, S., Xu, J.: LigHT: A Query-Efficient yet Low-Maintenance Indexing Scheme over DHTs. *IEEE Trans. Knowl. Data Eng. (TKDE)* 22(1), 59–75 (2010)

Efficient Duplicate Detection on Cloud Using a New Signature Scheme

Chuitian Rong^{1,2}, Wei Lu^{1,2}, Xiaoyong Du^{1,2}, and Xiao Zhang^{1,2,3}

¹ Key Labs of Data Engineering and Knowledge Engineering, MOE, China

² School of Information, Renmin University of China

³ Shanghai Key Laboratory of Intelligent Information Processing
{rct682, uqwlw, duyong, xiaozhang}@ruc.edu.cn

Abstract. Duplicate detection has been well recognized as a crucial task to improve the quality of data. Related work on this problem mainly aims to propose efficient approaches over a single machine. However, with increasing volume of the data, the performance to identify duplicates is still far from satisfactory. Hence, we try to handle the problem of duplicate detection over MapReduce, a share-nothing paradigm. We argue the performance of utilizing MapReduce to detect duplicates mainly depends on the number of candidate record pairs. In this paper, we proposed a new signature scheme with new pruning strategy over MapReduce to minimize the number of candidate record pairs. Our experimental results over both real and synthetic datasets demonstrate that our proposed signature based method is efficient and scalable.

Keywords: duplicate detection, MapReduce, Cloud.

1 Introduction

Duplicate detection is widely used in a variety of applications, including deduplication [1], data cleaning and record linkage [2][3]. Given a set of records [4], the objective of duplication detection is to identify records that refer to the same real-world entity. Due to a variety of quality problems, such as typos, abbreviations, update anomalies and combination thereof, two records that do not exactly match with each other may refer to the same entity. Two records whose similarity is not less than a pre-defined threshold are considered as duplicates. The straightforward approach to identify duplicates over a dataset requires the comparison on each pair of strings. Obviously, the computational cost will be extremely high when the dataset is large scale. As such, instead of offering each pair of strings as a candidate record pair, previous work mainly focuses on proposing efficient approaches, including indexing techniques such as inverted index [4], Trie Tree [5] and B⁺-Tree [6] to reduce the number of candidates pairs. However, with increasing volume of data, the performance of these approaches is still far from satisfactory especially when the dataset cannot be properly reside in the main memory.

¹ In this paper, for each record, we concatenate its attribute values into a single string.

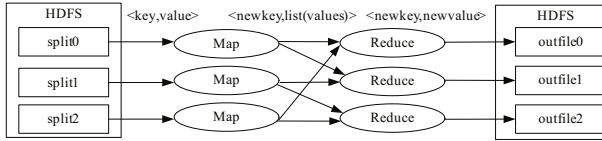


Fig. 1. Overview of the MapReduce Execution

In order to alleviate the problem that incurred by the increasing scale of datasets, we try to apply the MapReduce paradigm to detect the duplicates. MapReduce is a programming paradigm for processing large-scale datasets over a share-nothing parallel cluster. Fig. 1 shows its execution overview. It consists of two procedures: Map and Reduce. In this framework, key/value pairs are sorted and shuffled based on their keys; the data items with the same key will be grouped together. In case of duplicate detection by using MapReduce, the key work is to assign the duplicate candidates the same key, here we say signature. Then, the duplicate candidates can be shuffled into one group and sent to one Reduce node for last verification. The signature can be defined as the common characteristics of duplicates candidates. It must have discriminative power, that is to say duplicate candidates must have same signature and records with different signatures can not be duplicates. We argue that the performance of detecting duplicates in MapReduce framework mainly depends on the number of candidate set size in each group. To achieve better performance, new proper signature scheme and pruning strategy are proposed to decrease the pseudo duplicates in each candidate set.

As the MapReduce program paradigm is a share-nothing framework. There are two challenges need to be solved. One, we must apply proper signature scheme for duplicate candidates, which has direct effect on duplicate candidate size and the efficiency of verification. As the data is split and the task on different nodes is done independently, there is no more global information can be used. The other challenge is that we must devise new prune method to decrease the pseudo duplicates as many as possible in new environment. Hence, in this paper, we make the following contributions:

- We proposed solution for duplicate detection on cloud by using new signature scheme. The signature scheme can produce exact signatures for each candidate record pairs.
- We proposed a new pruning strategy to minimize the pseudo candidates number in each candidate set.
- Redundant comparisons for the same candidate record pairs are eliminated in our solution.

The rest of the paper is organized as follows: Section 2 presents the problem definitions and preliminaries. Section 3 describes our proposed signature scheme. Next, duplicate detection using this signature scheme is illustrated in Section 4.

Experiments study is given in Section 5. Related work is covered in Section 6 and Section 7 concludes this paper.

2 Problem Definition and Preliminaries

2.1 Problem Definition

Definition 1. *n*-gram token set Let r be a string. A set of n -gram tokens of r denoted as $G_r = \{g_1^r, g_2^r, g_3^r, \dots\}$, where g_i^r is defined as follows. $g_i^r = r[i - n + 1, i - n + 2, \dots, i]$, where $r[s, \dots, t]$ means the substring of r from s^{th} to t^{th} . If s is less than 0 or t is greater than the length of r , then the blank is replaced by special character. So, $|G_r| = |r| + n - 1$.

Example 1. For example, record “Jim Gray” can be tokenized as a set of 3-grams: { ###, #Ji, Jim, im , m G, Gr, Gra, ray, ra\$, a\$\$ }.

As there does not exist a similarity function can always perform the best than other similarity functions in all application scenarios. In this paper, we explored a widely used similarity function Jaccard. Unless otherwise specified, $sim(r_i, r_j)$ refers to $sim_{Jaccard}(r_i, r_j)$.

Definition 2. Similarity Given a record pair (r_1, r_2) , let $G_{r_1} = \{g_1^{r_1}, g_2^{r_1}, \dots\}$, $G_{r_2} = \{g_1^{r_2}, g_2^{r_2}, \dots\}$ be the n -gram token set of r_1 and r_2 , respectively. The similarity of record pair (r_1, r_2) is defined as $sim_{Jaccard}(r_1, r_2)$.

$$sim_{Jaccard}(r_i, r_j) = \frac{|G_{r_i} \cap G_{r_j}|}{|G_{r_i} \cup G_{r_j}|} \tag{1}$$

Using the defined similarity function to evaluate the similarity between any two records, we can formally define the result of duplicate detection: $\{\langle r_i, r_j \rangle | \forall r_i, r_j \in R, sim(r_i, r_j) \geq \theta\}$, where θ is pre-assigned by users.

For the sake of brevity, we give some necessary symbols and their definitions in Table 1 that will be used throughout this paper.

2.2 Properties of Jaccard Similarity Function

Definition 3. Global Ordering U is the universe of n -gram in data set R .

$$\forall g_i, g_j \in U, \text{ if } tf_{g_i} \leq tf_{g_j} \text{ then } g_i \preceq g_j.$$

Definition 4. Prefix Tokens For record r , $G_r = \{g_1^r, g_2^r, \dots, g_{|G_r|}^r\}$. The tokens g_i^r in G_r are sorted by global ordering. Its prefix token is denoted as G_r^p .

$$G_r^p = \{g_i | g_i \in G_r, 1 \leq i \leq |G_r| - \lceil |G_r| * \theta \rceil + 1\}.$$

Definition 5. θ Split Point The θ split point is the token that ranked at $\lceil |r_i| * \theta \rceil - 1$ to the rear of record’s sorted token list. The tokens that ranked before this token are the candidates for signatures.

Table 1. Symbols and their definitions

Symbol	Definition
R	collection of records
r	record in R
U	finite universe of n -grams
G_r	sequence of n -grams for record r
$ G_r $	the number of n -grams in G_r
G_r^p	the first p n -grams for record r
$ G_r^p $	the number of n -grams in G_r^p
g	an n -gram of U
tf_g	term frequency of gram g
θ	pre-assigned threshold
$sim(r_i, r_j)$	the similarity between record r_i, r_j
\preceq	partial order

tokens	oij	094	loi	94#	iji	idy	jic	dyn	434	tme	c_3	stm	343
term frequency	2	3	4	5	7	8	9	13	13	16	20	20	21

Fig. 2. Prefix Tokens

Example 2. Take the record $r =$ “Advance duplicate address detect” for example and set $\theta = 0.8$. So, $|G_r| = 34$ and $|G_r^p| = 34 - \lceil 34 * 0.8 \rceil + 1 = 7$. This is shown in Fig. 2

When gram tokens in each record are sorted by global ordering, the Jaccard similarity function has the property [7].

Property 1. $\forall r_i, r_j \in R$, let $G_{r_i}^p$ and $G_{r_j}^p$ be their prefix token set, respectively. If $sim(r_i, r_j) \geq \theta$, then $|G_{r_i}^p \cap G_{r_j}^p| \geq 1$.

As the above property of Jaccard similarity, if two records are duplicates they must have one common prefix token at least. If each of prefix tokens of the record is used as its signature, the duplicate records can have the chance to be shuffled into one group in MapReduce framework. Then, we can verify them.

3 Signature Scheme

The signature scheme can be considered as blocking or grouping process, in which the duplicate candidates will be assigned the same signature. In MapReduce, the signature is used as the key for duplicate candidate records, then they can be shuffled into one group and sent to one Reduce node for the last verification. As MapReduce is a share-nothing programming paradigm, there is no more global information about records can be used as in traditional solution. The property of Jaccard similarity function can be used to generate signature. The direct approach is to use each prefix token as its signature in [8] and take each word as a token.

Algorithm 1. *SignatureGeneration*

```

1 Map(key,value):
2 newvalue  $\leftarrow$  key;
3 foreach itemi  $\in$  value do
4   |   select itemj from value using LengthPruning;
5   |   newkey  $\leftarrow$  (itemi, itemj)(i > j);
6   |   write(newkey,newvalue);
7 Reduce(key,values):
8 newvalue  $\leftarrow$  values;
9 TokenNumber  $\leftarrow$  the token number in newvalue;
10 result  $\leftarrow$  ReducePrune(key,TokenNumber);
11 if result==true then
12 |   write(key,newvalue);

```

When using the n-gram based method for duplicates detection, it is not effective. There are three causes:(1)One single prefix token may be also the prefix token of many other records;(2)Many record pairs have more than one common prefix token;(3)The number of gram tokens for the same record will be many times of its word number. All these can be seen from the Example 3.

Example 3. Take these two records r_1 and r_2 below for example.

- r_1 : Advanced duplicate address detect
- r_2 : Advance duplicate address detection

Both r_1 and r_2 contain 4 words, but will produce 35 and 37 3-gram tokens respectively. Given threshold 0.8, if taking each word as a token they will produce only one prefix token, {*Advanced*} and {*Advance*} respectively [8]. If they are used as signature, these two records can't be shuffled into one group as candidates. So, taking each word as a token is sensitive to many typo errors. When split the record into 3-gram, their prefix tokens number are all 8. If each of gram in prefix tokens is used as the record's signature, there will be 8 signatures for each record. The common prefix tokens of them are {*dup, add, ddr, du, dre, upl, dva, Adv*}. If using the signature as in [8], these two records will be verified eight times. If just use single prefix token as signature, many redundancies and redundant verifications will be imported. As each prefix token of one record will be prefix token of many other records. Usually, many records will have more than one common prefix tokens.

By thorough analysis we proposed a new signature scheme using the records common prefix tokens. It is an exact signature scheme. During the signature generation we applied two prune strategies using record's properties.

3.1 Signature Generation

In order to produce the new signature we have to complete the works:(1)get a global ordering of all tokens in the data set;(2)build the inverted list just for

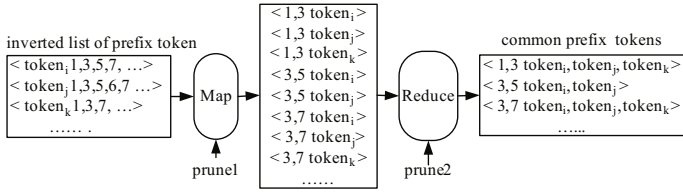


Fig. 3. Data Flow for Signature Generation

each distinct prefix token using MapReduce. Each item in inverted list contains the information about the record, such as record id, record length and its θ split value(term frequency). The item in inverted list is sorted by the value of record length in it; (3)get the maximum common tokens for each duplicate pair by analyzing the inverted list using pruning strategies. As build the inverted index is common place,we just give the details of signature generation.

This process is only based on the information of records, not the original record. The details are given in Algorithm 1 and the data flow in this MapReduce procedure is shown in Fig 3, record length and θ split value information is omitted for illustrating clearly. In this procedure, the inverted index files are also split and processed by different Map. Each input key/value pair in the Map is an sorted inverted list for one prefix token(line 1). To get the longest common prefix tokens of each record pair, we set the input *key* to the *newvalue*(line2). Scan the inverted list from start using the length pruning strategy(line 3-6) and write out the two items that satisfy the pruning condition as *newkey*. In Reduce, the prefix tokens with the same key are common prefix tokens of two records(line 8). Then it apply the second pruning method(line 10). If it satisfies our requirement the tokens in *newvalue* is their signature.

3.2 Pruning Method

In order to improve the efficiency of signature generation and decrease the number of pseudo duplicates, we proposed two pruning methods used in Map and Reduce respectively. The first pruning method is used for each input line in Map. That is an length filtering method as shown in Algorithm 1(line 8). The second pruning method is used in Reduce, as shown in Algorithm 1(line 19),*ReducePrune*. As the length filtering is widely used,we don't illustrate here[9]. Below, we give our pruning strategy used in Reduce. It is just based on record length,common prefix token number and θ split values. As all tokens in the data set are sorted by global ordering, each token can be seen as a dot in the number axis. We use number axis and set properties to prove its correctness. Assume their common token number is K , spt_i and spt_j are their θ split values. In Fig 4, we can see there are two situations for these two records, $spt_i \leq spt_j$ and $spt_i > spt_j$.

Lemma 1. For $spt_i \leq spt_j$,the prune condition is:

$$\frac{K + \lceil |G_{r_i}| * \theta \rceil - 1}{|G_{r_i}| + |G_{r_j}| - (K + \lceil |G_{r_i}| * \theta \rceil - 1)} \geq \theta$$

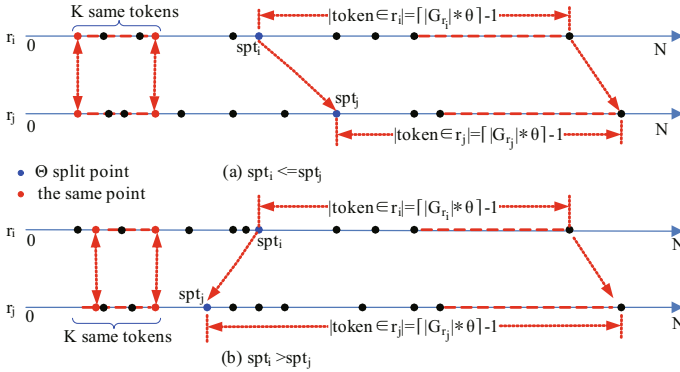


Fig. 4. Pruning Condition

Proof. The common tokens of these two records are come from two parts in the number axis, one part is before their split point spt_i and spt_j , the other part is after these two split points as shown in Fig 4.

The signature is their common tokens that come from the first part, the number is K . The maximum number of common tokens that come from the other part is determined by the length of two records and their θ split value. As $spt_i \leq spt_j$, the maximum number of common tokens from the last part is $\lceil |G_{r_i}| * \theta \rceil - 1$. So, the maximum common tokens for these two records is $K + \lceil |G_{r_i}| * \theta \rceil - 1$. If they are duplicates, the similarity of them can not less than the given θ . So the pruning condition is:

$$\frac{K + \lceil |G_{r_i}| * \theta \rceil - 1}{|G_{r_i}| + |G_{r_j}| - (K + \lceil |G_{r_i}| * \theta \rceil - 1)} \geq \theta$$

Lemma 2. For $spt_i > spt_j$, the prune condition is:

$$\frac{K + \lceil |G_{r_j}| * \theta \rceil - 1}{|G_{r_i}| + |G_{r_j}| - (K + \lceil |G_{r_j}| * \theta \rceil - 1)} \geq \theta$$

As the proof is the same to the last, we don't give here for length limit of paper.

4 Duplicate Detection Processing

After the signature generation we have got the exact signatures for each duplicate candidate pair. Using the exact signature, the duplicate detection can be completed efficiently. The whole process is given in Algorithm 2. Before doing duplicate detection we distributed the signature index file as distributed cache file to each computing node [10]. In setup procedure, each computing node will use this file to build signature structures in its local main memory. In order to save space, each node just read in the related block of cache file about the processed data split. In the Map, for each input record its signatures can be get from signature structures that built in setup (line 3-6). The records are assigned their

Algorithm 2. *DuplicateDetection*

```

1 Setup:
2 read cache file and build signature information;
3 Map(key,value):
4 Signatures  $\leftarrow$  GetSignatures(value);
5 foreach sig  $\in$  signatures do
6   write(sig,value);
7 Reduce(key,values):
8 foreach vali, valj  $\in$  values do
9   sim  $\leftarrow$  simjaccard(vali, valj);
10  if sim  $\geq$   $\theta$  then
11    newvalue  $\leftarrow$  (vali, valj);
12    write(key,newvalue);

```

signatures, then will be shuffled by their signatures and send to Reduce node for last verification. In Reduce, each pair of records with the same signature will be verified using similarity function $sim_{jaccard}(r_i, r_j)$ (line 9-11).

Intuitively, when the token number in signature is more the records with this signature is less. Using this signature scheme can remove redundant verifications and can decrease candidate set size. We can prove the candidate set size produced by using common prefix token is far less than using single prefix token. Let the candidates number of using single prefix token and using common prefix token is N_1, N_2 respectively. We can prove $N_2 \ll N_1$.

Proof. For direct approach, $\forall r_i \in R$, its prefix tokens is $G_{r_i}^p$. As, $\forall g^{r_i} \in G_{r_i}^p$ will be considered as signature and assigned to the record. So, the total intermediate records is $N_1 = \sum_{i=1}^n |G_{r_i}^p|$. As some prefix tokens just occur in one record, so

$$N_1 > 2 \sum_{i=1}^n \sum_{j>i}^n |G_{r_i}^p \cap G_{r_j}^p|.$$

For our proposed new signature scheme, $\forall r_i, r_j \in R$, it takes $G_{r_i}^p \cap G_{r_j}^p$ as their signature. If $G_{r_i}^p \cap G_{r_j}^p \neq \Phi (i \neq j)$, the result will be assigned to the record pair as signature, otherwise no signature is generated. The intermediate records number is $N_2 = 2 \left| \left\{ G_{r_i}^p \cap G_{r_j}^p \mid G_{r_i}^p \cap G_{r_j}^p \neq \Phi (i \neq j) \right\} \right|$. As many records in the large scale data set have more than one common prefix tokens, $N_2 \ll N_1$.

5 Experimental Evaluation

5.1 Experiment Setup

We study the performance of duplicate detection using Hadoop platform over nine computer nodes, among which one is set to the master and the others are set to the slaves. Each node is equipped with 2 cores, 2G main memory, and the operating system is Ubuntu. The maximum tasks of Map and Reduce are set

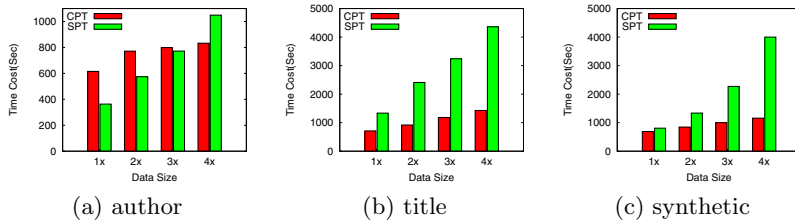


Fig. 5. Comparison

to 40, the other configurations are their default values. For this evaluation, we utilize three datasets (including two real datasets and one synthetic dataset):

- Titles: It consists of 1,772,301 titles that are extracted from DBLP² and CiteSeer³ with total size of 98M;
- Authors: It consists of 2,065,229 author names that are extracted from DBLP and CiteSeer with total size of 43M;
- Synthetic: It consists of 1,349,955 strings that are produced by Febrl⁴ with total size of 91M.

We use the following algorithms in the experiments.

- **SPT** serves as the baseline algorithm [8], where for each record r , every gram in the prefix of G_r is taken as a signature. Hence, for every two records that share multiple grams in their prefixes, multiple record pairs will be produced as the candidates;
- **CPT** is our proposed algorithm, where for every two records r_i, r_j , the longest common grams of G_{r_i} and G_{r_j} are offered as the signature. Hence, for every two records that share multiple grams in their prefixes, only one record pair is produced.

Unless otherwise specified, the threshold that we use to detect duplicates in this paper is set to 0.9.

5.2 Signature Scheme Evaluation

In this section, we study the function of both CPT and SPT over the above three datasets. We scale each dataset by copying every record for multiple times that varies from $1\times$ to $4\times$.

Fig. 5 plots the cost of detecting duplicates using CPT and SPT that apply different signature schemes. From Fig. 5(b)(c), as expected, CPT outperforms SPT. To better understand this point, we plot the distribution of longest common

² <http://www.informatik.uni-trier.de/~ley/db>

³ <http://citeseerx.ist.psu.edu>

⁴ <http://sourceforge.net/projects/febrl>

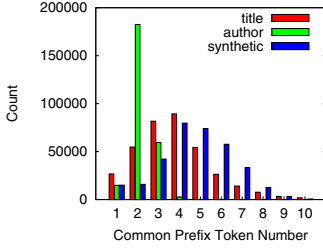


Fig. 6. Common Prefix Token Distribution

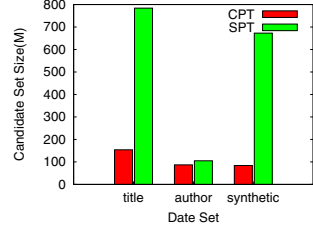


Fig. 7. Candidate Set Size

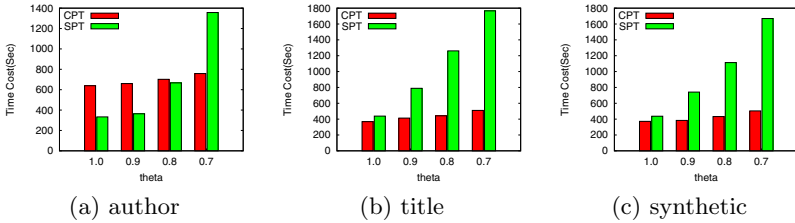


Fig. 8. Effect of threshold

prefix tokens of each candidate record pairs in Fig.6. As we can see that, over each dataset, there exist a large number of record pairs that share more than one common prefix tokens, redundant record pairs will be produced by applying SPT method. Hence, these redundant record pairs will be shuffled and verified meaninglessly. We show the size of candidate set for both SPT and CPT in Fig.7. Statistic information that is shown in Fig.6 and Fig.7 is extracted from the 1× dataset. Interestingly, SPT performs better than CPT over the author dataset when the size is small. As the record length in author data is short, the common prefix number for record pairs are almost between 2 to 3. So,when the data set size is small the cost for SPT method is smaller than CPT. When scaling the size of each dataset, the cost of both CPT and SPT increases. However, there is an obvious trend that SPT increases significantly while CPT increases smoothly.

5.3 Effect of Threshold

In this section, the effect of threshold is evaluated on 1× datasets. When the threshold decreases, prefix token number of each record and candidate pairs increase. So, the time cost increases. Fig.8(b) and (c) show that when the threshold decreases, the time cost of CPT increased smoothly, but the cost for SPT increased significantly. As threshold decreases, more candidates pairs are produced using SPT than using CPT. The special situation occurs in Fig.8(a) can be explained as in the last experiment.

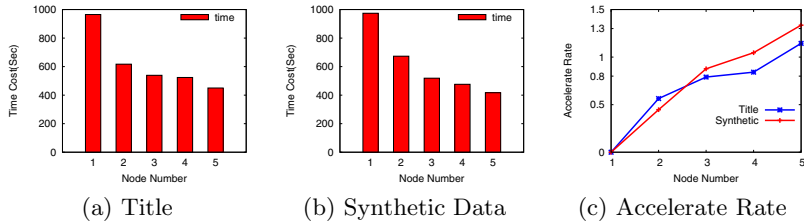


Fig. 9. Effect of Node Number

5.4 Effect of Node Number

The function of computing node number in the cluster is studied in this section. As the slave nodes in our cluster are heterogeneous, we chose five slaves that have same architecture to testify the function of node number. The $1\times$ title and synthetic datasets are used. We can see from Fig 9, the all time cost is decreased when the nodes number is increased. In Fig 9(c), we give an figure about accelerate rate. It is to demonstrate the relative improvement of performance when using more than one computing nodes contrast to using one node. We can see as the node number increased the value is increased, but it is not linearly increased as the data transfer cost is also increased. Overall, when the slave number is increased the process performance is improved.

6 Related Work

The general duplicate detection problem has been also known as merge-purge [11], record linkage [2] [3], object matching, reference reconciliation [12], deduplication and approximate string join, etc. All these works are the core of well-known data cleaning primitives. In order to improve performance, existing approaches usually have two phases: candidate generation and verification [13] [14]. In different work the candidate generation phase can be seen as signature assignment process, its goal is to put the candidates into one group by using different strategies, including BKV [3], SNN [15] and Hashing based strategy [16] [17]. When used in MapReduce to process large scale data, these methods are not effective as used in one machine. The Hashing based signature scheme is used in [18] [19]. In [20], the prefix based signature scheme is proposed and used by [8] for similarity join in MapReduce. It can be seen as an exact kind of signature method. It used the token in one record with the smallest frequency as the signature, but this method will cause many redundant candidates and verifications, especially when the record is long. In [8], it takes words in record as tokens and uses single prefix tokens as signature. It is sensitive to many typo errors. Further, when tokenized the record into n-gram tokens set, many redundant candidate pairs will be produced. To address these problems, we proposed a new signature scheme using common prefix tokens of each candidate pair and a new pruning strategy to minimize the pseudo duplicates in MapReduce framework.

7 Conclusions and Future Work

In this paper, the MapReduce framework is used for duplicate detection. New signature scheme and pruning strategies are proposed. Experiments demonstrate that they are effective to decrease the candidate set size and performance improvement. In the future, we will do further work on other signature schemes and pruning strategies using different strategies.

Acknowledgements. This work is partly supported by National 863 High Technology Program (No. 2009AA01Z149), the Important National Science & Technology Specific Projects of China (“HGJ” Projects, No.2010ZX01042-002-002-03), the Fundamental Research Funds for the Central Universities (the Research Funds of Renmin University of China, No.10XNI018), Shanghai Key Lab. of Intelligent Information Processing, China(No. I IPL-09-018).

References

1. Arasu, A., Ré, C., Suciu, D.: Large-scale deduplication with constraints using dedupalog. In: ICDE, pp. 952–963 (2009)
2. Fellegi, I., Sunter, A.: A theory for record linkage. *Journal of the American Statistical Association* 64(328), 1183–1210 (1969)
3. Gu, L., Baxter, R., Vickers, D., Rainsford, C.: Record linkage: Current practice and future directions. CSIRO Mathematical and Information Sciences Technical Report 3, 83 (2003)
4. Xiao, C., Wang, W., Lin, X., Shang, H.: Top-k set similarity joins. In: ICDE, pp. 916–927 (2009)
5. Wang, J., Feng, J., Li, G.: Trie-join: Efficient trie-based string similarity joins with edit-distance constraints. *PVLDB* 3(1-2), 1219–1230 (2010)
6. Zhang, Z., Hadjieleftheriou, M., Ooi, B., Srivastava, D.: Bed-tree: an all-purpose index structure for string similarity search based on edit distance. In: SIGMOD, pp. 915–926 (2010)
7. Xiao, C., Wang, W., Lin, X., Yu, J.: Efficient similarity joins for near duplicate detection. In: WWW, pp. 131–140 (2008)
8. Vernica, R., Carey, M., Li, C.: Efficient parallel set-similarity joins using MapReduce. In: SIGMOD, pp. 495–506 (2010)
9. Arasu, A., Ganti, V., Kaushik, R.: Efficient exact set-similarity joins. In: VLDB, pp. 918–929 (2006)
10. White, T.: *Hadoop: The Definitive Guide*. Yahoo Press (2010)
11. Hernández, M., Stolfo, S.: The merge/purge problem for large databases. In: SIGMOD, p. 138 (1995)
12. Dong, X., Halevy, A., Madhavan, J.: Reference reconciliation in complex information spaces. In: SIGMOD, pp. 85–96 (2005)
13. Elmagarmid, A., Ipeirotis, P., Verykios, V.: Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 1–16 (2007)
14. Naumann, F., Herschel, M.: An Introduction to Duplicate Detection. *Synthesis Lectures on Data Management* 2(1), 1–87 (2010)
15. Hernández, M., Stolfo, S.: Real-world data is dirty: Data cleansing and the merge/purge problem. *Data Mining and Knowledge Discovery* 2(1), 9–37 (1998)

16. Broder, A., Glassman, S., Manasse, M., Zweig, G.: Syntactic clustering of the web. *Computer Networks and ISDN Systems* 29(8-13), 1157–1166 (1997)
17. Henzinger, M.: Finding near-duplicate web pages: a large-scale evaluation of algorithms. In: *SIGIR*, pp. 284–291 (2006)
18. Cohen, E., Datar, M., Fujiwara, S., Gionis, A., Indyk, P., Motwani, R., Ullman, J., Yang, C.: Finding interesting associations without support pruning. *IEEE Transactions on Knowledge and Data Engineering* 13(1), 64–78 (2002)
19. Kim, H., Lee, D.: HARRA: fast iterative hashed record linkage for large-scale data collections. In: *EDBT*, pp. 525–536 (2010)
20. Chaudhuri, S., Ganti, V., Kaushik, R.: A primitive operator for similarity joins in data cleaning. In: *ICDE* (2006)

A Secure and Efficient Role-Based Access Policy towards Cryptographic Cloud Storage

Cheng Hong*, Zhiquan lv, Min Zhang, and Dengguo Feng

The State Key Laboratory Of Information Security,
100080, Beijing, China

{hongcheng, lvzhiquan, mzhang, feng}@is.iscas.ac.cn

Abstract. Cloud Storage, which provides cost-efficient and scalable storage services, has emerged as a hot paradigm today. As promising as it is, Cloud Storage also brings forth security challenges. Sensitive data may be outsourced for sharing on cloud storage servers, which are not within the same trusted domain as the data owner (DO). To keep the data confidential against unauthorized parties, cryptographic access control must be applied. Existing methods usually require the access policies be fully managed by the DO, which could lead to the DO-side bottleneck. This paper addressed the issue by implementing a cryptographic Role-Based Access Control via CP-ABE. The access policies are divided into two parts: Permission Assignments (PAs) and Role Assignments (RAs), and we develop an approach called propagation to allow RAs to be handled effectively by users besides the DO. Since most of the dynamic policies in the Cloud are triggered by RAs, the bottleneck could be successfully avoided.

Keywords: CP-ABE, RBAC, Cloud Computing, Cloud Storage.

1 Introduction

With the big trend of Cloud Computing, Cloud Storage has become one of the most popular storage solutions for IT Enterprise today. Cloud Storage Provider (CSP) combines data centers and transfers them into pools of data storage services, offering users cost-efficient and hardware-independent interfaces. Currently there are many CSPs available, including Amazon S3, Google Docs and Microsoft Azure, etc.

Despite of the flexibility of Cloud Storage, security concerns have risen upon the fact that the customers' data no longer reside within their physical possession. Those data may contain sensitive information that should not be revealed to unauthorized parties. Such a security goal can be achieved by making a server-side access control in traditional storage systems, but in Cloud Storage, the CSP may not be trustworthy, thus server-side access control could not be relied upon.

* Research supported by Key National Science and Technology Specific Projects of China (2010ZX01042-001-001-05), The Knowledge Innovation Program of the Chinese Academy of Sciences (NO.YYYJ-1013).

Cryptographic access control has been widely researched to protect the data confidentiality in the Cloud. Plutus [1] and SiRiUS [2] are the earliest ones, but their key assignment schemes do not scale well with the number of growing users. J.Bethencourt et.al [4] introduced the CP-ABE algorithm, where users are given private keys according to their user attributes, and ciphertexts are attached with access trees. Only if the user’s private key has the satisfying attributes can he perform decryption. KP-ABE [3] acts in a similar way except that its ciphertexts are attached with attributes, and private keys are associated with access trees. Both CP-ABE and KP-ABE integrate the access policies into cryptographic algorithms, and no further key distribution is needed after every one receives his private key, thus the complexity of key assignment is greatly reduced. Quite a few researches [5, 6, 15] have been made to design cryptographic access control based on ABE algorithms. In the proposals of [5] and [15], permission revocations are partially delegated to the CSP, but it is not safe if the CSP cannot hold the secret. How to design a cost-efficient cryptographic access control in untrusted Cloud Storage remains an open topic.

The Role-based Access Control (RBAC) model [16] is widely adapted in traditional storage systems to simplify the policy administrations. Several researches [17, 18] have been made to introduce RBAC to Cloud Storage, but they paid few attentions to dynamic policies.

In this work, we design and implement a cryptographic RBAC system upon Cloud Storage via CP-ABE. Our main contribution is that we deal with dynamic policies efficiently. The DO can appoint other users to modify the role-user assignments, which alleviate his administering burdens and avoid the bottleneck.

2 Preliminaries

2.1 RBAC

In the Role-Based Access Control model, access permissions are assigned to roles rather than users, and users must activate a role to gain permissions [16]. Figure 1 shows the basic components of role-based access control.

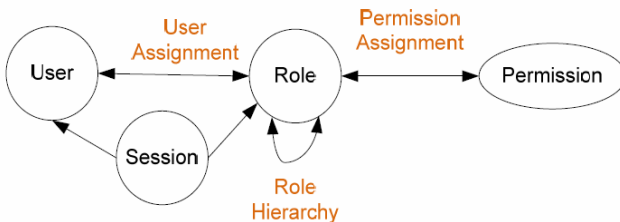


Fig. 1. Components of RBAC

For each user, the active role is the one that the user is currently using:

$$AR(u : user) = \{the\ active\ role\ for\ u\}$$

One or more permissions may be assigned to each role:

$$PA(r : role) = \{ permissions \text{ assigned to } r \}$$

One or more roles may be assigned to each user:

$$RA(u : user) = \{ roles \text{ assigned to } u \}$$

Predicate $permit(u,p)$ is used to denote whether user u has permission p :

$$permit(u : user, p : permission) = true \text{ iff } s \text{ has } p.$$

Three basic rules are required:

- 1) A user must select a role before receiving permission:

$$\forall u : user, p : permission, (permit(u, p) \Rightarrow AR(u) \neq \emptyset)$$

- 2) A user can only activate a role assigned to him:

$$\forall u : user, (AR(u) \subseteq RA(u))$$

- 3) A user has a permission only if the permission is authorized for the user's active role:

$$\forall u : user, p : permission, (permit(u, p) \Rightarrow p \in PA(AR(u)))$$

RBAC divides security policies into PAs and RAs , and RAs can be modified without changing the underlying access structure of PAs . Thus the administration of security policies could be simplified.

2.2 CP-ABE

CP-ABE [4] is a public key cryptography primitive for one-to-many communications. In CP-ABE, a user's private key is associated with a set of attributes, and the encryptor encrypts the file with an access tree. The access tree is organized in a way that its interior nodes are threshold gates and its leaf nodes are associated with user attributes. The user is able to decrypt if and only if his attributes satisfy the access tree. An example is shown in Figure 2.

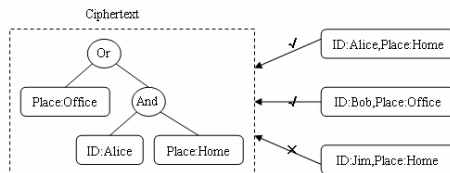


Fig. 2. Example of CP-ABE

The CP-ABE scheme is composed of four algorithms which can be defined as follows:

- **Setup:** The setup algorithm chooses a bilinear group G_0 of prime order p with generator g and the bilinear map function $e: G_0 * G_0 \rightarrow G_1$, then it chooses two random exponents $a, b \in \mathbb{Z}_p$. The public key is published as:

$$PK = \{G_0, g, g^b, e(g, g)^a\}$$

The master key $MK = \{b, g^a\}$ is kept only by the key generator.

- **Encrypt(PK, T, M):** The encryption algorithm encrypts a message M under access tree T . Each node n in the tree T (including the leaves) is associated with a polynomial $q_n(x)$. These polynomials are chosen in the following way: Let k_n be the threshold value of node n (i.e., for “AND” gates, $k_n = 2$, for leaf nodes and “OR” gates, $k_n = 1$). The degree d_n of the polynomial q_n must satisfy $d_n = k_n - 1$. Starting with the root node R the algorithm chooses a random $s \in \mathbb{Z}_p$ and choose polynomial $q_R(x)$ so that $q_R(0) = s$. For any other node n , it choose polynomial $q_n(x)$ so that $q_n(0) = q_{parent(n)}(index(n))$ in a top-down manner, where $index(n)$ means the index of n among its siblings. Let Y be the set of leaf nodes in T , $att(y)$ be the attribute associated with node y , the ciphertext can be constructed by computing

$$CT = \{T, C^r = Me(g, g)^{as}, C = g^{bs}, \forall y \in Y : C_y = g^{q_s(0)}, C'_y = H(att(y))^{q_s(0)}\},$$

where H is a hash function.

- **KeyGen(MK, S):** The key generation algorithm will take as input a set of attributes S and output a private key that identifies with that set. The algorithm first chooses a random $r \in \mathbb{Z}_p$, and then random $r_j \in \mathbb{Z}_p$ for each attribute $j \in S$. Then it computes the key as

$$SK = \{D = g(a+r)/b, \forall j \in S : Dj = grH(j)r_j, D'j = grj\}$$

- **Decrypt(CT, SK):** The decryption procedure is a long and recursive algorithm. The exact procedure can be found in [4]. Lacking of space, we write:

$$Decrypt(CT, SK) = M;$$

CP-ABE allows the fine-grained access policies to be integrated into the ciphertext, and is adapted in many researches on cryptographic Cloud Storages.

3 Construction

3.1 Security Assumption

Quite a few related researches [5, 7, 8] are designed under the assumption that the server is “honest but curious”, but this assumption may be too optimistic to suit the Cloud. S.Yu et al. [5] and X.Tian et al. [7] take advantage of a CSP-side re-encryption to reduce the cost of revocation. But the re-encryption keys may be leaked by an untrusted CSP. S.D.C Vimercati et al. [8] introduced a two-layer encryption, but a revoked user who already knows the first layer keys may collude with an untrusted CSP

to get the second layer keys and keep accessing. Because of the reasons mentioned above, we have to propose a weaker security assumption.

Definition 1. Non-Secrecy but Cooperative (NSC): The CSP will try the best in order to make the sensitive data visible with third parties, but it will act cooperatively as the DO required if the target cannot be reached (Uncooperative CSPs come down to the issues of Data Integrity and Data Availability, hence they are out of our topic).

The NSC assumption is closer to the real Cloud, and our security goal is to achieve data confidentiality under NSC CSPs.

3.2 System Setup

We design a cryptographic access control system, implementing most of the features in RBAC. Its components are defined as follows:

User: Users are subjects of the access control. Each user u is associated with an attributes set A_u that denotes his status. A unique user ID of u is included too.

File: Each file f is encrypted with a symmetric key k_f and stored in the storage of CSP.

Permission: There're two kinds of permissions in our proposal: The ability to read and write files. We assume that the write permissions are given to the DO, and users other than the DO should receive read permissions only. Users with write permissions can be straightforwardly designed via signatures as described in [1], thus they are not discussed here for simplicity. In the rest paper we will use p_f to denote the read permission on file f .

Role: Each role r is associated with an access tree T_r and a key set S_r .

PA and RA are defined in the same way as described in Section II:

$$RA(u : user) = \{roles\ assigned\ to\ u\} \quad PA(r : role) = \{permissions\ assigned\ to\ r\}$$

The following rules must be satisfied:

Rule 1: $\forall r : role, p_f : permission, (p_f \in PA(r) \Rightarrow k_f \in S_r)$

Rule 2: $\forall u : user, r : role, (r \in RA(u) \Rightarrow A_u\ satisfies\ T_r)$

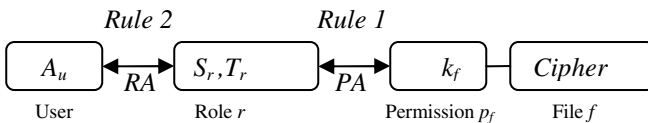


Fig. 3. PA and RA

In order to implement the rules above, we introduce the notion of **lockbox**. Each role r is associated with a lockbox L_r , which is divided into two parts: The lockbox header L_{rh} , which stores the RA information, contains a symmetric key k_r that is encrypted with T_r via CP-ABE. The lockbox body L_{rc} , which stores the PA information,

contains a key set S_r that is encrypted by k_r . With lockbox defined, we can describe the steps **Key Assignment**, **File Upload** and **Role Create** as follows. $SymEnc()$ and $SymDec()$ are used to denote any symmetric-key enc/dec algorithm.

- **Key Assignment:** The DO first chooses a secret parameter SP as input and runs *Setup* to generate the public parameter PK and master key MK . The DO signs PK and sends PK along with the signature to the CSP, and keeps MK himself.

The DO runs *KeyGen* so that each user of the Cloud is assigned a private key with his corresponding attributes. i.e., for user u with attribute set A_u , the DO generate the private key $sk_u = KeyGen(A_u, MK)$, and sent sk_u to u via a secure channel.

- **File Upload:** Before uploading a file f , the DO generates a random symmetric key k_f . Then he encrypts f with k_f , signs the ciphertext, and uploads the signed ciphertext to the CSP.
- **Role Create:** Creating a role r is implemented by creating its corresponding lockbox L_r . Whenever a role r with key set S_r and access tree T_r is to be created, the DO generates a random symmetric key k_r . He encrypts k_r with T_r to generate the lockbox header L_{rh} , and encrypts S_r with k_r to generate the lockbox content L_{rc} . The lockbox L_r is created as follows and sent to the CSP:

$$L_r = \{L_{rh} = Encrypt(PK, T_r, k_r), L_{rc} = SymEnc(k_r, S_r)\}$$

3.3 Handling Dynamic Policies

In this section we will describe how to deal with the dynamic policies, including *PA* modifications, *RA* modifications and file updates. Normally all of these operations have to be done by the DO, which may lead to a DO-side bottleneck. To solve the problem, we make improvements to the basic construction and introduce **propagation**.

3.3.1 PA Modification

PA Assignment

If the DO wants to assign the permission of file f to r , he has to insert k_f into S_r . The DO decrypts L_r via his private key to get S_r exactly the same as doing **File Access**, forms the new key set: $S'_r = S_r \cup \{k_f\}$ and generates the new $L'_r = \{L_{rh} = Encrypt(PK, T_r, k_r), L'_{rc} = SymEnc(k_r, S'_r)\}$. Then he sends L'_r to the CSP.

PA Revocation

To stop r from accessing f , normally a new symmetric key has to be used to re-encrypt f . In order to improve the efficiency, we introduce **lazy-re-encryption**, which means the DO just flags f as “to-be-encrypted”, and will not re-encrypt f until a new version of f is uploaded. Since there’s no technical way to stop users from visiting the data they have visited before, lazy-re-encryption is reasonable.

With lazy-re-encryption, the procedure of *PA* revocation can be described as follows: The DO decrypts L_r via his private key to get S_r , forms the new key set: $S'_r =$

$S_r | \{k_f\}$ and generates the new $L'_r = \{ L_{rh} = \text{Encrypt}(PK, Tr, kr), L'_{rc} = \text{SymEnc}(kr, S'r) \}$. Then the DO flags f as “to-be-encrypted” and sends L'_r to the CSP.

3.3.2 RA Modification

RA Assignment/Propagation

As indicated in [16], it’s desirable to allow users besides the DO to make the RA assignments without giving them the authority to manage PA assignments. It has already been implemented in traditional database systems: the DBMS server defines a “Grant” permission, and enforces that only those who have the “Grant” permission could make the corresponding RA assignments. But in the Cloud scenarios, the server is not fully trusted, and the “Grant” permission could not be relied upon, thus we have to seek new solutions.

In the following paragraph we design an approach called *propagation* to solve the problem. An example of propagation is given in Figure 4.

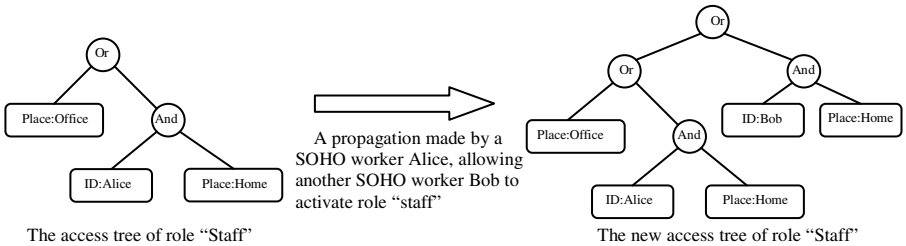


Fig. 4. Propagation

Given a role r and its access tree T_r , r can be propagated to user u in two steps: **I.** Construct a new access tree $T'_r = T_r \vee T_u$, where T_u is the conjunctive form constructed by u 's attribute set. i.e, for $A_u = \{ID:Bob, Place:Home\}$, $T_u = (ID:Bob) \wedge (Place:Home)$. **II.** Make a CP-ABE re-encryption, changing $L_{rh} = \text{Encrypt}(PK, T_r, k_r)$ into $L'_{rh} = \text{Encrypt}(PK, T'_r, k_r)$.

In order to allow the DO to control the "propagation" permission. We take further actions as follows:

Let’s observe the difference between L_{rh} and L'_{rh} :

$$L_{rh} = \text{Encrypt}(PK, T_r, M) = \{ T_r, Me(g, g)^{as}, g^{bs}, \forall y \in Y : g^{q_y(0)}, H(att(y))^{q_y(0)} \}$$

$$L'_{rh} = \text{Encrypt}(PK, T'_r, M) = \{ T_r \square T_u, Me(g, g)^{as'}, g^{bs'}, \forall y \in Y' : g^{q'_y(0)}, H(att(y))^{q'_y(0)} \}$$

If the access tree changes from T_r to T'_r , the root node of T_r (say R) will be the left child of the new root node (say R'). Thus the new polynomial of R must satisfy

$$q'_R(0) = q'_{parent(R)}(index(R)) = q'_R(0) = s'$$

As we know, $q_R(0) = s$. This means if the security parameter s is not changed during the propagation, the polynomials related to the old tree R does not need to change.

Thus the "propagation" permission can be controlled by distributing the security parameter s . If the DO wants to give user u the "propagation" permission for role r , he sends s to u via a secure channel (It's obvious that knowing s involves the ability to decrypt L_r). If u is no longer allowed to propagate r , the DO will use a new parameter s' to update L_r and send s' to all the other propagators of r except u .

Now the procedure of propagating r to v by u can be described: u generate polynomials for A_v without changing the CP-ABE security parameter s , and forms the new lockbox header: $L'_{rh} = \{ T_r \vee T_v, k_r e(g, g)^{as}, g^{bs}, \forall y \in Y \cup A_v : C_y = g^{q_y(0)}, C'_y = H(att(y))^{q_y(0)} \}$. With L_{rc} unchanged, p sends $L_r = \{ L'_{rh}, L_{rc} \}$ to the CSP.

Note that all the polynomials related to T_r do not need changes, thus our implementation of propagation also reduce the time cost of an RA assignment from $O(|Y|+|A_v|)$ to $O(|A_v|)$.

RA Revocation

Revoking user u from role r can be implemented by replacing $L_{rh} = \text{Encrypt}(PK, T_r, k_r)$ with $L'_{rh} = \text{Encrypt}(PK, T_r \setminus T_u, k_r)$, where we use $T_r \setminus T_u$ to denote the result of pruning T_u from T_r . Additionally, all of the symmetric keys in the key set S_r have to be updated to prevent u from further access. Similarly, there's no need of instant operations, and those related data files will just be flagged as "to-be-encrypted".

3.3.3 File Update

When the DO is about to update a data file f , he checks if f is flagged as "to-be-encrypted". If no: the symmetric key k_f used does not change. Otherwise, the DO randomly chooses a new symmetric key k'_f to encrypt the new version of f , and uploads the ciphertext to the CSP. Additionally, for roles that have access to f , their lockboxes must be updated. i.e. for role r that $k_f \in S_r$, its old lockbox is:

$$L_r = \{ L_{rh} = \text{Encrypt}(PK, T_r, k_r), L_{rc} = \text{SymEnc}(k_r, S_r) \}.$$

Firstly the DO has to check the validity of L_{rh} as described in the **Propagation** section, then he randomly chooses a new lockbox key k'_r , modifies S_r to $S'_r = (S_r \setminus \{k_f\}) \cup \{k'_f\}$, generates

$$L'_r = \{ L'_{rh} = \text{Encrypt}(PK, T_r, k'_r), L'_{rc} = \text{SymEnc}(k'_r, S'_r) \},$$

and send the CP-ABE security parameter s used in L'_{rh} to all the valid propagators of r . A **File Update** may trigger more than one lockbox updates, but advantages are that after k'_f is determined, the lockbox updates become independent with the file, and can be run in parallel with file encryption and file upload.

4 Security Analysis

Our scheme makes use of the CP-ABE and symmetric encryption algorithm in a straightforward way, except that the CP-ABE security parameter s used in the current

ciphertext is given away to the propagators. We will prove that the propagator benefits nothing more than to decrypt the current ciphertext and to propagate. It's proved by giving security games as follows:

Game 0: Security game of CP-ABE

- **Setup.** The challenger runs the Setup algorithm and gives the public parameters PK to the adversary.
- **Phase 1.** The challenger runs KeyGen to generate private keys SK_1, \dots, SK_q corresponding to sets of attributes $A = \{A_1, \dots, A_q\}$ and send them to the adversary.
- **Challenge.** The adversary submits two equal length messages M_0 and M_1 . In addition the adversary gives a challenge access tree T such that none of the sets A_1, \dots, A_q from Phase 1 satisfy T . The challenger flips a random coin b , and encrypts M_b under T . The ciphertext CT is given to the adversary.
- **Phase 2.** Phase 1 is repeated and more private keys SK_{q+1}, \dots, SK_n are sent to the adversary with the restriction that none of A_{q+1}, \dots, A_n satisfy T .
- **Guess.** The adversary outputs a guess b' of b . The advantage of an adversary in this game is defined as $\Pr[b' = b] - 1/2$.

Game 1: Security game of our scheme

In Game 1, the adversary has extra information about another ciphertext (let's call CT_N) that is independent of CT . Thus a new Phase 1 has to be designed as follows.

- **Phase 1.** The challenger runs KeyGen to generate private keys SK_1, \dots, SK_q corresponding to sets of attributes A_1, \dots, A_q and send them to the adversary. The adversary gives a challenge access tree T' such that there exists an $A_i \in A$ satisfy T' . The challenger encrypts a random message N under T' with a random security parameter s , and sends s and the ciphertext CT_N to the adversary.

The steps **Setup**, **Challenge**, **Phase 2** and **Guess** are the same for **Game 0** and **Game 1**.

J. Bethencourt et al. [4] has proved that CP-ABE is secure under random oracles with **Game 0**. We can prove that if there is a polynomial time algorithm A that wins **Game 1** with non-negligible advantage, it can be used to construct a polynomial time algorithm B to win **Game 0**. The construction is shown as follows.

Algorithm A:

Input:

T ;
 $CT_A = \text{Encrypt}(PK_A, T, b_A)$;
 PK_A ;
 $\{SK_{iA}\} : 1 < i < m$;
 $CT_N = \text{Encrypt}(PK_A, T', N)$;
 s ; /* s is used in CT_N */

Output: b'_A , a guess of b_A

Algorithm B:

Input:

$A;$
 $T;$
 $CT_B = \text{Encrypt}(PK_B, T, b_B);$
 $PK_B;$
 $\{SK_{iB}\}: 1 < i < m;$

Output: b'_B , a guess of b_B

1. random choose t from Z_p ;
2. random choose a message M ;
3. choose an access tree T' such that there exists an $A_i \in A$ satisfy T' .
4. compute the following ciphertext:

$$CT_M = \text{Encrypt}(PK_B, T', M)$$

$$= \{T, C = \text{Me}(g, g)^{at}, C = g^{bt},$$

$$\forall y \in Y : C_y = g^{ay^{(0)}}, C'_y = H(\text{att}(y))^{ay^{(0)}}\};$$

5. return $b'_B = A(T, CT_B, PK_B, \{SK_{iB}\}, CT_M, t)$

As the construction shows, our scheme is equivalent in security to the CP-ABE scheme, thus we successfully achieve data confidentiality with NSC CSPs.

5 Performance

This section numerically evaluates the performance of our proposed scheme. We design the experiments based on the scenarios listed in Table 1. The test machine is an Intel Xeon 2.4 GHz client with 1GB memory running Red Hat Enterprise 5. The coding is based on the cpabe-0.10 library.

Table 1. Cloud storage scenarios

	Firm A	Firm B	Firm C
Number of roles	10	50	100
Number of users	100	1000	10,000
Average number of roles assigned to each user	2	5	10
Number of files	10,000	100,000	1,000,000
Average number of files assigned to each role	1000	10,000	100,000

5.1 Average Performance

Table II presents the average client side overhead introduced per single operation. **File Update** is relatively slow here because there may be many roles with access to the file to update. As described in section 3, the lockbox updates in **File Update** can be done in parallel with file encryption and file upload, which is the key time-consuming operation in most cases (depending on the size of file and network condition), thus we consider the overhead acceptable.

Table 2. Average overhead

Operations	Time cost (in milliseconds)		
	Firm A	Firm B	Firm C
RA assignment	173	193	194
RA revocation	160	186	189
PA assignment	144	192	265
PA revocation	160	197	293
File Update	417	1,870	4,200
File Access	140	185	260

5.2 Benefits from Propagation

The NIST study [12] indicates that *PA* policies, unlike *RA* policies, tend to change relatively slowly. Thus propagation could make the DO’s task easier. Since the frequency of policy changes is hard to be measured numerically, we evaluate the effect of propagation qualitatively in Table III.

Table 3. Performance with & without propagation

Operations	Without propagation		With propagation			Frequency
	Time cost	Must be done by DO?	Time cost	Must be done by DO?	by	
PA assignment	$O(A) + O(S)$	Yes	$O(A) + O(S)$	Yes	by	Low
PA revocation	$O(A) + O(S)$	Yes	$O(A) + O(S)$	Yes	by	Low
RA assignment	$O(Y)$	Yes	$O(A)$	No	by	High
RA revocation	$O(Y)$	Yes	$O(A)$	No	by	High

* $|S|$ denotes the average number of files available to a role, $|Y|$ denotes the average number of leaf nodes of a role’s access tree, and $|A|$ denotes the average size of a user’s attribute set. In most cases $|Y| >> |A|$

As is shown, the administering overhead of DO could be greatly reduced, especially when the percentage of *RA* changes rise, thus the DO-side bottleneck could be successfully avoided.

6 Conclusion

RBAC was proposed to support complicated security policies and simplify authorization administration. This primitive has been explored by researchers and well

implemented in traditional storage systems, but little work has been done to implement dynamic RBAC on untrusted Cloud Storage yet. We proposed a cryptographic RBAC policy to alleviate the DO's administering task in the Cloud. Role propagation is introduced, allowing the frequent user-role permission changes to be handled by users other than the DO. Our scheme is proven secure against CSP with Non-Secrecy, which represents a typical problem in the Cloud. Experiments show that we successfully avoid the DO side bottleneck, and the overall performance is acceptable.

References

1. Kallahalla, M., Riedel, E., Swaminathan, R., Wang, Q., Fu, K.: Plutus-scalable secure file sharing on untrusted storage. In: Proceedings of the Second USENIX Conference on File and Storage Technologies (FAST). USENIX (March 2003)
2. Goh, E., Shacham, H., Modadugu, N., Boneh, D.: SiRiUS: Securing remote untrusted storage. In: NDSS (2003)
3. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute Based Encryption for Fine-Grained Access Control of Encrypted Data. In: ACM Conference on Computer and Communications Security, ACM CCS (2006)
4. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: Proceedings of 2007 IEEE Symposium on Security and Privacy (2007)
5. Yu, S., Wang, C., Ren, K., Lou, W.: Achieving secure, scalable, and fine-grained data access control in cloud computing. In: Proceedings of IEEE INFOCOM 2010, pp. 15–19 (2010)
6. Hong, C., Zhang, M., Feng, D.: AB-ACCS: A cryptographic access control scheme for cloud storage. In: NDBC 2010 (2010)
7. Tian, X., Wang, X., Zhou, A.: DSP RE-Encryption: A Flexible Mechanism for Access Control Enforcement Management in DaaS. In: 2009 IEEE International Conference on Cloud Computing, pp. 25–32 (2009)
8. De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Over-encryption: Management of access control evolution on outsourced data. In: Proc. of VLDB 2007, Vienna, Austria (2007)
9. Ferraiolo, D., Cugini, J., Kuhn, R.: Role-based access control: Features and motivations. In: Proceedings of the Annual Computer Security Applications Conference. IEEE Press, Los Alamitos (1995)
10. Atallah, M.J., Frikken, K.B., Blanton, M.: Dynamic and efficient key management for access hierarchies. In: Proceedings of 12th ACM Conference on Computer and Communications Security, pp. 190–202 (2005)
11. Crampton, J., Martin, K., Wild, P.: On key assignment for hierarchical access control. In: Proc. Of the 19th IEEE CSFW 2006, Venice, Italy (July 2006)
12. Ferraiolo, D.F., Gilbert, D.M., Lynch, N.: An Examination of Federal and Commercial Access Control Policy Needs. In: Proc. NIST-NCSC National Computer Security Conf., Nat'l. Inst. Standards and Technology, Gaithersburg, Md., pp. 107–116 (1993)
13. Cachin, C., Keidar, I., Shraer, A.: Trusting the cloud. ACM SIGACT News 40(2), 81–86 (2009)
14. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. In: ACM CCCS (2007)
15. Yu, S., Wang, C., Ren, K., Lou, W.: Attribute based data sharing with attribute revocation. In: ASIACCS (2010)

16. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. *IEEE Computer* 29(2), 38–47 (1996)
17. Malek, B., Miri, A.: Combining attribute-based and access systems. In: Muzio, J.C., Brent, R.P. (eds.) *Proc. IEEE CSE 2009, 12th IEEE Int'l Conf. on Computational Science and Engineering*, pp. 305–312. IEEE Computer Society, Los Alamitos (2009)
18. Narayanan, H.A.J., Güneş, M.H.: Ensuring Access Control in Cloud Provisioned Health-care Systems, <http://www.cse.unr.edu/~mgunes/papers/eHealth11.pdf>

Tagging Image by Exploring Weighted Correlation between Visual Features and Tags

Xiaoming Zhang¹, Zhoujun Li¹, and Yun Long²

¹ School of Computer Science and Engineering, Beihang University, Beijing, China

² Economic and Management Department, University of South China, Hengyang, China
{yolixs,lizj}@cse.buaa.edu.cn, longyun_2003@yahoo.com.cn

Abstract. Automatic image tagging automatically label images with semantic tags, which significantly facilitate image search and organization. Existing tagging methods often derive the probabilistic or co-occurring tags from the visually similar images, which based on the image level similarity between images. It may result in many noisy tags due to the problem of semantic gap. In this paper, we propose a novel automatic tagging algorithm. It represents each test image with a bag of visual words and a measure to estimate the correlation between visual words and tags is designed. Then, for each test image, its visual words are weighted based on their importance, and the more important visual word contributes more to tag the test image. To tag a test image, we select the tags which have strong correlation with the greatly weighted visual words. We conduct extensive experiments on the real-world image dataset downloaded from Flickr. The results confirm the effectiveness of our algorithm.

Keywords: image retrieval, image tagging, feature correlation.

1 Introduction

Recently we have witnessed an explosion of web images available online (e.g. Flickr), which requires an effective image search technology. Although Content-based image search [1] has been researched for more than a decade, it still has several bottlenecks. First, the visually similar images aren't always similar in semantic meaning, which is also the semantic gap problem. This is because current visual feature extraction techniques are not effective enough in representing the semantics of an image. The second problem is computational expensiveness. Due to the high dimensionality of visual features, the efficiency and scalability of CBIR are usually very low. The problems are further exacerbated in the social media environment because of its' huge volumes and the noise generated by users. Image tagging is the task that assigns an image with several descriptive keywords called tags. With the tags, images can be searched like web documents [16]. Thus, a set of quality image tags is very useful to alleviate with the problems existing in Content-based image search. There are already some existing web sites which allow users to upload personal images and also tag them manually. However, manually tagging images is intellectually expensive and time consuming. Furthermore, persons or community provided tags lack consistency and present numerous irregularities (e.g. abbreviations and mistypes) [2].

Thus, automatic image tagging has recently attracted much attention [3], [4], [5], [6], [17], [18]. However, it remains a challenging problem to the image and video tagging due to unsatisfactory performance. There are several problems. First, many existing methods which tag the test image based on visual content often heavily rely on complicated machine learning algorithms. In general, the machine learning based methods boil down to learning a mapping between low level visual features and high-level semantic concepts. Since the potentially vocabulary existing in social tagging is unlimited, only a very limited number of visual concepts can be modelled using small-scale datasets. The complexity of training and the significant diversity of visual appearance might make the learned models unreliable and hardly generalizable. Some other existing automatic tagging approaches suggest the most common tags derived from the visually similar images to users. However, the visually similar images aren't always semantically similar. As a result, many noisy tags are assigned to the test images. In the other side, these approaches are usually based on the image level similarity. Since a single image may contain several objects, the image level similarity can hardly reflect the truth. For example, when tag a test image which contains airplane only, the tags associated with the image which contains airplane and cloud may be ignored because the two images are less similar.

In this paper, we propose a novel automatic tagging algorithm which represents each image with a bag of visual words, and the tags which are strongly correlative to the important visual words are assigned to the test image. In the training process, a measure to estimate the correlation between visual words and tags is designed based on their co-occurrence. Based on the assuming that the more important a visual word is the more it contributes to tag the test image, the visual words of the test image are ranked and weighted based on the visual word association graph. Then we select the tags which are most correlative to the greatly weighted visual words to tag the test images. Since an image can be reconstructed from the subspaces of other images, the correlation between visual words and tags is more informative than the correlation between images and tags. Then, by representing each image with a bag of visual words and exploring the correlation between visual words and tags, the tags which are most descriptive to the objects of the test images are selected, which can alleviate the problem existing in the image-level similarity based approaches. We conduct extensive experiments on real-world datasets downloaded from Flickr. Results show that our method outperforms existing methods significantly.

The rest of the paper is organized as follows. Section 2 reviews the related work. We introduce the estimation of correlation between visual words and tags in section 3, followed by visual words weighting in Section 4 and tagging the test image in Section 5. Experimental results are reported in Section 6. We conclude the paper in Section 7.

2 Related Works

Image tagging has attracted more and more interests in recent years. The key problem is to decide which tag can be assigned to tag the query image. Some works use a supervised machine learning method to tag the query images. For example, Lei et al. [6] proposes a multi-modality recommendation model based on both tag and visual correlation of images. Rankboost algorithm is then applied to learn an optimal combination

of those ranking features from different modalities. Barnard et al. [7] develops a number of models for the joint distribution of image regions and words to tag the query images. Lei et al. [8] propose a novel probabilistic distance metric learning scheme that automatically derives constraints from the uncertain side information and efficiently learns a distance metric from the derived constraints. To tag a query image, the proposed approach first retrieves k social images that share the largest visually similarity with the query image. The tags of the query image are then derived based on the tagging of the similar images. Geng et al. [9] model the concept affinity as a prior knowledge into the joint learning of multiple concept detectors, and then the concept detectors which also are classifiers are used to tag the query images. Wu et al. [19] formulate image tagging as multi-label image annotation which is treated as a regression model with a regularized penalty. However, these machine learning methods try to learning a mapping between low-level visual features and high-level semantic concepts, which are not scalable to cover the potentially unlimited array of concepts existing in social tagging. Moreover, uncontrolled visual content generated by users creates a broad domain environment which has a significant diversity in visual appearance, even for the same concept.

Some other works use an example based on a training dataset to tag the query images. Timothée et al. [10] present a random walk graph-based scheme founded on the GCap method to perform automatic image tagging. It uses the canonical correlation analysis technique (CCA) to shorten the semantic gap in the image space and defines a new metric in the text space to correlate tags with content. Zhou et al. [3] uses a heuristic and iterative algorithm to estimate the probabilities that words are in the caption of an image by examining its surrounding text and region matching. Words with high probabilities are selected as tags. The approach in [4] first estimates initial relevance scores for the tags based on probability density estimation, and then performs a random walk over a tag similarity graph to refine the relevance scores. There is also a learning of social tag relevance by neighbour voting [5], which learns tag relevance by accumulating votes from visual neighbours. However, these approaches usually assigned the tags which appear frequently in the visually similar images, and many noisy tags may be introduced due to the semantic gap problem.

3 Correlation between Visual Words and Tags

One of the most important parts of the image tagging is how to reduce the semantic gap between visual features and tags. This task is very difficult for computers to perform. In the next two sections we propose an algorithm to reduce the semantic gap by exploring the correlation between visual features and tags, and then learn the weighted visual features of the test image. By combining the weight of features and their correlation with tags, the tags which are most relative to the test image are assigned. In this section, we first generate the visual vocabulary and representing each training image as “Bag of Words”, and then we estimation the correlation between these visual words and tags.

3.1 Visual Word Vocabulary

We represent each image using the bag-of-words model. In this model, the visual feature in image can be considered as “word” in document. On top of bag-of-words model, we further take into consideration the pattern of spatial distributions of these visual features. To represent an image by a bag of visual words, we first detect the feature points and their surrounding patches in this image, then represent each feature of the point by a feature descriptor. For each feature, a feature vector is extracted from an image patch surrounding this feature point within a specific scale. The well-known feature descriptors SIFT [11] is applied to fulfill this task. Once the visual feature has been represented, a k -means algorithm is applied to cluster these extracted visual features to a fixed-size visual feature vocabulary, namely visual word vocabulary [12], [13]. The center of each cluster is regarded as a “visual word” that represents the specific local pattern shared by the patches around these key points. Thus, each word in the visual word vocabulary is the center of the corresponding cluster of visual features.

The reason lies behind to form a visual word vocabulary is as follows. The correlation between visual word and tag is more informative than the correlation between image and tag, and an image level similarity may be semantically different. Meanwhile, to weight the visual features of the batches from the test image, the image feature which is extracted directly from the image always slightly different from each other. Thus, it is hardly to estimate the relation between different visual features directly. Consequently, the image can be represented as a bag of words by replacing each image feature with the center of corresponding features cluster which is also a visual word.

3.2 Correlation Estimation

As each image can be represented by a bag of words and also contains a number of tags, the correlation between visual word and tags which will be denoted by word-to-tag correlation can be estimated by exploring their co-occurrence. The proposed estimation of word-to-tag correlation is motivated by the observation that, if a visual word f_i appears in both image I_x and I_y , then the word f_i is likely to contain the content for the tags shared by image I_x and I_y . Moreover, the more number of images that f_i and tag t_i co-occur the more correlative they are. For example, Fig.2 shows an example, in which the three images share a tag “airplane” and a visual word f_i . Thus, it is assumed that visual word f_i has some correlation with tag “airplane”. We use the following formula to estimate the correlation score between visual word f_i and tag t_i :

$$Cor(f_i, t_i) = \frac{|I(f_i) \cap I(t_i)|}{\log |I(f_i) + I(t_i) - I(f_i) \cap I(t_i)|}$$

where $I(f_i) \cap I(t_i)$ is the set of images which contain both word f_i and tag t_i . To alleviate the bias on frequent visual words and tags, we divide the correlation score by the $\log(\cdot)$ value.

To estimate the word-to-tag correlation, we index the training images by both visual words and tags, and the procedure to estimate the correlation matrix in the training process is detailed in Fig.2. When new images are added to the training dataset, it is

also easy to update the correlation matrix by just update the entries corresponding the words and tags associated with these new images.

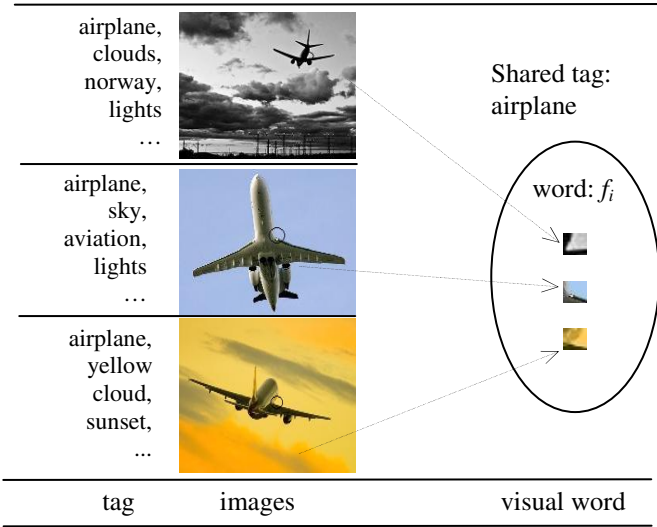


Fig. 1. Example of co-occurrence of visual word and tag

Input: visual word vocabulary $\{f_i : i=1, \dots, N_f\}$ and the tag vocabulary $\{t_j : j=1, \dots, N_t\}$

Output: The word-to-tag correlation matrix $C_{N_f \times N_t}$

1. for each tag t_j ;
2. $I(t_j)$ =set of images which indexed by t_j ;
3. for each visual word f_i ;
4. $I(f_i)$ =set of images which indexed by f_i ;
5. $Cor(f_i, t_j) = |I(f_i) \cap I(t_j)|$;
6. $Cor(f_i, t_j) = Cor(f_i, t_j) / (\log |I(t_j)| + \log |I(f_i)| - \log Cor(f_i, t_j))$;
7. $C_{ij} = Cor(f_i, t_j)$;
8. return C ;

Fig. 2. Procedure of word-to-tag correlation estimation

4 Visual Word Weighting

For the test image, some visual words may be more important than other visual words and these visual words will contribute more to tag the test image. In this section, we propose to identify the importance of visual words based on the visual word graph construct from a set of similar images, and the importance of a visual word is represented by a weight value.

4.1 Visual Word Association Graph

To weight the importance of each feature of the test image I_t , we first retrieve a set of similar images S_a of the test images from the training dataset, and then a visual word association graph $G=<F, E, W>$ is constructed from the set of similar image $S_t= S_a \cup \{I_t\}$. Finally, the visual words of the test image are ranked based on this visual word association graph G . As discussed in the previous section that each image in S_t can be represented by a bag of visual words, all these visual words and all the relationships between any two words form a graph G . The nodes of this graph represent the visual words of the set of similar images, and the weighted edges represent the relationships.

The edges of the G are generated according to the conditional co-occurrence of two visual words. For example, if two visual words f_i and f_j conditionally co-occur in at least one image, an edge $e_{ij} = (f_i, f_j) \in E$ between them is generated; otherwise no edge is generated. The edges can either be undirected or directed, and we use the undirected edge in this paper. Then the weight of edge is calculated by accumulating the sub weight of co-occurrence in each image:

$$w_{ij} = \sum_{I_x \in S_t} Con_oc_{ij}(I_x)$$

where the $Con_oc_{ij}(I_x)$ is the sub weight of conditionally co-occurrence of f_i and f_j in image I_x . Assume an image $I_x \in S_t$ contains n_i number of instances of visual word f_i and n_j number of instances of visual word f_j , then the sub weight $Con_oc_{ij}(I_x)$ in image I_x is calculated as following:

$$Con_oc_{ij}(I_x) = \sum_{p=1}^{n_i} \sum_{q=1}^{n_j} \frac{\varphi(dist(f_{ip}, f_{jq}) < d)}{M_x}$$

where $\varphi(.)$ is a delta function, and its value is 1 if the distance of f_{ip} and f_{jq} is smaller than d and 0 else. f_{ip} denotes the p^{th} instance of visual word f_i and f_{jq} denotes the q^{th} instance of visual word f_j . M_x is the total number of visual word instances in image I_x . d is an important parameter related to the constraint of co-occurrence. Since objects may have various scales in different images, the d should be properly computed to achieve scale invariance. The ideal constraint should limit the two features in the same object of interest, and d should vary with the object scale. However, it is impractical to computer an ideal constraint for all of the objects. Similar to [14] we simply calculate d with the following:

$$d = scale_i * P_d$$

where $scale_i$ is the scale of the interest point from which the instance of visual word f_i is computed. P_d is a parameter which controls the constraint of co-occurrence, and we also set P_d with 4 in this paper.

4.2 Visual Word Ranking and Weighting

Existing image tagging algorithms usually set equal weight to different features. However, for each image there are some features which are important to describe the

main objects and some other features which are just used to describe the background. By ranking the visual words, the words which related with the background can be set with a small weight, which will alleviate the risk of assigning noisy tags to the test image.

Inspired by random walk with restart (RWR) [15], we rank the visual word based on the weight propagation on the visual word association graph. Our ranking method based on the observation: given a set of similar images, there are some common objects and the background may different in each image. In other word, there are some visual words which relate to the common objects co-occur frequently in the set of similar images. Thus a visual word which co-occurs with significant visual words frequently is usually more important than the one co-occurs with a trivial feature.

With the RWR, the important visual words with high co-occurrences are mutually enhanced, and the importances of visual words are propagated through the visual word association graph. To perform the RWR, we first construct a propagation matrix $W_{|f| \times |f|}$ whose element w_{ij} is calculated in the above sub section, where $|f|$ is the total number of visual words in the image set S_i . The sum of each column of $W_{|f| \times |f|}$ is normalized as 1. Then the initial probability to choose a visual word f_i is set with $P_0(f_i)$, and the normalized initial probability vector is \vec{P}_0 . The probability vector of the n^{th} propagation is denotes by \vec{P}_n . Finally, the ranking of visual words with RWR is calculated used the following formulas:

$$P_0(f_i) = \frac{tf(f_i)}{\sum_{j=1}^{i} tf(f_j)}$$

$$\vec{P}_{n+1} = \alpha * W * \vec{P}_n + (1 - \alpha) \vec{P}_0$$

where $tf(f_i)$ is the frequency of visual word f_i in the image set S_i , and α is the restart factor which indicates the portion of importance for propagation at each step. After a number of iteration, the visual words which have high co-occurrence have very high converged values. Since the nodes with strong edges are usually extracted at the multi-occurring semantic concept in several images, the nodes with high converged values can represent the most important local visual words of this image set. Finally, the converged values are considered as the weights of visual words.

5 Image Tagging

In this section we will introduce how to tag test image by combining the visual words weight and the correlation between visual words and tags. For a test image $I_i = \langle f_1, f_2, \dots, f_m \rangle$, we first retrieve a set of visually similar image S_i and a set of tags V_i which are associated with these visually similar images. To efficiently search a set of visually similar image by content from a large scale image dataset, we divide the whole dataset into smaller subsets by the K -means clustering. Each subset is indexed by a cluster centre. For a test image, we find neighbours within fewer subsets whose centres are the closest to it.

Then, we rank the visual words of the test image based on the visual word association graph constructed from the similar image set S_t . Usually it is not true that every retrieved image is semantically similar with the test image. However, since the set of retrieved images has a certain proportion of images which are semantically similar with the test image, the ranking algorithm can enhance the important visual words that are shared by the semantically similar images. The top ranked visual words combined with their correlation to tags will contribute more to derive tags for the test image, and thus these visual words should be weighted greatly. This approach can alleviate the semantic gap problem which existing in the approaches based on image level similarity.

After the ranking process, the converged vector \vec{P}^t of visual word weights is constructed for the test image I_t . For each tag t_i in the tag set V_t , a vector \vec{C}_i of correlation scores is constructed with each element is the correlation scores of tag t_i and a visual word f_i of the test image I_t . Finally, the top tags which have the largest values calculated by the following formula are assigned to the test image.

$$PC(t_i, I_t) = \vec{P}^t \cdot \vec{C}_i$$

The pseudo code of image tagging based on the weighted correlation between visual words and tags is described in Fig.3.

Input: A test image $I_t = \langle f_1, f_2, \dots, f_m \rangle$;
Output: A tag set with n tags.

1. retrieving a set of similar images S_t ;
2. set $V_t =$ the set of tags associated with images in S_t ;
3. performing the visual words ranking process;
4. set $\vec{P}^t = \langle P_n(f_1), P_n(f_2), \dots, P_n(f_m) \rangle$;
5. for each tag t_i in V_t
6. set $\vec{C}_i = \langle Cor(f_1, t_i), Cor(f_2, t_i), \dots, Cor(f_m, t_i) \rangle$
7. $PC(t_i, I_t) = \vec{P}^t \cdot \vec{C}_i$
8. return top n tags with the greatest $PC(.)$ value;

Fig. 3. Pseude code of image tagging

6 Evaluation

In the following experiments we compare our approach (named CorTag) with the neighbor voting based tagging algorithm (named NVTag) [5] and the canonical correlation subspace based image tagging (named EGCap) [10].

6.1 Dataset and Evaluation Metrics

We have collected about 305,000 images and associated about 286,800 tags from Flickr as the training dataset. To alleviate the affection of noisy tags, we adopt filtering method by removing the tags that appear less than 10 times and more than 10000

times in the collection. We split the dataset of image into two disjoint partitions: a training set with 200,000 images and a test set with the remaining images.

We implement the Difference-of-Gaussian (DoG) [11] method to detect keypoints in the image dataset and 2,355,389 key points are found. We then these keypoints are grouped into 50000 clusters using the K-means algorithm. The key points are then assigned to the clusters with the nearest-neighbor principle. The center of each cluster is regarded as a “visual word” that represents the specific local pattern shared by the patches around the key points. After building the visual word list, an image can now be represented as a visual-word vector in which each component measures the appearance count of the corresponding visual word.

We employ two standard criteria to evaluate the image tagging performance, i.e., average precision (Av_P) and average recall (Av_R), where N_{test} is the total number of test images. With m result tags, the precision and recall are denoted by $Av_P@m$, $Av_R@m$.

$$Av_P = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} P(I_i), \quad P(I_i) = \frac{|A_i \cap B_i|}{|A_i|}$$

$$Av_R = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} R(I_i), \quad R(I_i) = \frac{|A_i \cap B_i|}{|B_i|}$$

where A_i is the set of tags assigned to the test image I_i by the automatic tagging algorithm and B_i is the set of human-produced tags for the test image I_i .

6.2 Experiment Results

In these experiments, two parameters are evaluated first. One is the size K of S_a which indicates how many visually similar images are retrieved for the test image, and the other one is the restart parameter α used in visual words ranking. After the parameters were fixed, we compare the $Av_P@m$ and $Av_R@m$ of different tagging algorithms.

In the algorithm NVTag, K is also the size of retrieved visually similar images. So, K is a common and crucial parameter of the two algorithms NVTag and CorTag. To facilitate the further evaluations, K is first decided. All the two algorithms retrieve K images for each test image, and the number of tagging result is fixed to top 15 tags for all the algorithms. The average precision and recall are shown in Fig. 4 (a) and (b) with K varied from 50 to 500. It shows that the changes of these curves of different algorithms are similar though their peaks are different. The performance of NVTag is the best when K is 300. As the CorTag weight the visual words, its curves are more smooth after that they achieve their peaks. The performance of CorTag is similar when K is equal or larger than 200. Thus, we in the following experiments, K is set to be 300 and 200 for NVTag and CorTag respectively.

The other parameter to be evaluated is the restart parameter α . As the EGCap also use RWR to determinethe most relevant words, we will record the performance of EGCap and CorTag with α varied from 0.1 to 0.9. The number of result tags m is also set to be 15. Fig.5 (a) and (b) show the average precision and recall of both algorithms. It indicates that the change trends of both figures are similar, and both

precision and recall rates reach to the lowest value when α is 0.9 with m fixed. Both of the two algorithms achieve their best performances when α is set to be 0.3, Therefore, in all of the following experiments, the parameter α is set to be 0.3 for EGCap and CorTag.

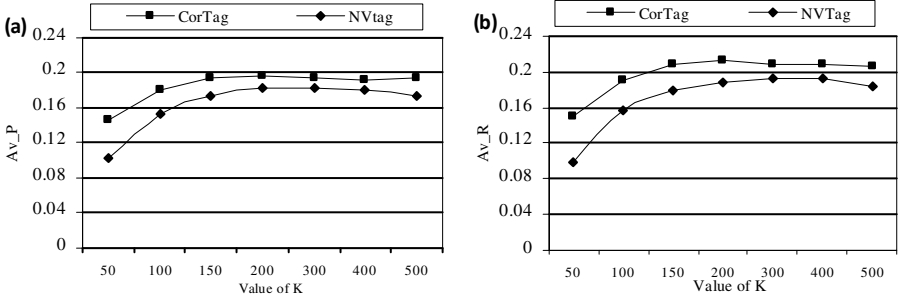


Fig. 4. (a), (b) Average precision and recall of different value of K

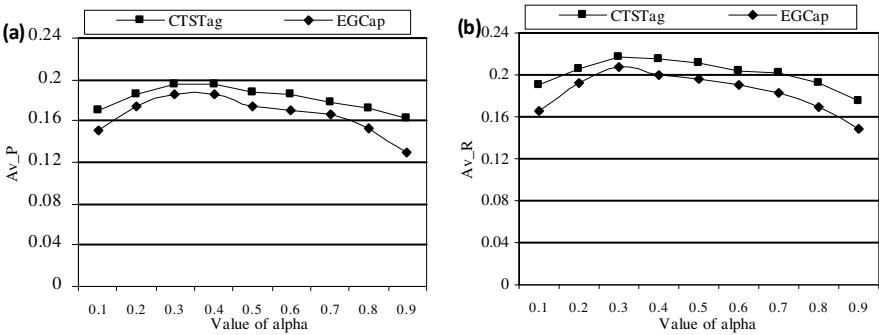


Fig. 5. (a), (b) Average precision and recall of different value of α

Based on the aforementioned parameters evaluation, we compare the $Av_P@m$ and $Av_R@m$ of different algorithms in this experiment. Fig.6 (a) and (b) show the $Av_P@m$ and $Av_R@m$ with m varied from 1 to 15. According to these figures, CorTag consistently outperforms both NVTag and EGCap. There are several reasons. The first one is that we reduce the effect of noisy images by weighting the visual words. Second, the NVTag use the frequency of a tag minus its prior frequency to train the tags, which also consider less on the visually similarity and is not scalable. Though the EGCap use the correlation between image feature and tags, its weight of image-to-tag has consideration on the test image. Our algorithm combines both the visual word weight and the correlation between visual words and tags to derive the tags. Fig.7 shows the examples of image tagging using different approaches.

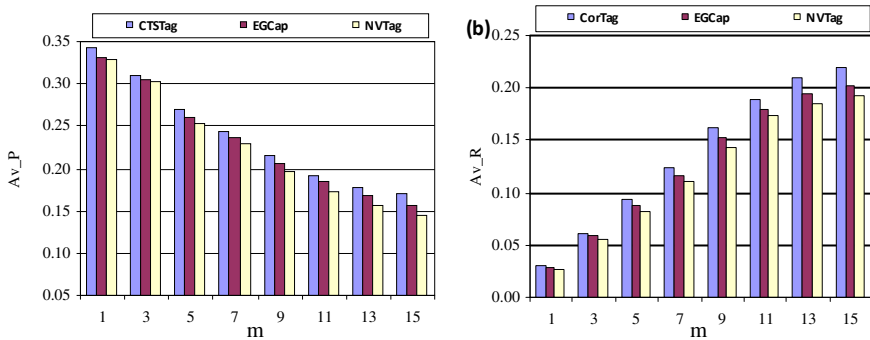


Fig. 6. (a), (b) Average top-m precision and recall of different algorithms




	Ground truth	CorTag	EGCap	NVTag
	airplane, sunset, landing, aircraft, sky, sun, lights, aeroplane, land, wheels.	<i>airplane, sunset</i> , rain, tree, <i>lights</i> , transportation, <i>sky, sun</i> , <i>aeroplane</i> , cloud.	<i>airrplane, sun</i> , jet, tree, airport, <i>sky</i> , clouds, <i>aeroplane</i> , boeing.	<i>airrplane</i> , airport, <i>sun</i> , clouds, <i>aircraft</i> , boeing, airbus, <i>sky</i> , sunset, <i>aeroplane</i> .
	White, ocean, water, horse, sand, beach, sky, animal, island.	<i>horse</i> , cloud, sea, <i>ocean</i> , pet, dog, surfing, <i>animal, water</i> .	<i>water, ocean</i> , sea, road, sun, cloud, girl, jumping, <i>sand</i> .	<i>beach, ocean</i> , sea, <i>sky</i> , sun, cloud, girl, runing, car.
	Ocean, beach, water, clouds, beautiful, travel, blue, sky, Clouds, week.	<i>Ocean, sky, beach, clouds</i> , jumping, boat, <i>water</i> , sea, nature, boy.	<i>Water</i> , ocean, <i>beach</i> , girl, <i>sky</i> , grass, jumping, island, boat, sunshine.	<i>Ocean, water, beach</i> , running, <i>blue</i> , sea, mountain, tree, model, grass.

Fig. 7. Examples of Image Tagging

7 Conclusion

In this paper, we present a automatic image tagging algorithm by exploring the weights of visual words and its correlation with tags. By taking the image as a bag of visual words, the image can be processed as a document. The correlation between visual words and tags can be estimated based on their co-occurrence, which is more informative than the correlation between images and tags. Also the visual words are ranked based on the visual word association graph constructed from a set of visually similar images, and the important visual words which appear in the multi-occurring semantic concept are set with great weights. Then, by combining the correlation and

visual words weight, the tags which have strong correlation with the greatly weighted visual words are assigned to the test image. Experimental results on real-world dataset downloaded from Flickr show the effectiveness of our tagging algorithm. In the future works, we can improve the performance of image tagging by combining the semantic meaning of tags and the performance of image retrieval can also be improved by using the correlation.

References

1. Liu, Y., Zhang, D., Lu, G., Ma, W.Y.: A survey of content-based image retrieval with high-level semantics. *Pattern Recogn.* 40(1), 262–282 (2007)
2. Naphade, M., et al.: Large-Scale Concept Ontology for Multimedia. *IEEE Multimedia* 13(3), 86–91 (2006)
3. Zhou, X., Wang, M., Zhang, Q., Zhang, J., Shi, B.: Automatic image annotation by an iterative approach: Incorporating keyword correlations and region matching. In: *Proceedings of the 6th ACM International Conference on Image and Video Retrieval*, pp. 25–32 (2007)
4. Liu, D., Wang, M., Hua, X.S., Zhang, H.J.: Tag Ranking. In: *Proceeding of the 18th ACM International Conference on World Wide Web*, pp. 351–340 (2009)
5. Li, X.R., Snoek, C.G.M., Worring, M.: Learning tag relevance by neighbor voting for social image retrieval. In: *Proceeding of 1st ACM International Conference on Multimedia Information Retrieval*, pp. 30–31 (2008)
6. Lei, W., Linjun, Y., Nenghai, Y., Hua, X.S.: Learning to tag. In: *Proceedings of the 18th ACM International Conference on World Wide Web*, pp. 20–24 (2009)
7. Barnard, K., Duygulu, P., Forsyth, D., Freitas, N., Blei, D.M., Jordan, M.I.: Matching words and pictures. *Jour. Machine Learning Research* 3(6), 1107–1135 (2003)
8. Lei, W., Steven, C.H., Jin, H.R., Jianke, Z., Nenghai, Y.: Distance Metric Learning from Uncertain Side Information with Application to Automated Photo Tagging. In: *Proceeding of 17th ACM International Conference on Multimedia*, pp. 15–24 (2009)
9. Geng, B., Yang, L., Xu, C., Hua, X.: Collaborative learning for image and video annotation. In: *Proceeding of the 1st ACM International Conference on Multimedia Information Retrieval*, pp. 443–450 (2008)
10. Bailloleul, T., Zhu, C.Z., Xu, Y.h.: Automatic Image Tagging As A Random Walk With Priors on the Canonical Correlation Subspace. In: *Proceeding of 9th ACM International Conference on Multimedia Information Retrieval*, pp. 75–82 (2008)
11. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. *IJCV* 60(2), 91–110 (2004)
12. Nister, D., Stewenius, H.: Scalable recognition with a vocabulary tree. In: *Proc. CVPR*, pp. 2161–2168 (2006)
13. Wu, Z., Ke, Q.F., Sun, J.: Bundling features for large-scale partial-duplicate web image search. In: *Proceeding of Computer Version and Pattern Recognition*, pp. 25–32 (2009)
14. Zhang, S., Tian, Q., Hua, G., Huang, Q., Li, S.: Descriptive visual words and visual phrases for image applications. In: *Proceedings of the Seventeen ACM International Conference on Multimedia*, pp. 75–84 (2009)
15. Tong, H., Faloutsos, C., Pan, J.: Random walk with restart: fast solutions and applications. *Knowl. Inf. Syst.* 14(3), 327–346 (2008)
16. Yang, K., Wang, M., Zhang, H.: Active tagging for image indexing. In: *Proceedings of the 2009 IEEE International Conference on Multimedia and Expo*, pp. 1620–1623 (2009)

17. Siersdorfer, S., San Pedro, J., Sanderson, M.: Automatic Video Tagging using Content Redundancy. In: Proceeding of the 32nd ACM International Conference on Research and Development in Information Retrieval, pp. 16–23 (2009)
18. Heesch, D., Yavlinsky, A., Ruger, S.: Nnk: networks and automated annotation for browsing large image collections from the world wide web. In: Proceedings of the 14th ACM International Conference on Multimedia, pp. 493–494 (2006)
19. Wu, F., Han, Y.H., Tian, Q., Zhuang, Y.T.: Multi-label Boosting for Image Annotation by Structural Grouping Sparsity. In: Proc. ACM Multimedia (2010)

Credibility-Oriented Ranking of Multimedia News Based on a Material-Opinion Model

Ling Xu, Qiang Ma, and Masatoshi Yoshikawa

Graduate School of Informatics, Kyoto University,
Yoshida-Honmachi, Sakyo-ku, Kyoto 606-8501, Japan

Abstract. To detect subjective intentions in a multimedia news item and help users avoid the misleading, we propose a Material-Opinion model for ranking the multimedia news with a credibility score. In the model, material is the visual description of the video news. Opinion consists of the subjective words extracted from the surrounding text (or closed captions) of that video news. After extracting materials and opinions from multimedia news items, we compare any two news items to compute their material and opinion dissimilarities, respectively. By considering the support relationship between material and opinion, we compute a credibility score for each multimedia news item. Intuitively, the credibility score of a video news item consists of material and opinion credibility scores. In the event, material credibility score is computed based on the idea that high credible material should be used in most items and they support similar opinions. On the other hand, the idea of computing opinion credibility score is that high credible opinion should be claimed in most news items by using different materials. In this paper, we also present the experiment results that validate our methods.

1 Introduction

With the rise of the Internet and the electronic signal television, video programs have broken the constraints of time and space. Multimedia news is an indispensable part of daily life. However, multimedia news that evolved from traditional textual news, is also facing the bias issue. More or less there exist subjective intentions in multimedia news. What's worse, the impression of rich visual content which is supposed to be the advantage of multimedia news, plays a negative role in encouraging audience to be conscious of perspectives. Well edited multimedia news looks very believable even it is reported by one side.

To help user avoiding the effects from the subject intentions in the multimedia news, we study on the contents analysis in views of the relative credibility in the event. Since event contents may have different meaning for different people, either individual or system can estimate the objective extent or degree in the absolute way. Our method is to evaluate multimedia news by comparing visual descriptions and textual descriptions respectively as well as considering the relationship between them. To provide user the results easy to be understood,

we rank the multimedia news in the event ordering by their relative credibility scores. In the event, news with higher credibility score would be the more well-reported item and be suggested to the user.

Our analysis targets are the multimedia news consisting of video clips and its surrounding texts (=closed captions in this paper). Since there are textual description and visual description in the multimedia news, we compute the credibility score of each multimedia news by considering both parts. We use a Material-Opinion model to compare any two of the multimedia news reporting a same event and rank all the news in the event by their credibility. In the model, material is the visual description of the video news. Opinion consists of the subjective words extracted from the surrounding text of that video news.

Because the visual description and textual description always come to the user synchronously, we think there is internal relationship between material and opinion. Textual description in multimedia news has two common functions to the user. 1) Declarative function: emphasis or complementarity of the visual descriptions. 2) Argumentative function: conclusion or extension of the visual descriptions. In both functions, textual description is presented closely connected with visual description. Actually, the former depends on the latter. So does the opinion on the material. Thus we think that for each multimedia news, there is supporting relationship between its material and opinion, which would help the credibility evaluation later.

By considering the supporting relationship between material and opinion, we compute a credibility score for each multimedia news item. Intuitively, the credibility score of a video news item consists of material and opinion credibility scores. Material credibility score is computed based on the idea that high credible material should be used in most items and they support similar opinions. On the other hand, the idea to compute opinion credibility score is high credible opinion should be claimed in many news items by using different materials.

As a simple realization of using Material-Opinion model evaluating the credibility, we use the stakeholder model representing the contents for comparing materials and opinions respectively. Stakeholders are the important entities in the event, whose descriptions are supposed to be the most valuable parts for the comparison. Currently we focus on the person stakeholder only. We compute the material dissimilarity by using their segment information consisting of the stakeholder appearance and the duration. Opinion dissimilarity is computed by comparing the sentiment polarities of the stakeholder-related textual descriptions in the news. We also propose the method of computing a credibility score of each multimedia news to determine its relative credibility in the event.

2 Related Work

There are lots of research on opinion mining and sentiment summarization of evaluative text [8, 4]. Semantic analysis has been widely used for analysing user reviews on the products or detecting the opinion-rich resources. There is subtle difference in opinion mining of news articles that the concern is a particular event instead of a particular product. For assisting reading news articles.

NewsInEssence [2] and Columbia’s Newsblaster [7] summarize news articles from multiple sources into one virtual news article. Ma et al. proposed a comparative query generation method for cross-media search between text news and video news [6]. They considered news of different media to be complementary and their method is to search for complementary news.

These researches aimed at complementing differences between news articles. Some others aimed at making users conscious of differences between news articles. Nadamoto et al. focused on differences between news articles about the same topic in different countries, and developed a bilingual comparative web browser (B-CWB) [5]. Users can compare news articles in two countries and find their differences of the content. Park et al. think that there are some bias in news articles, and developed the system NewsCube to classify the news articles by their aspects and present them to users [9].

Above researches summarize viewpoints in text, and explore characteristic or offer essential functions to the audience. They are concerning more about how to collect and classify the different views, whose study is limited with the articles (text). Our analysis takes use of the different media types of contents in the multimedia news. We use the composite consideration of textual and visual descriptions to compare two news in a more detail way, and rank the multimedia news in views of their credibility.

There are previous studies focused on the analysis of credibility. Y. Yamamoto et al. proposed a method of judging the credibility of uncertain facts on the Internet by comparing the uncertain fact with other facts that is replaced with the subject or object of the clause [10]. ImageAlert [12] study on the credibility of text-image pairs, and explore the typical image of a product name by using supportive relations between text and image. Our proposal has the similar idea of considering the supporting relations and the causality. Our methods focus on the contents analysis of multimedia news items where item is often consisting of more than one fact and the link of material and opinion (video and text in the news) is less visible than the link of causality that can be detected by particular verbs in the descriptions. In our methods, contents of the multimedia news items reporting a same event can be presented as pairs of material and opinion. A news item reported with credible material and credible opinion can be regarded as the credible news item itself. Rather than focusing on the mainstream material or opinion on the Internet (what is most published or admitted), we also consider the diversity of the materials supporting the similar opinions and the diversity of the opinions coming with the similar materials in the event.

3 Framework of Material-Opinion Model

To evaluating the relative credibility of multimedia news, we propose a Material-Opinion model for the comparison. In the model, material is the visual description of each video that filmed by the camera as a reflection of the real world, which is supposed to be believable despite the aggrandizement or forgery in films. Opinion is extracted from the textual description of each news, consisting of the sentiment words put forward by the news agency. For a multimedia news c_k ,

$$c_k = (M_k, O_k) \quad (1)$$

$$M_k = \{seg_p | seg_p \in V_k\} \quad (2)$$

$$O_k = \{word_q | word_q \in S_k\} \quad (3)$$

where V_k is the visual description of c_k ; S_k are the sentences in the textual descriptions. p is the number of segments. M_k and O_k are the material and opinion of c_k . seg_p is a segment in video of c_k .

We think that material in news items play as objective description in the event and opinions are expressed as the subjective description from the agencies. Material and opinion in the same multimedia news interact with each other when reporting a news. For example, in the event “summit conference between Obama and Sarkozy on July 7, 2009”, one news item c_k offers the video of “Obama and Sarkozy entering the ceremony”, “Obama and Sarkozy talking with each other in the hall”, “Obama speaking of the North Korea issue seriously”. The corresponding texts are “...They entered the ceremony together...”, “...they 110% agree with each other on the North Korea issue...”, “...I strongly condemn their reckless action...(by Obama)”¹. Here, V_k of two presidents offer the visual descriptions of activities that really occurred at that time, when the text supplies the sightless parts such as “Obama and Sarkozy are close to each other” and “Obama expressed strong feelings and emotions to North Korea”. In this multimedia news, the opinion(s) supported by materials(s) sounds credible when the visual description V_k and the textual description S_k appear together to the user in the multimedia news.

However, with different perspectives, a news agency is free to determine the priority of the materials and raise their own points of view. In the previous event, there is another news c'_k expressing the conclusion of “bad relationship” because of “absence of dinner” and “appearing disengaged on the conference”. Here, either “good relationship” or “bad relationship” is hard to be refuted.

To reveal such difference between multimedia news, we propose the Material-Opinion model for comparing the objective descriptions and the subjective descriptions respectively in the event. Because of the synchronism between video and text in multimedia news, we think the opinions are expressed closely connected with the materials. Considering that the textual descriptions always work as the complementary or the conclusion of the visual descriptions, we think there is supporting relationship between material and opinion.

Section 4 introduce the methods of computing credibility scores by using a stakeholder model representing the contents for the comparison. When comparing two news items by using the Material-Opinion model, we extract the segments for materials and descriptive words for opinions from the stakeholder-related descriptions in the multimedia news. We use the stakeholder model proposed in our previous work [11] to represent the textual and visual descriptions for the further analysis. In most of the news items, there is at least one entity

¹ Texts in English are translated from the Japanese news.

in the contents, as a person, an organization, or a location. This feature makes it possible to represent the contents by the descriptions on important entities as the most valuable parts in the contents. We focus on the stakeholder-related descriptions in the multimedia news for the comparison in material and in opinion. Here, stakeholders are the main participants (persons, organizations, etc.) in the news event. In one event, stakeholders are the important entities who appear (or be mentioned) in video (or text) frequently in most of the news items. After extracting the stakeholders in the event by computing the exposure degrees, we compute material dissimilarities by using the segments with its stakeholder information, and compute the opinion dissimilarities by extracting the sentiment words in the stakeholder mentioned sentences.

4 Credibility-Oriented Ranking of Multimedia News by Using Stakeholder Model Representing the Contents

With an overview picture of evaluating a multimedia news, we evaluate the credibility by considering both material and opinion. Obviously, a credible news (than the others in the event) would be reported with credible material and credible opinion.

As a implement of using Material-Opinion model for the comparison, we use stakeholder model representing the contents for comparing the materials and opinions in the event. We first extract the stakeholder(s) in the event by detecting entities and clustering them by their exposure degrees in both text and video [11]. Currently we focuses on the persons stakeholder only. Second we compute the material dissimilarity and opinion dissimilarity between any two news items. Materials are compared by using the segments labelled with stakeholder appearance. Opinions are compared by using positive, negative and objective polarities of each stakeholder extracted from the text. Then we cluster the items in the same event with similar materials and similar opinions. According to the clustering result, four factors to trace and analyse are the material majority, material diversity, opinion majority and opinion diversity. We propose a notion of credibility score Cre of each multimedia news by considering conditions as follows. For each multimedia news c_k ,

1. Is there any (or lots of) other items agreed with c_k 's opinions in the event.
2. Is there any (or lots of) other items using similar material in the event.
3. Is there any (or lots of) other videos supporting similar opinion of c_k in the event.
4. Is there any (or lots of) other news using similar material but raising different views against c_k in the event.

Here, the first and the second indicates opinion majority and material majority; the third and the fourth indicates the diversity of all in opinion and the diversity of all in material. Multimedia news with diverse supported opinions and significant materials (supporting agminate opinions) are supposed to be more credible than others.

The repeating counts rely on the frame amount of each segment with settled time interval (plus 1 per 500ms in or current implement). The character string of each video is the ordered chain of the repeated Sid_p of the segments. Our comparison of two videos is actually the comparison of the two transferred character strings.

- The material dissimilarity $Dis_m(c_1, c_2)$ of items c_1 and c_2 is calculated as follows.

$$Dis_m(c_1, c_2) = \frac{|LCS(c_1, c_2)|}{ll(c_1, c_2)} \quad (4)$$

where $LCS(c_1, c_2)$ is the Longest Common Subsequence [1] of multimedia news c_1 and c_2 . ll is the longest length of the character string of the two videos. Respectively, $0 \leq Dis_m(c_1, c_2) \leq 1$.

4.2 Opinion Dissimilarity

Similar with how we compute the material dissimilarity, opinion dissimilarity is calculated by using the stakeholder-related descriptions in the contents. To compute the opinion dissimilarity, we use a method of extracting descriptive polarities from stakeholder-related descriptions and make feature vectors by using the sentiment polarities of stakeholder-related descriptions in the event.

First we use CaboCha [3] to build a grammar tree of each sentence in the text. If any of the nodes in the tree contains the name of a stakeholder s , the sub-tree started from this node's parent would be the s -related phrase. For each s -related phrase in the news item, we use a sentiment dictionary evaluating each word with positive, negative and objective scores [13]. We multiply the score of nodes in main clause (node where s exists, sibling nodes and parent node) with coefficient = 1 and multiply the polarity of rest with coefficient = 0.5. After adding the products up, for each multimedia news c , we can extract the polarity scores as follows.

$$pls = (pol(s_1, P), pol(s_1, N), pol(s_1, O), \dots, pol(s_n, P), pol(s_n, N), pol(s_n, O)) \quad (5)$$

where $pol(s, P)$, $pol(s, N)$, $pol(s, O)$ are the positive, negative and objective sum of the textual description on the stakeholder in a multimedia news.

Because we compute the polarity scores of each stakeholder in item by adding up the scores of the descriptive semantic words in the phrases, standardization is required for the large scores caused by the long texts.

Table 1 shows an example of polarity scores in the event. There are two stakeholders s_1 (Obama) and s_2 (Sarkozy) in the event. Rows of $pls(c_2)$ and $pls(c_3)$ show that there is no opinion in the s_2 -related textual descriptions of item c_2 and c_3 . Even the four multimedia news are reporting the same event, c_2 and c_3 offer little descriptions on s_2 (The agency concerned more about Obama than Sarkozy). We compute the opinion dissimilarity Dis_o of item c_1 and c_2 as follows.

Table 1. Positive, Negative and Objective Polarity Scores of Stakeholders in the Event

	$pol(s_1, P)$	$pol(s_1, N)$	$pol(s_1, O)$	$pol(s_2, P)$	$pol(s_2, N)$	$pol(s_2, O)$
$pls(c_1) :$	0.2	0.27	0.53	0.25	0.36	0.4
$pls(c_2) :$	0.24	0.29	0.46	0	0	0
$pls(c_3) :$	0.26	0.32	0.42	0	0	0
$pls(c_4) :$	0.3	0.22	0.48	0.47	0.08	0.45

$$Dis_o(c_1, c_2) = \sqrt{1 - \frac{(pls(c_1) \cdot pls(c_2))^2}{|pls(c_1)|^2 |pls(c_2)|^2}} \tag{6}$$

where $pls(c_1)$ and $pls(c_2)$ are polarity value vectors of news items c_1 and c_2 respectively.

4.3 Clustering by Material Dissimilarities

We separate the items into groups by the method of aggregative hierarchical clustering by using their material dissimilarities. Items in one group are sharing similar materials. Material majority Maj^M and material diversity Div^M of multimedia news item c is as follows.

$$Maj^M(c) = \frac{|Group_c^M|}{\sum |Group^M|} \tag{7}$$

$$Div^M(c) = \frac{\sum_{c' \in Group_c^M, c' \neq c} Dis_O(c, c')}{|Group_c^M|} \tag{8}$$

where, $Group_c^M$ is the group that item c belongs to by the clustering results using material dissimilarities. For each item c , $Maj^M(c)$ is the account ratio of c 's group in all according to the clustering result. $Div^M(c)$ is the average opinion dissimilarity between c and the other items in the same group $Group_c^M$.

4.4 Clustering by Opinion Dissimilarities

We separate the items into groups by the method of aggregative hierarchical clustering by using their opinion dissimilarities. Items in one group are sharing similar opinions. Opinion majority Maj^O and opinion diversity Div^O of multimedia news item c is as follows.

$$Maj^O(c) = \frac{|Group_c^O|}{\sum |Group^O|} \tag{9}$$

$$Div^O(c) = \frac{\sum_{c' \in Group_c^O, c' \neq c} Dis_M(c, c')}{|Group_c^O|} \tag{10}$$

where, $Group_c^O$ is the group that item c belongs to by the clustering results using opinion dissimilarities. For each item c , $Maj^O(c)$ is the account ratio of c 's group in all according to the clustering result. $Div^O(c)$ is the average material dissimilarity between c and the other items in the same group $Group_c^O$.

4.5 Computing Credibility Score

After computing the dissimilarities and clustering the multimedia news in the event, we compute the credibility score of each news by using its material credibility score and its opinion credibility score. Material credibility score Cre^M is computed based on the idea that high credible material should be used in most items and they support similar opinions.

$$Cre^M(c) = Maj^M(c) \cdot (1 - Div^M(c) + \epsilon) \quad (11)$$

In the case of $Div = 0$, we use an arbitrarily small positive quantity ϵ in the computation.

Opinion credibility score Cre^O is computed based on the idea that high opinion should be claimed in may news items by using different materials.

$$Cre^O(c) = Maj^O(c) \cdot (Div^O(c) + \epsilon) \quad (12)$$

Therefore, we compute the credibility score as follows.

$$Cre(c) = Cre^M(c) + Cre^O(c) \quad (13)$$

Because of the relative comparison, we use credibility scores as the relative measures of ranking the credible news in the event. The absolute value of credibility score is meaningless in the results.

5 Experiments

To validate the methods, we carried out three experiments:

1. Experiment of comparing materials.
2. Experiment of and comparing opinions.
3. Experiment of ranking multimedia news by their credibility scores.

Analysis targets are the news videos and their surrounding texts extracted from the video news web sites² in Japan. The surrounding texts can be seen the same as the closed captions in the videos.

The system collects the multimedia news published on the Internet every day. Videos (file in flv or wma) and the surrounding texts (can be seen as the closed captions) are downloaded for the analysis. Multimedia news who are reported within 24 hours and sharing similar titles in text are clustered into a same event.

In the experiments, we selected 6 events consisting of 35 news events published from January 11, 2011 to January 20, 2011 when there are at least one person stakeholder in the event. In addition, the selected events are consisting of at least 3 news items from different broadcasters for the comparison. Videos of the test data add up to more than 53 minutes and the average duration of the multimedia news is about 92 seconds. We determined several pre-set parameters for the appropriate performance of the clustering of multimedia news. We set threshold of material $T_{material} = 0.4$ and threshold of opinion $T_{opinion} = 0.4$ after some preliminary experiments.

² NTV: www.news24.jp; FNN: www.fnn-news.com; TBS: news.tbs.co.jp; ANN: www.tv-asahi.co.jp/ann

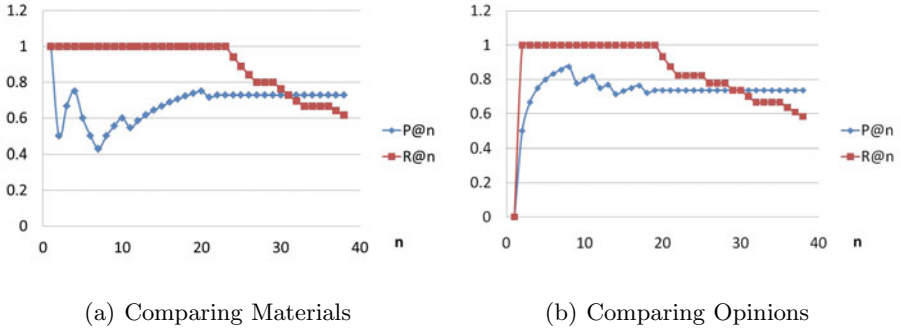


Fig. 2. Precision and Recall Ratios of Topmost Results

5.1 Experiment of Comparing Materials

To evaluate the method of comparing the materials of the multimedia news, we compared the system results with user rating of material dissimilarities between any two of the news items. After watching the videos of each individual news items, user rated dissimilarity between any pairs of videos on five criteria: 0.2, 0.4, 0.6, 0.8, 1.0, where 0.2 indicates *there is litter difference* and 1.0 indicates *there is big difference*.

We set the threshold of material dissimilarity $\vartheta_m = 0.5$. If $Dis_M > \vartheta_m$, there is difference according to the system result. Fig. 2(a) shows the precision at n $P@n$ and the recall at n $R@n$ where n is the count of the topmost results (ordered by material dissimilarities). We also count the sample correlation coefficient $r_M = 0.02$. Reasons of the low sample correlation coefficient value are: 1) The manual evaluation considers the videos as all. For manual evaluation, time duration difference indicating increase or decrease of contents also performs high dissimilarity between two materials. In our method, the factor of time duration is isolated with the method of Edit Distance. Time duration is not statistically irrelevant directly with material dissimilarity. 2) When two videos are common in part but far different in another, it is hard to evaluate how dissimilar two videos are.

5.2 Experiment of Comparing Opinions

To evaluate the method of comparing the opinions of the multimedia news, we compared the system results with user rating of opinion dissimilarities between any two of the news items. After reading surrounding text of each individual news items, user rated dissimilarity between any pairs of closed captions on five criteria: 0.2, 0.4, 0.6, 0.8, 1.0, where 0.2 indicates *there is litter difference* and 1.0 indicates *there is big difference*.

We set the threshold of opinion dissimilarity $\vartheta_o = 0.5$. If $Dis_O > \vartheta_o$, there is difference according to the system result. We also count the sample correlation coefficient $r_O = 0.24$. Fig. 2(b) shows the precision at n $P@n$ and the recall

at $n R@n$ where n is the count of the topmost results (ordered by opinion dissimilarities).

According to the sample correlation coefficient and the precision ratios, opinion comparison in multimedia news performs well with high precision.

5.3 Experiment of Ranking Multimedia News by Credibility Scores

To evaluate the method of evaluating the multimedia news by credibility scores, we compared the system results with user rating of material and opinion dissimilarities between news items. After reading closed captions and watching the videos of each individual news items in the event, user rated the manual credibility score rated on five criteria 1, 2, 3, 4 and 5 where 5 indicates a very credible item and 1 indicate it is reported with bias evaluated by user. We count the sample correlation coefficient $r_C = 0.19$, which indicates that the credibility-oriented ranking of the multimedia news performs valid with the user understanding.

6 Conclusion

To the aim of detecting the bias and avoiding the misleading caused by the news reports, we proposed a Material-Opinion model to compare multimedia news and rank them by their relative credibility scores. In the model, visual descriptions in the multimedia news are supposed to be the materials that supporting the opinions raised in the text. In the implement of using Material-Opinion model for the analysis, we compare the stakeholder-labelled segments of video to compute the material dissimilarities; and compare the sentiment polarities of the stakeholder-related phrases in the closed captions to compute the opinion dissimilarities. By using the dissimilarities, we cluster the multimedia news and compute the credibility score for ranking. Multimedia news item that reports opinion similar to most of the others and such that opinion is supported by diverse visual descriptions in material will be signed with high credibility scores. We also carried out some experiments validating our methods. Our future work is to modify the Material-Opinion model for comparing the multimedia news.

References

1. Jiang, T., Li, M.: On the Approximation of Shortest Common Supersequences and Longest Common Subsequences. In: Shamir, E., Abiteboul, S. (eds.) ICALP 1994. LNCS, vol. 820, pp. 191–202. Springer, Heidelberg (1994)
2. Mckeown, K.R., Barzilay, R., Evans, D., Hatzivassiloglou, V., Klavans, J.L., Nenkova, A., Sable, C., Schiffman, B., Sigelman, S.: Tracking and Summarizing News on a Daily Basis with Columbia’s Newsblaster. In: Proc. of Human Language Technology Conference (2002)
3. Taku, K.: CaboCha: Yet Another Japanese Dependency Structure Analyzer (online). Technical report, Nara Institute of Science and Technology (2004)
4. Hu, M., Liu, B.: Mining and Summarizing Customer Reviews. In: Proc. of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 168–177 (2004)

5. Nadamoto, A., Ma, Q., Tanaka, K.: B-CWB: Bilingual Comparative Web Browser Based on Content-Synchronization and Viewpoint Retrieval. *World Wide Web Journal* 8(3), 347–367 (2005)
6. Ma, Q., Nadamoto, A., Tanaka, K.: Complementary Information Retrieval for Cross-Media News Content. *Information System* 31(7), 659–678 (2005)
7. Radev, D., Otterbacher, J., Winkel, A., Blair-Goldensohn, S.: NewsInEssence: Summarizing Online News Topics. *Communication of the ACM* 48, 95–98 (2005)
8. Pang, B., Lee, L.: Opinion Mining and Sentiment. *Foundations and Trends in Information Retrieval* 2(1-2), 1–135 (2008)
9. Park, S., Kang, S., Chung, S., Song, J.: NewsCube: Delivering Multiple Aspects of News to Mitigate Media Bias. In: *Proc. of 27th International Conference on Human Factors in Computing Systems (CHI 2009)*, pp. 443–452 (2009)
10. Yamamoto, Y., Tanaka, K.: Finding Comparative Facts and Aspects for Judging the Credibility of Uncertain Facts. In: Vossen, G., Long, D.D.E., Yu, J.X. (eds.) *WISE 2009*. LNCS, vol. 5802, pp. 291–305. Springer, Heidelberg (2009)
11. Xu, L., Ma, Q., Yoshikawa, M.: A Cross-media Method of Stakeholder Extraction for News Contents Analysis. In: Chen, L., Tang, C., Yang, J., Gao, Y. (eds.) *WAIM 2010*. LNCS, vol. 6184, pp. 232–237. Springer, Heidelberg (2010)
12. Yamamoto, Y., Tanaka, K.: ImageAlert: Credibility Analysis of Text-Image Pairs on the Web. In: *Proc. of 26th ACM Symposium on Applied Computing, SAC 2011* (2011)
13. Ishida, S.: Comparative Analysis of News Agencies by Extracting Entity Descriptions and Its Application, Thesis Paper of Master Course (2011)

Actions in Still Web Images: Visualization, Detection and Retrieval

Piji Li, Jun Ma, and Shuai Gao

School of Computer Science & Technology, Shandong University,
Jinan, 250101, China
peegeelee@gmail.com, majun@sdu.edu.cn, gao_shuai@mail.sdu.edu.cn
<http://ir.sdu.edu.cn/>

Abstract. We describe a framework for human action retrieval in still web images by verb queries, for instance “phoning”. Firstly, we build a group of visual discriminative instances for each action class, called “Exemplarlets”. Thereafter we employ Multiple Kernel Learning (MKL) to learn an optimal combination of histogram intersection kernels, each of which captures a state-of-the-art feature channel. Our features include the distribution of edges, dense visual words and feature descriptors at different levels of spatial pyramid. For a new image we can detect the hot-region using a sliding-window detector learnt via MKL. The hot-region can imply latent actions in the image. After the hot-region has been detected, we build a inverted index in the visual search path, which we called Visual Inverted Index (VII). Finally, fusing the visual search path and the text search path, we can get the accurate results either relevant to text or to visual information. We show both the detection and retrieval results on our newly collected dataset of six actions as well as demonstrate improved performance over existing methods.

Keywords: Web image retrieval, action detection, multiple kernel learning, visual inverted index, exemplarlet.

1 Introduction and Motivation

Human actions represent essential content and elements of many web images, so searching actions from still web images is useful and important. However, with the advent of the digital camera and the popularity of internet photo sharing sites, the size of digital image collection is increasing rapidly. Therefore, finding the interesting human action images from this big scale image collection is a significant and challenging work.

Human action retrieval in still images is an example of “semantic gap” problems. For instance, submit several actions (e.g. “phoning” and “playing guitar”) as queries to text-based image search engines such as Google Image Search¹, Bing Image Search² or Flickr³, we would get some result ranking lists. What

¹ <http://images.google.com/>

² <http://images.bing.com/>

³ <http://www.flickr.com/>

do you see? For most of us, these retrieval results are not as satisfactory as we expect. The reason for this phenomenon is that the visual irrelevant images contains the textual relevant information. Therefore we face a problem: how to mining the satisfactory results relevant to both text and visual information when submit an action query to a text-based image search engine?

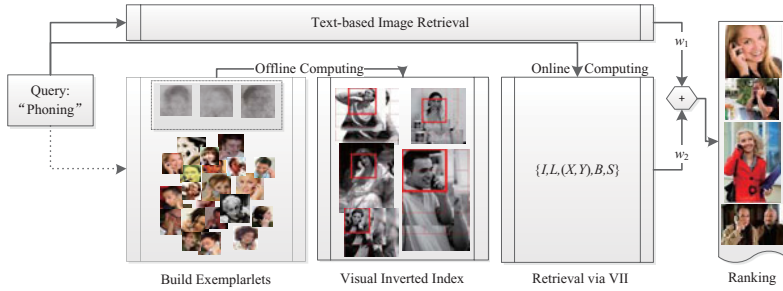


Fig. 1. Illustration of our proposed framework for human action retrieval in still web images via text-based search

To address the aforementioned problem, we propose a novel framework for human action retrieval in still web images. The high-level idea of our approach can be seen from Figure 1. Illuminated by the idea of web image retrieval re-ranking and action classification in still images, we divide our framework into two components: online computing component and offline computing component. When queries are submitted to the search system, the online computing component is called to retrieval goal images via searching textual and **Visual Inverted Index** (VII) respectively. Thereafter, results are combined using a weighted procedure by parameter w . Finally, the results are listed in the order calculated by the ranking function.

The offline computing component is implemented for the purpose of building the visual inverted index for action queries. Firstly, we build a group of **Exemplarlets** for each action query. Intuitively speaking, each exemplarlet is a minimum sub-image (bounding box) to picture the action in global level. Thereafter we employ Multiple Kernel Learning (MKL) [20] to learn some detectors to detect **Hot Regions** in images and then the visual inverted index will be built. We will interpret our action retrieval framework specifically in Section 3. Specific experiments are designed and implemented for validating the performance of action detection and retrieval on our newly collected dataset of six actions, and significant improvement is reported in terms of accuracy and precision.

The rest of this paper is organized as follows. It starts with a brief review of related works in Section 2 while the novel model for action retrieval is proposed in Section 3. The experimental results and discussions are provided in Section 4. Concluding remarks and future work directions are listed in Section 5.

2 Related Work

In this section we firstly discuss the state-of-the-art re-ranking method in web image retrieval, then we will review some research works on action detection and classification.

2.1 Web Image Retrieval Re-ranking

Recently, some approaches are proposed for re-organizing text based search results by incorporating visual information [3,13,16,17].

M. Chi *et al.* [3] take a subset of images and segment all images into blobs. When clustered, densities of blob clusters become directly proportional to the relevancy of images in that cluster to the query. Using this idea, remaining images are inserted to appropriate clusters and images are re-ranked. [13] proposes the Inter-cluster rank and Intra-cluster rank to reorder the clusters and the images in some clusters respectively. [16] introduces a lightweight re-ranking method that compares each result to an external, contrastive class. Using a diversification function ensures that different aspects of a query are presented to the user. [17] presents an active re-ranking method to target the user’s intention, and also proposes a local-global discriminative dimension reduction algorithm by transferring the local geometry and the discriminative information from the labeled images to the whole (global) image database, to localize the user’s intention in the visual feature space. Intuitively speaking, the online computing component is adapted from the re-ranking methodology, i.e., re-organize text based search results by incorporating visual information.

2.2 Action Detection and Classification

The visual inverted index is the most important data in online computing, and it will be built via action detection progress in offline computing. Most of the work in this field focuses on recognizing actions from videos [11,14,15] using motion cues, and a significant amount of progress has been made in the past few years. Action recognition from still images, on the other hand, has not been widely studied with the exception of few related papers focused on specific domains, such as sports actions [8,9,21,23] or, more recently, people playing musical instruments [22]. Learning from still images to recognize actions in video was investigated in [10]. The proposed methods [8,9,21,23] have mainly used the body pose as a cue for action recognition. While promising results have been demonstrated on sports actions [8,9,21,23], typical action images such as illustrated in Figure 1 often contain heavy occlusions and significant changes in camera viewpoint, which are serious challenges for current body pose estimation methods. Inspired by a more general methodology [5], to deal with various types of actions in still images, we avoid explicit reasoning about body poses and investigate more general classification methods.

3 Framework

Follow the illustration of our approach in Figure 1, firstly, we will elaborate on how to collect exemplarlets for each action query; thereafter, we employ multiple kernel learning to refine discriminative exemplarlets and learn the action detectors; then we will build the visual inverted index for each action query; finally, we utilize a ranking function to re-organize the result list.

3.1 Defining Exemplarlet

An exemplarlet A is defined as the minimum sub-image (bounding box) which contains enough visual information for us to identify the action. For instance, the red bounding boxes in Figure 1 are examples of exemplarlets for “phoning”. We just focus on the smallest region here, for the reason that small exemplarlets could speed up the learning and detection procedure. Moreover, exemplarlet contains enough visual information and will not suppress the performance seriously.

Let Y be the action label of exemplarlets. During implementation, Y is the *id* set of all the action classes in the database. We denote A as $A = \{A, B, Y\}$, where A is the visual appearance and B describe the size of exemplarlets. B is denoted by $\{b_0, b_1, \dots, b_{K-1}\}$, where K is the number of exemplarlets. The configuration of the k -th exemplarlet b_k is represented as $b_k = (h_k, w_k)$, where (h_k, w_k) is the height and weight value of the k -th exemplarlet.

The exemplarlets selected and segmented from several web image collections (Google, Bing, Flickr) manually are called “Seed Exemplarlets”, such as illustrated in Figure 2. To name a few, exemplarlets in first row of Figure 2 are the visualization of action query “phoning”. Each action class is assigned about 60 seed exemplarlets.

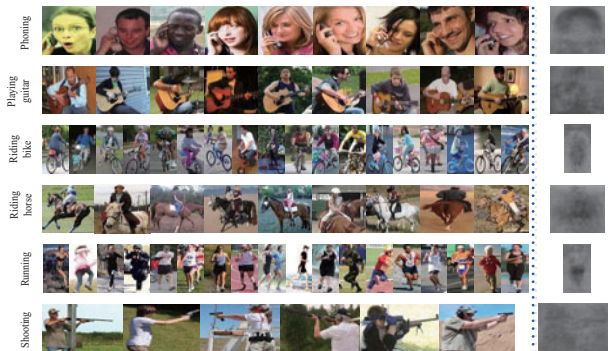


Fig. 2. Examples of seed exemplarlets we collected for actions of “phoning” “playing guitar” “riding bike” “riding horse” “running” and “shooting”

3.2 Refining Discriminative Exemplarlets

To recognize subtly different scenes, we would like to select a rich set of exemplarlets that are not only highly characteristic of the action class, but also highly discriminative compared to other classes. We propose a multiple kernel learning algorithm for discovering discriminative exemplarlets. Exemplarlet Λ is discriminative for class c means that Λ has larger score value for class c than the other classes. In the rest of this section, we first describe how to compute the score of an exemplarlet and then describe the method to select discriminative exemplarlets. Firstly, we make an important simplifying assumption that we have a detector D obtained for each action class c . Then the score v of an exemplarlet for class c is expressed as:

$$v = \sum_{a \in \mathcal{Y}} D(\Lambda, Y; \Theta) \cdot \mathbf{1}_a(Y) \quad (1)$$

where $\mathbf{1}_a(Y)$ is an indicator that takes the value 1 if $Y = a$, and 0 otherwise. We take the detectors learning procedure (we will elaborate on it in the next section) as binary classification problem, then $\mathcal{Y} = \{-1, +1\}$. We assume $D(\Lambda, Y; \Theta)$ takes the following form:

$$\begin{aligned} D(\Lambda, Y; \Theta) &= D(A, B, Y; \Theta) \\ &= \Theta^T \Psi(A, B, Y) \\ &= \alpha^T \phi(A, Y) + \beta^T \varphi(B, Y) \end{aligned} \quad (2)$$

where D is parameterized by Θ , $\Psi(A, B, Y)$ is a feature vector observed from the exemplarlet Λ . $\Theta^T \Psi(A, B, Y)$ can be defined as the sum of each potential function $\alpha^T \phi(A) + \beta^T \varphi(B)$, where A is the visual appearance of exemplarlet Λ and B is the size of Λ . The detector parameter Θ is simply the concatenation of the parameters in all the potential functions, i.e. $\Theta = \{\alpha, \beta\}$. Given the score value for each exemplarlet of class c , we could select the discriminative exemplarlets with the largest v .

3.3 Learning Detectors via MKL

Given a set of N training examples $\{(\Lambda^{(n)}, Y^{(n)})\}_{n=1}^N$, we have to learn a discriminative and efficient detector function $D : \mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{R}$ over an exemplarlet Λ and its class label Y , where \mathcal{X} denote the input space of exemplarlets and $\mathcal{Y} = \{-1, +1\}$ is the set of class labels. D is parameterized by Θ and during testing and detecting, we can predict the action type Y^* of an input exemplarlet as:

$$Y^* = \arg \max_{Y \in \mathcal{Y}} D(\Lambda, Y; \Theta) \quad (3)$$

The detector function $D(\Lambda, Y; \Theta)$ is learnt, along with the optimal combination of features and spatial pyramid levels, by using the Multiple Kernel Learning (MKL) technique proposed in [19,20]. In our work, we just focus on the

visual appearance A of exemplarlets, and ignore the impact of size B . Therefore, $D(A, Y; \Theta)$ in Eq. (2) can be rewritten as:

$$\begin{aligned} D(A, Y; \Theta) &= \sum_{i=1}^N y_i \Theta_i K(A, A^i) \\ &= \sum_{i=1}^N y_i \alpha_i K(A, A^i) \end{aligned} \quad (4)$$

where A^i , $i = 1, 2, \dots, N$ denote the feature descriptors of N training exemplarlets, selected as representative by the SVM, $y_i \in \mathcal{Y}$ is their class labels, and K is a positive definite kernel, obtained as a liner combination of histogram kernels by η :

$$\begin{aligned} K(A, A^i) &= \sum_{k=1}^{\#A} \eta_k k(A_k, A_k^i) \\ \sum_{k=1}^{\#A} \eta_k &= 1 \end{aligned} \quad (5)$$

where $\#A$ is the number of features to describe the appearance of exemplarlets, in this paper $\#A = 2$ for two kinds of features (Pyramid HOG and Pyramid SIFT). MKL learns both the coefficient α_i and the histogram combination weight $\eta_k \in [0, 1]$.

Many research work [12, 22] have observed that the histogram intersection kernel performs better than the other kernels. Therefore, we employ multiple histogram intersection kernel in this paper:

$$k(A, A^i) = \sum_{i=1}^N \min(A, A^i) \quad (6)$$

3.4 Building Visual Inverted Index

Illumined by the idea of effectively and efficiently Inverted Index in textual information retrieval, we establish a mapping between action queries (e.g. “phoning” “riding bike”) and image visual information. We named this mapping “Visual Inverted Index” and “VII” for short. The online computing component of our framework (Figure 2) could finished in time via searching the VII only, and then return the intermediate results list for ranking. After learnt the action detectors D , VII can be builded easily in the detection procedure.

The goal of the detection procedure is to find the “hot region”. For an new input image I , we zoom it into $|L|$ scales at first, i.e., build the image pyramid. Then we will run the detection algorithm on candidate regions via a sliding-window at the different pyramid level L . A candidate region R , is a sub-image (window) of the input image I . We can express R as:

$$(I, A, B, C, l, v_c) \quad (7)$$

where I is the parent image of R , A is the visual appearance, B is the size of R (*height * width*). C is the location of R on image I and we record the left-top

corner of the window. $l \in L$ is the pyramid level of image. v_c is the likelihood (score) of R for action class c , and can be computed via Eq. (4) as $D(R, Y; \Theta)$.

The hot region R^* of an image I for class c is the region that assigned the maximal v_c :

$$R^* = \arg \max_{R \in I} (v_c) \quad (8)$$

After finding the hot region R^* , the VII between image I and action query q could be builded,

$$\langle q, \{R_1^*, R_2^*, \dots, R_r^*\} \rangle \quad (9)$$

For instance, when comes to a query “running”, the online component will search the VII for this keyword: $\langle 'running', \{1, 4, 436, \dots, 3475\} \rangle$, and find the hot regions id $\{1, 4, 436, \dots, 3475\}$. Then the goal images will be linked via formula (7). Finally, the results are submitted to the re-ranking function.

3.5 Retrieval

In this paper, we focus on tackling the “Semantic Gap” between action query and visual information of images. Therefore, we treat the retrieval task as the re-ranking work, i.e., re-organize the ranking list of text-based search engine [13,18]. In the online computing component, we will fuse the results retrieved by textual information and VII. For a query q and an image I , the final ranking score is defined as:

$$rank(q, I) = \omega \times rank_t + (1 - \omega) \times rank_v \quad (10)$$

In this paper, we simply utilize the original rank value (e.g. GoogleRank, BingRank) of image I as $rank_t$. $rank_v$ is the score of the hot region R^* of image I . Finally, return the results list order by $rank(q, \{I_1, I_2, \dots, I_i\})$. On a PC with two 2.93GHz CPU, our framework can return the results below 1 second.

4 Experimental Setup

4.1 Dataset

We collect about 1200 images in total for six action queries: phoning, playing guitar, riding bike, riding horse, running and shooting. Most of the images are collected from Google Image, Bing and Flickr, and others are from PASCAL VOC 2010 [6] and PPMI dataset [22]. Each action class contains about 200 images.

4.2 Appearance Descriptors

The appearance descriptors of the candidate regions R are constructed from a number of different feature channels. These features are used in [4,7,12,20].

Dense SIFT words [12]. Rotationally invariant SIFT descriptors are extracted on a regular grid each five pixels, at four multiple scales (10, 15, 20, 25 pixel

radii), zeroing the low contrast ones. Descriptors are then quantized into 300 visual words.

Histogram of oriented edges [4]. The Canny edge detector is used to compute an edge map. The orientation and weight of each edge pixel p is assigned by the underlying image gradient. Then the orientation angle is quantized in eight bins with soft linear assignment and a histogram is computed.

Spatial pyramid. For each feature channel a three-level pyramid of spatial histograms is computed, similar to [12][20].

4.3 Implementation Details

In the exemplarlets refining procedure, we manually select about 100 exemplarlets for each action class c as positive training examples, and sample about 100 exemplarlets from other action classes as false positives (“hard examples”). After running five times five-fold cross validation via multiple kernel SVM [20], we calculate the mean score v for each exemplarlet and selected top 60 as the seed exemplarlets for each action class. The size (*height* \times *width*) of exemplarlet for each class are 200×200 , 200×200 , 300×150 , 200×200 , 300×150 and 200×300 . Some examples of exemplarlets are listed in Figure 2.

In the training procedure, detectors D is performed with the SVM classifier using the 1-vs-all scheme. We employ the Matlab package libsvm [2] as the implement of SVM. Finally, we also valid the training procedure in a multiple classification task. During the training and testing procedure, discriminative exemplarlets were refined simultaneously.

We let $L = 5$ and $ratio = 4$ to zoom images into different scales in VII building procedure. Take image $I(h \times w)$ for example, after zoom+ and zoom- processing, we will obtain five new images in different scale: $(\frac{5}{4}h \times \frac{5}{4}w)$, $(\frac{4}{4}h \times \frac{4}{4}w)$, $(\frac{3}{4}h \times \frac{3}{4}w)$, $(\frac{2}{4}h \times \frac{2}{4}w)$ and $(\frac{1}{4}h \times \frac{1}{4}w)$.

5 Results

5.1 Analysis of MKL

Multiple kernel learning is the basic learning model in our framework, we must make sure that the detectors learnt via MKL are effective and discriminative. We compare the MKL method with the-state-of-art [12][22][23].

We show results on the datasets Yao and Fei Fei [22] in Figure 3(left). Both BoW and SPM [12] use the histogram representation, while BoW does not consider spatial information in image features and SPM accounts for some level of coarse spatial information by building histograms in different regions of the image. As shown in Figure 3(left), the MKL method outperforms the approach of [22] and [12] by 1.44% to 8.71%.

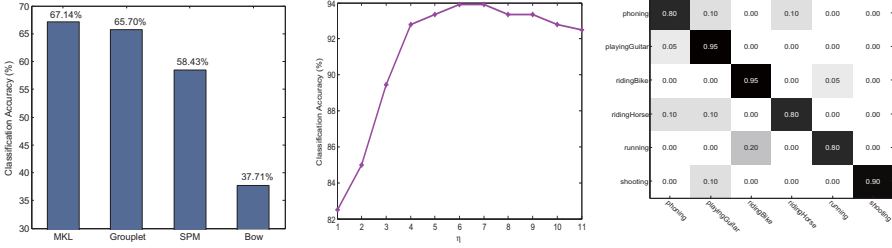


Fig. 3. Left: 7-class classification using the normalized PPMI+ images. Y-axis indicates the classification accuracy of each method on the 7 classes. Middle and right: 6-class classification using exemplarlets. Middle: the weight η for the kernel based on the SIFT feature channel, we choose $\eta = 0.7$. right: Confusion matrix obtained by MKL.

5.2 The Discrimination of Exemplarlets

We treat analyzing the discrimination of exemplarlets and refining the exemplarlets as a six-classification problem and compute the mean accuracy of five times five-fold cross validation. As shown in Figure 3(right), all the accuracy are greater than 80% and the mean accuracy is 86.67%. We believe that the exemplarlets for each action class are discriminative enough to train effective detectors. The visualization of every action is shown in the last column of Figure 2.

5.3 The Detection Results

We use the detectors learnt via MKL to detect the candidate regions R from all the test images, and employ KNN method for baseline. Then we get the detection results as shown in Figure 4. The mean detection accuracy of MKL detectors is 79.33%, however, only 30.67% for KNN method.

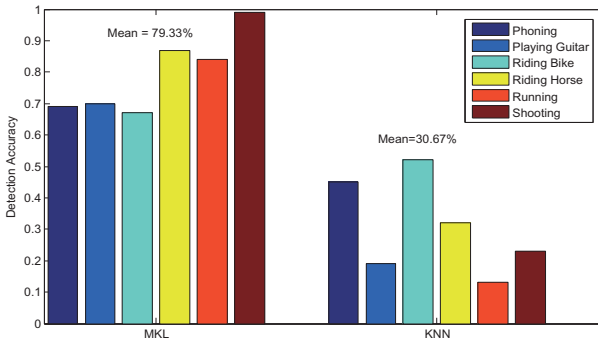


Fig. 4. The mean detection accuracy of MKL detectors and KNN method for action “phoning” “playing guitar” “riding bike” “riding horse” “running” and “shooting”

5.4 The Retrieval Results

To evaluate the quality of a ranking for a set of test images, we use many performance measures commonly used for information retrieval, all are briefed as follows:

- (1) $P@n$ (Precision at Position n) [11]

For a given query, its precision of the top n results of the ranking list is defined as Eq. (11):

$$P@n = \frac{N_{rel_n}}{n} \tag{11}$$

where N_{rel_n} is the number of relevant images in top n results.

- (2) MAP (Mean Average Precision) [12]

MAP is obtained as the mean of average precisions over a set of queries. Given a query, its MAP is computed by Eq. (12), where N_{rel} is the number of relevant images, N is the number of total retrieved images, $rel(n)$ is a binary function indicating whether the n th image is relevant, and $P(n)$ is the precision at n .

$$AP = \frac{1}{N_{rel}} \sum_{n=1}^N P(n) \times rel(n) \tag{12}$$

Moreover, we give a constraint condition that the overlap region between the detected hot regions R^* and the real hot region R_0^* is over 50%, e.g.,

$$\frac{\#(R^* \cap R_0^*)}{\#R_0^*} \geq \frac{1}{2} \tag{13}$$

For the reason that fuse the results of text-based search and VII based search together, the re-ranking results is improved significantly, here select fusion parameter $w = 0.4$. The precision at position 100 is shown in Figure 5.

MAP for MKL and KNN method is 66% and 39%.

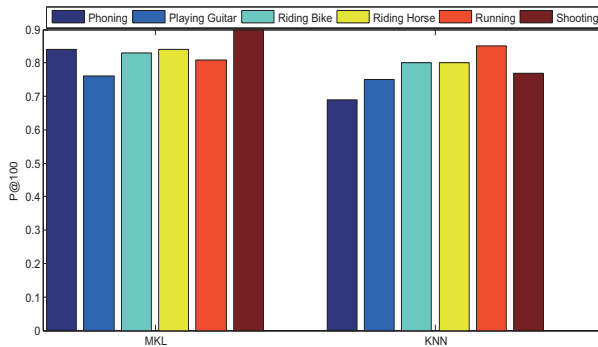


Fig. 5. The retrieval results $P@100$. MKL detectors and KNN method for action “phoning” “playing guitar” “riding bike” “riding horse” “running” and “shooting”

6 Conclusions

In this paper we describe a framework for human action retrieval in still web images by verb queries. First we build a group of visual discriminative instances for each action class, called “Exemplarlets”. Thereafter we employ Multiple Kernel Learning (MKL) to learn an optimal combination of histogram intersection kernels, each of which captures a state-of-the-art feature channel. Our features include the distribution of edges, dense visual words and feature descriptors at different levels of spatial pyramid. For a new image we can detect the hot-region using a sliding-window detector learnt via MKL, and this hot-region can imply the latent actions in the image. After the hot-region has been detected, we build an inverted index in the visual search path, termed Visual Inverted Index (VII). Finally, fusing the visual search path and the text search path, we can get the accurate results, both relevant to text and visual information.

In the future we will study an effective and efficient model to selected exemplarlets from web automatically. Moreover, we will find a new way to mine the discriminative visual descriptors for actions.

Acknowledgments. This work is supported by the Natural Science Foundation of China (60970047), the Natural Science Foundation of Shandong Province (Y2008G19), the Key Science Technology Project of Shandong Province (2007GG10001002, 2008GG10001026) and the Graduate Independent Innovation Foundation of Shandong University(YZC10067).

References

1. Baeza-Yates, R., Ribeiro-Neto, B., et al.: Modern information retrieval. Addison-Wesley Harlow, England (1999)
2. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines (2001), <http://www.csie.ntu.edu.tw/~cjlin/libsvm>
3. Chi, M., Zhang, P., Zhao, Y., Feng, R., Xue, X.: Web image retrieval reranking with multi-view clustering. In: Proceedings of the 18th International Conference on World Wide Web, pp. 1189–1190. ACM, New York (2009)
4. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 886–893. IEEE, Los Alamitos (2005)
5. Schuldts, C., Laptev, I., Caputo, B.: Recognizing human actions in still images: a study of bag-of-features and part-based representations. In: British Machine Vision Conference (2009)
6. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. International Journal of Computer Vision 88(2), 303–338 (2010)
7. Felzenszwalb, P., Girshick, R., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part based models. IEEE Transactions on Pattern Analysis and Machine Intelligence (2009)

8. Gupta, A., Kembhavi, A., Davis, L.: Observing human-object interactions: Using spatial and functional compatibility for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31(10), 1775–1789 (2009)
9. Ikizler, N., Cinbis, R., Pehlivan, S., Duygulu, P.: Recognizing actions from still images. In: 19th International Conference on Pattern Recognition, pp. 1–4. IEEE, Los Alamitos (2009)
10. Ikizler-Cinbis, N., Cinbis, R., Sclaroff, S.: Learning actions from the web. In: IEEE 12th International Conference on Computer Vision, pp. 995–1002. IEEE, Los Alamitos (2010)
11. Laptev, I., Marszalek, M., Schmid, C., Rozenfeld, B.: Learning realistic human actions from movies. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8. IEEE, Los Alamitos (2008)
12. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 2169–2178. IEEE, Los Alamitos (2006)
13. Li, P., Zhang, L., Ma, J.: Dual-ranking for web image retrieval. In: Proceedings of the ACM International Conference on Image and Video Retrieval, pp. 166–173. ACM, New York (2010)
14. Moeslund, T., Hilton, A., Krüger, V.: A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding* 104(2-3), 90–126 (2006)
15. Niebles, J., Wang, H., Fei-Fei, L.: Unsupervised learning of human action categories using spatial-temporal words. *International Journal of Computer Vision* 79(3), 299–318 (2008)
16. Popescu, A., Moëllic, P., Kanellos, I., Landais, R.: Lightweight web image reranking. In: Proceedings of the seventeen ACM International Conference on Multimedia, pp. 657–660. ACM, New York (2009)
17. Tian, X., Tao, D., Hua, X., Wu, X.: Active reranking for web image search. *IEEE Transactions on Image Processing* 19(3), 805–820 (2010)
18. van Leuken, R., Garcia, L., Olivares, X., van Zwol, R.: Visual diversification of image search results. In: Proceedings of the 18th International Conference on World Wide Web, pp. 341–350. ACM, New York (2009)
19. Varma, M., Ray, D.: Learning the discriminative power-invariance trade-off (2007)
20. Vedaldi, A., Gulshan, V., Varma, M., Zisserman, A.: Multiple kernels for object detection. In: 2009 IEEE 12th International Conference on Computer Vision, pp. 606–613. IEEE, Los Alamitos (2010)
21. Yang, W., Wang, Y., Mori, G.: Recognizing human actions from still images with latent poses. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 2030–2037. IEEE, Los Alamitos (2010)
22. Yao, B., Fei-Fei, L.: Grouplet: a structured image representation for recognizing human and object interactions. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 9–16. IEEE, Los Alamitos (2010)
23. Yao, B., Fei-Fei, L.: Modeling mutual context of object and human pose in human-object interaction activities. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 17–24. IEEE, Los Alamitos (2010)

Social Analytics for Personalization in Work Environments

Qihua Wang and Hongxia Jin

IBM Almaden Research Center, San Jose, California, USA

Abstract. A user's context in work environments, or *work context*, provides fine-grained knowledge on the user's skills, projects, and collaborators. Such work context is valuable to personalize many web applications, such as search and various recommendation tasks. In this paper, we explore the use of work contexts derived from users' various online social activities, such as tagging and blogging, for personalization purposes. We describe a system for building user work context profiles, including methods for cleaning source data, integrating information from multiple sources, and performing semantic enrichment on user data. We have evaluated the quality of the created work-context profiles through simulations on personalizing two common web applications, namely tag recommendation and search, using real-world data collected from large-scale social systems.

1 Introduction

People have various backgrounds. Knowledge on users' backgrounds is highly valuable to improve web applications through personalization. The problem of mining Internet users' backgrounds has attracted significant amount of interests in the research communities. Mining user backgrounds from private information sources, such as search history, emails, and desktop documents, has been widely studied in the literature [51]. With the growing popularity of social networks, more and more valuable information can be retrieved from users' online social activities. Researchers [74] have also recently proposed to explore folksonomy from public social bookmarking systems, such as Del.icio.us and Flickr, to create user interest profiles.

Most existing solutions on mining user backgrounds are designed for general Internet users. Little attention has been paid to users in specific environments. Different from the prior art, this paper studies mining user contexts in work environments. A user's *work context* contains work-related information, such as skills, projects, and collaborators. Users' work contexts may be used by organizations to improve their internal web applications. Many organizations, such as schools, government agencies, and large companies, provide a variety of web applications to allow their employees to publish information and manage resources in the intranets. Example applications include resource annotations, search, and news recommendation. Most of these applications may benefit from personalization based on the work contexts of users.

While public social activities provide valuable information about users, there are some known challenges in using such data to mine user contexts. First, the texts on social systems are oftentimes informal and may even be broken. In particular, many

compound words in users' tags may be broken into individual words, (some not even in right order), which change the semantics of the annotations. Furthermore, the texts on many items may be short and the semantics of words may be unclear from their textual content. For that, existing work [3] turns to outside source such as WordNet for help.

When mining user context from social activities in working environment, there are additional challenges which make the existing approaches infeasible to use:

- Work context contains fine-grained knowledge about users. Many users in an organization share similar backgrounds. For example, many employees in an IT company have background in computer science. The information in users' work contexts needs to be fine-grained enough to distinguish different sub-fields in a broad area. For instance, the acronym "SNA" has 16 possible meanings in Wikipedia. Even if we only consider the entries related to computer science, the word "SNA" may mean "system network architecture", "social network analysis", and so on. To understand a query term "SNA" from a user, it does not suffice to know that the user is a computer scientist; we need further knowledge on the specific areas in computer science which the user works on. Finer-grained knowledge in work context imposes higher requirements on the semantics in user information.
- The semantics of words may differ from one work environment to another. Unlike existing solutions for Internet users, one cannot turn to out outside sources, such as Wikipedia or WordNet, for semantic information about words in a work environment. For instance, "Koala" may be the name of some internal project in an organization, but it has a different meaning in the outside world. We have to mine the semantics of words from the available data in an organization.

We tackle the above challenges in our solution. Our contributions include: (1) We design an approach to semantically model users' work contexts from multiple public social information sources that provide heterogeneous data. The words in a user's work-context profile are semantically enriched. We propose to mine the semantics of the words through three aspects: co-occurrences on resources, co-occurrences on people, and users' social connections. (2) In order to show that our approach may generally benefit different types of web applications, we perform simulations by using work contexts to personalize two popular applications: search and tag recommendation. Our simulations are performed on real-world data collected from large-scale social systems. Compared to existing user modeling approaches on social data, such as [10] and [2], that are specifically designed to personalize a certain application, we demonstrate that our solution is more general and may be use to enhance various applications.

2 Deriving User Context Profile from Online Social Activities

We learn about a user's work context through her activities on multiple online social systems in an organization. Potential social information sources include blogs, mini blogs, social bookmarks, mutual tags (or people tagging), and so on. The major steps and data flow of our approach are shown in Figure 1.

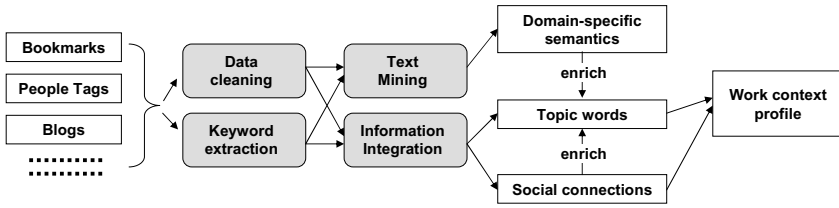


Fig. 1. Steps and data flow for mining work context from multiple social information sources

2.1 Data Cleaning: Semantically Repairing Broken Tags

As a preparation step, our system performs cleaning on user data, especially free-form tags. In particular, we observe that there exist compound words in tags that are broken into multiple individual words. The broken-tag problem arises when a tagging system misunderstands the tags entered by users. The user interface of many systems allows a user to enter multiple tags at a time, assuming that different tags are separated by spaces. If a tag is a compound word (e.g. “new york”), the user is supposed to connect the components of the tag with connecting symbols (e.g. “new-york”). However, using connecting symbols is not the usual way people write those words in everyday life. Users, especially those new to tagging, often forget to apply the connecting symbols in a compound word. When the connecting symbols are missing, a tagging system may think that multiple individual tags, rather than one compound-word tag, are entered. The misunderstanding results in breakup of the compound word. For instance, given the input sequence “social network analysis”, the system will think that the user enters three tags “social”, “network” and “analysis”.

The most important step in broken tag recovery is to identify the broken compound words from a set of tags. Recognizing phrases in a sequence of words is well-studied [6]. A common solution on phrase recognition is to compare the word-level bigrams, trigrams, ..., n -grams in a word sequence with the compound words in a dictionary. Unfortunately, the assumption on correct word order does not hold in the context of broken tag recovery, as many tagging systems may automatically rearrange the individual tags in alphabetical order (see Figure 2 for a real-world example).

Next, we describe our TagFix algorithm on recovering broken tags. The algorithm consists of a preparation phrase and a recovery phrase. See Figure 2 for example recovery results by TagFix.

Preparation. TagFix creates a dictionary from the correctly-applied compound words in existing tags. For each compound word w in the dictionary, TagFix computes the set S_w of tags that frequently co-occur with w . The set S_w is recorded as the context of w in the dictionary. For each $t \in S_w$, we store the probability $p(w|t)$ that an item that is annotated with t is also tagged with w .

Recovery. A tagging instance is represented as a tuple $\langle I, X, u, T \rangle$, where I is an item, X is the text of I , u is a user, and T is a set of tags. The instance indicates that u has applied tags in T to I . When the instance is from social bookmarks, I is a web resource and X is a snippet of I ; when the instance is from mutual tags, I is a person and X is

SNA Learning Series and Self Study Resources - Social Network Analysis - w3ki

 Sven Loeffler and 3 others - Jan 17 | Tags: analysis, learning, network, sna, social

Countdown to product launch: Are you confident customers will buy?

 RAJHESH PANCHANADHAN - Nov 23 2009 | Tags: analysis, development, network, new, product, social

Before Fixing

SNA Learning Series and Self Study Resources - Social Network Analysis - w3ki

 Sven Loeffler and 3 others - Jan 17 | Tags: learning, sna, social-network, social-network-analysis

Countdown to product launch: Are you confident customers will buy?

 RAJHESH PANCHANADHAN - Nov 23 2009 | Tags: new, product-development, social-network, social-network-analysis

After Fixing

Fig. 2. Screenshots from Bookmarks in Lotus Connections. Note that the broken components of the phrases “social network analysis” and “product development” are not in right order rendering a simple n-gram type approach useless, but our algorithm recovers those broken phrases.

empty. TagFix examines tagging instances one by one. Given $\langle I, X, u, T \rangle$, TagFix does the followings to recover the potential broken tags in T .

First, we find a set S_c of candidate compound words in the dictionary, such that for every $w \in S_c$, all the components of w appear as individual tags in T .

Second, we compute the set S_I of contextual words of the item I . S_I contains all the tags that have been applied to I by any user in the system, including u . When the text X of I is not empty, we also add the keywords in X to S_I .

Third, for each candidate compound word w in S_c , we compute the probability that I is (not) related to w as $p(\neg w|t_1, \dots, t_m) = p(\neg w|t_1)p(\neg w|t_2) \dots p(\neg w|t_m)$, where t_1, \dots, t_m are the contextual words in S_I , $p(\neg w|t_i) = 1 - p(w|t_i)$, and $p(w|t_i)$ is the probability that an item that is annotated with t_i is also tagged with w (recall that $p(w|t_i)$ is computed and stored in the dictionary in the preparation phase).

If $p(\neg w|t_1, \dots, t_m)$ is smaller than a threshold θ (say, 25%), we have high confidence that w is semantically related to I . In this case, we should recover w for I . To recover w , we connect its components with “-” and add it to T . We also remove the individual tags that are components of w from T . This deletes the broken pieces and adds back the recovered compound words.

2.2 Information Retrieval and Integration

We retrieve users’ information from multiple public social systems in an organization. Given a user’s activities on a social system, we extract two vectors from her activities: a *topic-word vector* and a *social-connection vector*. A topic-word vector is represented as $\{(w_1, c_1), \dots, (w_m, c_m)\}$, where w_i is a keyword and c_i is a real-number score of w_i in the vector. Similarly, a social-connection vector is represented as $\{(u_1, c_1), \dots, (u_n, c_n)\}$, where u_i is a user and c_i is a real-number score of u_i in the vector. The approach on information retrieval depends on the information source. In the following, we discuss three example sources.

- *Blogs*: Each blog entry of a user u_i is represented as a pair $\langle X_j, U_j \rangle$, where X_j is the textual content of the blog entry and U_j is the set of people commented on the entry. We extract the top k (say, $k = 5$) keywords from X_j ; to do so, we may compute the TF-IDF score of each word in X_j and take the k ones with the highest scores. The topic-word vector V_1^t on u_i 's blogs contains all the top k keywords extracted from u_i 's blog entries. For each item (w_l, c_l) in V_1^t , we have $c_l = \log(N_w(w_l) + 1)$, where $N_w(w_l)$ is the number of u_i 's blog entries that have w_l as one of their top k keywords. Similarly, the social-connection vector V_1^c on u_i 's blogs contains all the people who have commented on u_i 's blog entries. For each item (u_l, c'_l) in V_1^c , we have $c'_l = \log(N_u(u_l) + 1)$, where $N_u(u_l)$ is the number of u_i 's blog entries that have comments from u_l .
- *Social bookmarks*: Each bookmark entry of a user u_i is represented as a pair $\langle X_j, T_j \rangle$, where X_j is the textual content of the bookmark (usually the snippet of the bookmarked webpage) and T_j is the set of tags u_i applies on the bookmark. The topic-word vector V_2^t on u_i 's social bookmarks contains all the tags in u_i 's bookmark entries. For each item (t_l, c_l) in V_2^t , we have $c_l = \log(N_t(t_l) + 1)$, where $N_t(t_l)$ is the number of times u_i has applied t_l as tag in her bookmarks. In contrast, the social-connection vector on social bookmarks is always empty, since social bookmarks do not directly indicate the connections among users.
- *Mutual tags*: Each mutual-tagging instance in the system is represented as $\langle u_1, u_2, T_j \rangle$, where u_1 and u_2 are users and T_j is a set of tags. The instance indicates that u_1 has applied tags in T_j on u_2 . In this case, the tags in T_j are among the *incoming tags* of u_2 and among the *outgoing tags* of u_1 . Note that even if a user is not active on a mutual-tagging system, she may still have incoming tags, because her friends or colleagues who are active may apply tags on her.

The topic-word vector V_3^t on u_i 's mutual tags contains all of her incoming tags and outgoing tags. For each item (t_l, c_l) in V_3^t , we have $c_l = \log(N_{in}(t_l) + 1) + \log(N_{out}(t_l) + 1)$, where $N_{in}(t_l)$ and $N_{out}(t_l)$ are the numbers of times t_l serves as an incoming tag and an outgoing tag for u_i , respectively. It is also possible to place more weight on incoming tags or on outgoing tags when constructing V_3^t , but this is not currently implemented so as to keep our solution simple.

Similarly, the social-connection vector V_3^c on u_i 's mutual tags contains all the people who have applied tags on u_i or received tags from u_i . For each item (u_l, c'_l) in V_3^c , we have $c'_l = \log(N_p(u_l, u_i) + 1) + \log(N_p(u_i, u_l) + 1)$, where $N_p(u_1, u_2)$ is the number of tags u_1 has applied on u_2 . Note that the computation of c'_l favors users who have mutual interactions with u_i (i.e. both applied tags on u_i and received tags from u_i).

Even though we retrieve user information from multiple online social systems, we do not require a user to be active on every system to create a profile for her. If a user has no activity on a certain system, her vectors corresponding to the system are empty.

A user's work-context profile also contains a topic-word vector V_{all}^t and a social-connection vector V_{all}^c . The topic-word vector V_{all}^t is acquired by combining the topic-word vectors from all the social information sources we consider. Let V_1^t, \dots, V_n^t be the topic-word vectors from those sources. V_{all}^t contains all the keywords in V_1^t, \dots, V_n^t .

For each item (w_l, c_l) in V_{all}^t , we have $c_l = \sum_{i=1}^n \alpha_i \cdot c(V_i^t, w_l)$, where $c(V_i^t, w_l)$ is the score of w_l in vector V_i^t and α_i is the combination weight of the vector V_i^t . For simplicity, we set all the combination weights as one in the current implementation. Determining different combination weights for different types of information is interesting future work. The social-connection vector in a work-context profile may be computed similarly by combining the social-connection vectors from different sources.

2.3 Semantic Enrichment

So far, the topic-word vector in a user's work-context profile contains only the words appearing in the user's public social activities. As stated in Section 1, the text associated with an item on a social system is normally short. The meanings of words in a piece of short text may be unclear. To acquire fine-grained knowledge about a user's work context, we need to semantically enrich the topic-word vector in her work-context profile. More specifically, we would like to expand the words in the topic-word vector with their domain-specific synonyms and closely related terms. Furthermore, the meanings of words in work environments may be domain-specific. We mine information on synonyms and related terms from the available data rather than turning to outside sources such as Google and Wikipedia for semantic information. Our mining is based on the words' co-occurrences on resources as well as their co-occurrences on people.

Word co-occurrence on resources has been widely used in existing literature to mine synonyms and related terms. Intuitively, the number of topics a resource has is very limited; thus, if two words are often found to be on the same resources, they are likely to be on the same topic. For any word w , let $R(w)$ be the set of resources that contain or are annotated with w . Also, let $R(w_1, w_2)$ be the set of resources that contain or are annotated with both w_1 and w_2 . We compute the correlation degree of two words w_1 and w_2 with regards to resources as $coh_r(w_1, w_2) = |R(w_1, w_2)| / (|R(w_1) \cup R(w_2)|)$. The words w_1 and w_2 are considered to be closely related if $coh_r(w_1, w_2)$ is larger than a threshold δ_r , where $\delta_r \in (0, 1]$.

In addition to co-occurrence on resources, we also attempt to discover word semantics from co-occurrence on people. For example, if we find that most people associated with the project "Koala" have knowledge on "data mining", we may infer that "Koala" is related to "data mining". Let $U(w)$ be the set of users whose current topic-word vectors contain w . We may compute the correlation degree $coh_p(w_1, w_2)$ of two words w_1 and w_2 with regards to people in a similar way as $coh_r(w_1, w_2)$. The words w_1 and w_2 are considered to be closely related if $coh_p(w_1, w_2)$ is larger than a threshold δ_p . Note that in most cases, the threshold δ_p should be greater than δ_r , because the number of topics a person has is usually greater than the number of topics a resource has. We set a larger correlation threshold for co-occurrence on people so as to reduce the false positive rate. By mining information from both co-occurrence on resources and co-occurrence on people, we may acquire more comprehensive knowledge about the domain-specific semantic information of words. To enrich a user's work-context profile, for every word w in the user's topic-word vector, we add w 's closely related words to the vector; the scores of those added related words may be a portion (say, 50%) of the score of w in the vector so as to attach more importance to w .

Finally, we may also enrich a user’s work-context profile using the profiles of her most connected people. A user’s most connected people are those with the highest scores in her social-connection vector. The intuition is that close friends or colleagues usually have something in common. If a user does not have much information in her work-context profile (because she is not an active social-web user), we may treat the common context of her close connections’ as the user’s own work context. However, we do not directly add the important topic words of a user’s closely connected people to her profile. We refer to connected people’s topic-word vectors at runtime so as to take advantage of people’s most up-to-date profiles. We will see examples on this in Sections [3.1](#) and [3.2](#).

3 Evaluation

We have implemented our solution on deriving work context from users’ social activities and performed evaluation. In particular, we would like to answer the question: Are the derived user contexts useful for personalizing various web applications?

In our evaluation, we perform simulations over a larger scale of user data. Our simulations aim to evaluate whether a work context profile may be used to personalize different types of web application. The two applications we choose in our tests are tag recommendation and web search. The reason we select these two applications is because they focus on very different aspects: tag recommendation assist users to *input* information, while web search helps users to *retrieve* information. Information input and retrieval are among the most fundamental services provided by web applications. If both tag recommendation and web search can benefit from work contexts, it is likely that many other common web applications in work environment, such as news filtering, expertise finding, can also benefit from work contexts similarly. We design simple approach to personalize these two applications using work-context profiles.

There exists a lot of other work on personalized tag recommendation [\[8,9\]](#) and personalized search [\[5,7,4\]](#). Due to space limit, we are unable to discuss individual work in detail. The solutions proposed in the existing work are specialized on a single application. It is not clear whether or how a proposed solution may be generalized to personalize applications other than the targeted one. We emphasize that our objective is not to show that work contexts can perform better than the above specialized solutions. Instead, we would like to test if work contexts can be generally and reasonably useful to personalize different applications in simple ways.

3.1 Personalized Tag Recommendation

We describe a personalized tag recommendation algorithm that takes advantages of users’ work-context profiles. The proposed personalized tag recommendation approach will be evaluated using simulation over real-world data. Our goal is to determine whether and how much tag recommendation may benefit from work-context profiles.

Personalized Tag Recommendation. Assume that we recommend tags to Alice on resource R . As the first step, we retrieve three vectors. The first vector V_1 is the keyword vector of the textual content of R . V_1 contains all the words in R and each word in V_1 is

associated with its TF-IDF weight in R . The second vector V_2 is the topic-word vector in Alice's work-context profile. The third vector V_3 is the union of the topic-word vectors of Alice's top k (say, $k = 3$) connected people, i.e. the k users with highest scores in Alice's social-connection vector. By taking V_2 and V_3 into account, our recommender personalizes its suggestions for Alice, as different people will have different V_2 and V_3 .

Next, we combine the three vectors to get a candidate-tag vector. The candidate-tag vector contains all the words in V_1 , V_2 and V_3 . Each candidate tag is associated with a priority score; the higher the score, the higher the candidate tag is ranked. To compute the priority scores, we first normalize the scores of words in V_1 , V_2 and V_3 so that the average scores of their top x (say, $x = 10$) words are equivalent. Second, for each candidate tag w_i , its priority score $p(w_i)$ is computed as:

$$p(w_i) = \beta_1 \cdot c(V_1, w_i) + \beta_2 \cdot c(V_2, w_i) + \beta_3 \cdot c(V_3, w_i)$$

where $c(V_j, w_i)$ is the score of w_i in vector V_j and β_j is the combination weight of vector V_j . In the current implementation, we have $\beta_1 = 3$, $\beta_2 = 2$, and $\beta_3 = 1$. In other words, we attach most importance to the keywords in R ; we also give heavier weights to the words from Alice's own work-context profile than those from her connected people.

Our tag-recommendation approach employs type-ahead recommendation: we only recommend tags that start with the prefix that the user has entered and the list of recommended tags changes as the user types. Type-ahead recommendation is widely used in practice, such as Google's recommendation on search keywords. Let pre be the prefix Alice has entered. Our tag recommender selects all the candidate tags starting with pre and ranks them based on their priority scores. In particular, when pre is empty, all candidate tags will be chosen. The top y selected candidate tags with the highest scores are suggested to Alice, where y is the maximum number of suggestions that are allowed.

Keystroke-based Measurement. Almost all existing work on tag recommendation evaluates their solutions using recall. However, most real-world tagging systems employ type-ahead recommendation. Recall, as a static measure, does not reflect the dynamic input process with type-ahead. Here, we propose a novel test method to evaluate type-ahead tag recommendation. Our test method aims to measure the average amount of effort a user makes to enter a tag with the help of a tag recommend approach. A user's effort is measured by the number of keystrokes. For example, to enter the tag "python", if a recommendation approach suggests the tag before the user types anything, the number of keystrokes is zero; if the approach recommends the tag after the user types two characters (i.e. "py"), the number of keystrokes is two; in the worst case, the number of keystrokes needed to enter "python" is six (i.e. length of the word).

In our test method, we simulate user input by gradually providing the prefix of a tag to a type-ahead recommender until the tag is suggested by the recommender or the entire tag has been entered. We compute the *average number of keystrokes per tag* (ANKT) with regards to a user as a metric for tag recommenders. Assume that a user u makes sum keystrokes to input m tags. The ANKT of u is $a(u) = sum/m$. While ANKT is a straightforward measurement of user effort in tagging, it does not reveal how much effort is saved with the help of a tag recommendation solution. We further define the *average percentage of keystrokes per tag* (APKT) as another criterion. Let $p(u)$ denote the APKT for user u and T be the set of tags entered by u . We have $p(u) = \frac{1}{|T|} \sum_{t \in T} \frac{k_t}{l_t}$, where l_t is the length of the tag t , and k_t is the number of keystrokes u

made to enter t with the help of the tag recommender. As to the values of ANKT and APKT, the smaller the better.

Even though recall is the most widely used criterion for tag recommendation, as we will see from experimental results, the fact that a recommender has a higher recall than another does not automatically imply that the former has a smaller ANKT or APKT than the latter. The metrics ANKT and APKT we proposed complement recall as effective evaluation criteria for type-ahead recommenders in the real world.

Baseline Algorithms. For convenience, we refer to our work-context-based tag algorithm as *WorkContext*. We compare *WorkContext* with six baseline algorithms. The six baseline algorithms implement three common tag recommendation strategies and the combinations of them. All the six baseline algorithms provide type-ahead recommendation. The three tag recommendation strategies are referred to as *Self*, *Keyword*, and *Popular*, respectively. The three strategies differ in the choices of vocabulary and the ways to rank candidate tags.

- *Self* makes recommendation based on the current user's past tags. The more times the user has used the tag t , the higher ranking score t has. The idea behind *Self* is that users often repeat their favorite tags.
- *Keyword* suggests tags based on the keywords of the target webpage. The more important a word is with regards to the webpage (e.g. higher TF-IDF weight), the higher ranking score the word has.
- *Popular* makes recommendation based on the popularity of existing tags. The more times a tag t is used in the tagging system, the higher ranking score t has. Simple as it is, *Popular* is being used by many, including Amazon and IBM's Lotus Connections.

A hybrid algorithm may employ any two of the above three strategies. The three hybrid algorithms we consider are referred to as *Pop+Self*, *Key+Pop*, and *Key+Self*, respectively. *Key+Pop*, which combines the tags suggested by *Keyword* with those recommended by *Popular*, is a common collaborative filtering algorithm. *Key+Self* combines *Keyword* and *Self*, and is a simple personalized tag recommendation algorithm that considers target resource's textual content as well as the user's past tags.

Data Sets. Our experimental data was collected from a real-word Lotus Connections system in IBM. Lotus Connections is a large-scale work-oriented social computing system that contains a blogging sub-system called *Blogs*, a social bookmarking system called *Bookmarks*, and a people-tagging system called *PeopleTags*. Our data set from *Bookmarks* consists of more than 420,000 bookmarks contributed by over 5,000 users. In our *PeopleTags* data set, more than 54,000 users have been tagged with over 170,000 mutual tags. In our experiments, for each user, 25% of his/her bookmarks in *Bookmarks* were used in mining work context, while all the other 75% were used as test cases. There is no overlap between the two sets of bookmarks so as to avoid bias.

Not every user is involved in all the three types of online social activities we considered. In our experiments, we chose those users who have written at least 10 blogs, received no less than 10 people tags, and bookmarked 20 webpages or more. There are 208 users in our data sets that meet our selection criteria. On average, each of the selected user has 26 people tags, 129 blogs, and 391 bookmarks.

Table 1. Experimental results on tag recommendation, comparing with baseline methods

Recommendation Methods		Work Context	Self	Keyword	Popular	Key+Self	Key+Pop	Pop+Self
Recall	Top 5 Tags	0.23	0.15	0.19	0.04	0.19	0.13	0.09
	Top 10 Tags	0.30	0.22	0.24	0.05	0.23	0.15	0.14
ANKT	Top 5 Tags	2.25	3.81	5.14	3.48	4.14	3.56	3.37
	Top 10 Tags	1.96	3.64	5.02	3.09	3.98	3.12	3.01
APKT	Top 5 Tags	0.28	0.48	0.43	0.63	0.51	0.44	0.42
	Top 10 Tags	0.24	0.46	0.61	0.38	0.48	0.39	0.37

Experimental Results. Our experiments were performed on a workstation with an Intel Core 2 Duo Processor at 3GHz and 3GB of main memory. It took our system 651 seconds to create the work-context profiles for all the 208 selected users. The average time to create a work-context profile for one user was 3.2 seconds. This did not include retrieving user information from the three social systems. Depending on the amount of information a user has, the time to retrieve a user's information ranged from tens of seconds to a few minutes. All improvements reported below are statistically significant at 0.01 according to t-test.

First, we consider the recall of different recommenders. The results are shown in Table 1. `WorkContext` had the highest recalls, 0.23@5 and 0.30@10. `Keyword` took the second place in the tests. The performances of `Key+Self` and `Self` were close to `Keyword`, while other baseline algorithms performed considerably worse. An interesting observation is that `Popular` performed poorly in the tests with recalls lower than 0.1. This may not be very surprising as `Popular` considers neither the target document nor the current user when it makes recommendation. But why many practical systems still use `Popular` if it is so poor in suggesting what users want? As we will see next, the real-world performance of `Popular`, when measured by ANKT and APKT, may not be as poor as indicated by recall.

Second, we consider ANKT and APKT. `WorkContext` had the lowest ANKTs, 2.25@5 and 1.96@10. Its advantages over other recommenders were pretty large. The average length of the tags in the experiments was 7.92. `WorkContext`'s APKTs were 0.28@5 and 0.24@10. This indicates that, with the help of `WorkContext`, a user may find a tag by typing about a quarter of its length on average. This indicates that `WorkContext` is highly effective in predicting user input. It is worth noting that, even though `Popular` performed poorly in recall, it had a better performance than `Self` and `Keyword` when it comes to ANKT/APKT. `Popular`'s simplicity and decent performance in ANKT/APKT demonstrate that it is a usable approach in practice. Our results indicate that a higher recall does not necessarily lead to a lower ANKT/APKT. Therefore, measuring only recall is not sufficient in evaluating the performance of tag recommenders in real world, where type-ahead is the mainstream.

In general, the fact that `WorkContext` outperforms other recommenders with regards to all the three criteria (recall, ANKT, APKT) demonstrates the effectiveness of our approach on mining users' work contexts from their public online social activities.

3.2 Personalized Search

We describe and evaluate a simple search personalization approach that makes use of work contexts. Our solution employs result processing to perform search personalization and may be used with most existing search engines.

First, upon receiving a search query from a user Alice, we forward the query to the underlying search engine. The search engine will then return a list of webpages for the query. The search engine will also associate a relevance score to each returned webpage, such that webpages which are ranked higher (i.e. those considered to be more relevant and important to the query) have higher relevance scores. The relevance scores will be normalized into $[0, 1]$.

Second, we retrieve the topic-word vector in Alice's work-context profile. We also retrieve the topic-word vectors of Alice's top k (say, $k = 3$) connected people and combine them with her topic-word vector into an overall vector. In the current implementation, the weight of Alice's topic-word vector is twice of the weights of those from her connected people, i.e., we attach more importance to Alice's own work context.

Third, for each of the webpages returned by the underlying search engine, we compute an interest score based on how well the webpage matches Alice's overall topic-word vector. The interest score is computed as the cosine similarity between the word vector of the webpage (or its snippet) and Alice's overall topic-word vector.

Fourth, we compute a final ranking score for each of the webpages. Let $g_r(x)$, $g_i(x)$, and $g_f(x)$ be the relevance score, the interest score, and the final ranking score of webpage x , respectively. We have $g_f(x) = g_r(x) \times (1 - \alpha) + g_i(x) \times \alpha$, where α is a real-number in $[0, 1]$. The value of α determines to what extent the ranking of the search results is affected by Alice's work context. In our experiments, α was set to 0.25.

Finally, the webpages are re-ranked based on their final ranking scores, and the new list of sorted results is returned to Alice.

Evaluation Method. We employ the evaluation method proposed in [7], which uses people's bookmarks and tags to represent their opinions. The intuition is that, if a user u bookmarked a webpage and tagged the webpage with a word t , then u must consider the webpage useful and relevant to t . We may issue a query consisting of the keyword t on behalf of u , and then check whether those webpages tagged with t by u are ranked among the top k results returned by the personalized search system. A drawback of this evaluation method is the potentially high false negative rate: a webpage not being tagged with t may still be considered useful by u . In spite of the drawback, this method still allows us to get a pretty good idea on the performance of a personalized search system. The same evaluation method is also used in other existing work [2].

The data set we used to test search personalization is the same as the one for tag recommendation in Section 3.1. We tested the 208 chosen users one by one. For each user, 25% of her bookmarks from Bookmarks together with her blogs and people tags were used to create her work-context profile; the other 75% of the user's bookmarked webpages in Bookmarks (which do not overlap with the 25% bookmarks used in profile building), together all the webpages bookmarked by other Bookmarks users, were used as the testing corpus. For the i th user u_i , we randomly selected 30 words u_i has applied as tags on her bookmarks. For each of those 30 words, a search query consisting of

Table 2. Experimental results on search

Search Methods		Normal	Work Context	Mutual Tags	Bookmarks	Blogs
Recall	Top 10 Results	0.209	0.288	0.251	0.260	0.263
	Top 20 Results	0.319	0.438	0.389	0.401	0.400
Improvement Rate	Top 10 Results	0	0.378	0.201	0.244	0.258
	Top 20 Results	0	0.305	0.219	0.257	0.254

the word was issued on behalf of u_i . We then compute the recall of the top k results returned by the search algorithm.

Experimental Results. For convenience, we refer to our work-context-based search approach as *WorkContext* and the underlying non-personalized search approach as *Normal*. Our experimental results are given in Table 2. All improvements reported below are statistically significant at 0.01 according to t-test.

In the first set of experiments, we test whether performing personalization using work context improves the search quality. From Table 2, we can see that *WorkContext* outperformed *Normal*. The recalls of *WorkContext* were 0.288@10 and 0.438@20, comparing to *Normal*'s 0.209@10 and 0.319@20. In other words, *WorkContext* improved the recall of *Normal* by 37.8% and 30.5% when we consider top 5 and top 10 search results, respectively. The experimental results demonstrate that the created work-context profiles are effective in prioritizing search results.

In the second set of experiments, we study whether using information from multiple social systems is superior to relying on information from a single source with regards to mining users' work contexts. We compare work-context profiles created from three information sources with those built from a single information source only. For convenience, we call search approaches that use the special versions of work-context profiles created with only blogs, only social bookmarks, and only mutual tags, *Blogs*, *Bookmarks*, and *MutualTags*, respectively. From Table 2, we can see that *WorkContext* had higher recalls than all of its three special versions. Such a result, together with our findings in Section 3.1, provide sufficient evidence that integrating information from multiple social information sources does allow us to gain more comprehensive knowledge about a user. The results also show that our approach is effective in such information integration.

4 Conclusion

We have proposed a general and lightweight end-to-end solution to mine users' work contexts from different types of public online social activities. As a data cleaning step, we have designed an approach to semantically recover the broken words in the folksonomy on social tagging systems. We have also described how to integrate and semantically enrich the information we retrieve from heterogeneous data to create a work-context profile for a user. Finally, we have evaluated the quality of the created work-context profiles by using them to personalize two common web applications.

References

1. Cao, H., Jiang, D., Pei, J., Chen, E., Li, H.: Towards context-aware search by learning a very large variable length hidden markov model from search logs. In: WWW 2009: Proceedings of the 18th International Conference on World Wide Web (2009)
2. Carmel, D., Zwerdling, N., Guy, I., Ofek-Koifman, S., Har'el, N., Ronen, I., Uziel, E., Yogev, S., Chernov, S.: Personalized social search based on the user's social network. In: CIKM 2009: Proceeding of the 18th ACM Conference on Information and Knowledge Management (2009)
3. Hu, X., Sun, N., Zhang, C., Chua, T.-S.: Exploiting internal and external semantics for the clustering of short texts using world knowledge. In: CIKM 2009: Proceeding of the 18th ACM Conference on Information and Knowledge Management (2009)
4. Noll, M.G., Meinel, C.: Web search personalization via social bookmarking and tagging. 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. 367–380. Springer, Heidelberg (2007)
5. Qiu, F., Cho, J.: Automatic identification of user interest for personalized search. In: WWW 2006: Proceedings of the 15th International Conference on World Wide Web (2006)
6. Wang, X., McCallum, A., Wei, X.: Topical n-grams: Phrase and topic discovery, with an application to information retrieval. In: ICDM 2007: Proceedings of the 7th IEEE International Conference on Data Mining (2007)
7. Xu, S., Bao, S., Fei, B., Su, Z., Yu, Y.: Exploring folksonomy for personalized search. In: SIGIR 2008: Proceedings of the 31st International Conference on Research and Development in Information Retrieval (2008)
8. Garg, N., Weber, I.: Personalized, interactive tag recommendation for flickr. In: RecSys 2008: Proceedings of the 2008 ACM Conference on Recommender Systems (2008)
9. Rendle, S., Schmidt-Thieme, L.: Pairwise interaction tensor factorization for personalized tag recommendation. In: WSDM 2010: Proceedings of the 3rd ACM International Conference on Web Search and Data Mining (2010)
10. Wang, Q., Jin, H.: Exploring online social activities for adaptive search personalization. In: CIKM 2010: Proceeding of the ACM Conference on Information and Knowledge Management (2010)

Finding Appropriate Experts for Collaboration

Zhenjiang Zhan¹, Lichun Yang¹, Shenghua Bao²,
Dingyi Han¹, Zhong Su², and Yong Yu¹

¹ Shanghai Jiao Tong University
Shanghai, China

{zhanzj, lichunyang, handy, yyu}@apex.sjtu.edu.cn

² IBM China Research Laboratory
Beijing, China

{baoshhua, suzhong}@cn.ibm.com

Abstract. In our daily life, people usually want to find someone to collaborate with for the purpose of information sharing or work cooperation. Studies show social relationships play an important role in people's collaborations. Therefore, when finding appropriate experts for a user, two aspects of an expert candidate should be considered: the expertise and the social relationship with the user. One basic model is to filter out expert candidates by one aspect and rank them by the other (FOM). Another basic model tries to combine them using linear combination method (LCM). Both models as baselines here fail to exploit the intrinsic characteristic of social relationships for the tradeoff between two above aspects. In this paper, we formally define two factors respectively (i.e., expert authority and closeness to user) and propose a novel model called friend recommendation model (FRM) which tightly combines both factors in a natural friend recommendation way and is formalized by probability and Markov Process theories. Experiments were carried out in a scenario that a user looks for coauthors in the academic domain. We systematically evaluated the performances of these models. Experimental results show FRM outperforms other two basic models in finding appropriate experts.

1 Introduction

Expert finding is nowadays a very hot research topic in the Information Retrieval (IR) field. It aims to extract expert information (e.g., name, address, contact detail, etc.), and to return these information to users for further collaboration.

For collaboration, it is important for users to get rapid response from those experts they referred to, and afterwards to communicate with them easily, smoothly and effectively. Even though previous expert finding models [1, 2, 7, 13] in the IR field return the precise ranked list of experts based on their authority on a given query, these experts may not be suitable for collaboration. Consider that you are looking for an expert to help you make investments. You turn to an expert finding system to look for those who are good at investing. System returns a ranked expert list to you, then you contact the top-ranked expert (probably Warren Buffett) and look forward to his response. Here comes the problem. These top-ranked experts may be too “far” for

users to reach, and their responses may not be in time to users. People are mostly more willing to interact with “familiar” persons than strangers. When we collaborate with others, it would be more effective if they are “close” to us. Moreover, users are likely to pay more attention to advice from those familiar and trusted experts. We can conclude that finding appropriate experts on some specific topic for collaboration is not only related to their authority on the topic, but also to their closeness to the user at least in three aspects: expert’s responsiveness, efficiency of communication, and the user’s choice for answers or advice.

Hence, choosing an appropriate expert to collaborate with should include two aspects: (a) estimating the expertise of the expert candidate on some specific topic; and (b) measuring the responsiveness and efficiency of collaboration with the expert candidate. Studies have revealed the importance of the underlying social network structure for understanding the patterns of collaboration and information sharing [3, 6]. In the Computer-Supported-Cooperative-Work (CSCW) field, researchers have studied how to incorporate social relationships into finding an appropriate expert using users’ social networks. Nevertheless, to our best knowledge, previous work [10, 12, 14] only focused on the study of users’ behaviors and psychology. Regarding how to combine these two aspects to find appropriate experts, it has not been clearly defined, sufficiently modeled and systematically evaluated.

In this paper, to solve the problem of finding appropriate experts for collaboration, we first formally define the problem as well as two factors which should be considered simultaneously: (a) expert authority, representing the expert candidate’s knowledge on specific topic; and (b) closeness to a user, modeling the social distance between the user and the expert candidate. By taking both expert authority and closeness between the user and expert candidates into consideration, Filtering Out Model (FOM), Linear Combination Model (LCM) are presented here as baseline models, while Friend Recommendation Model (FRM) is proposed as a novel model. In these models, FOM filters out expert candidates by applying these factors step by step; LCM tries to combine factors using linear combination method; and FRM is motivated by the procedure how we look for experts with some specific expertise in our daily life (see Sec. 3.3 for details). We carried out experiments on DBLP data set in a scenario that a user looks for coauthors on some specific topic. We systematically evaluated these models under different parameters settings. Experimental results show that FRM outperforms others in all measure metrics.

The remainder of this paper is organized as follows. We give our definitions of the problem and two related factors in Section 2. Then, we describe three models, FOM, LCM and FRM in detail in Section 3. Section 4 presents our experimental evaluation setting using the DBLP data set including the assessment method and measure metrics. Next, in Section 5 we present the evaluation results and the discussions of our main findings. In Section 6, we briefly summarize the related work and we conclude the paper and discuss future work in Section 7.

2 Problem Definition

In this paper, we focus on solving the problem: Given a user’s topic query and her social network information, who is the most *appropriate expert* for collaboration?

Appropriate experts are defined as those experts with a certain level of expertise, as well as a high possibility of collaborating with the user. We consequently consider two factors of an expert candidate, i.e. expert authority and closeness to the user. We present detailed definitions of them in following subsections.

2.1 Expert Authority

Expert authority represents an expert's knowledge on some specific topic, and is often estimated by mining related corpus or documents, which hold evidences of expert candidates' expertise. In this paper, we use a generative probabilistic language model to calculate experts' authority to a given query, i.e. we use probability $P(c | q)$ to represent the authority of an expert candidate c to a give topic query q . According to Bayesian theorem, we have,

$$P(c | q) = \frac{P(q | c) \times P(c)}{P(q)}, \quad (1)$$

where $P(c)$ is the prior probability of a candidate being an expert. $P(q)$ is the probability of query q to be emerged. $P(q | c)$ is the probability of generating query q given an expert candidate c .

2.2 Closeness

Closeness models the social distance between an expert candidate and the user. In different scenarios, it can be defined in various forms (e.g., friendship, similarity between people, even distance in geography, etc.).

Here, we extract closeness from a user's social network information. We define a user's global social network by using the "small world phenomenon" theory in social network analysis [15]. User u_0 is the center of the social graph, and her personal network composes the level-1 network. Friend nodes of level-1 nodes are added to form the level-2 nodes. We extend a user's network to level-6 to obtain u_0 's global social network. A directed graph $G_{u_0} = \langle V, E \rangle$ represents user u_0 's global social network. $u \in V$ denotes an individual node in the social network, and suppose $|V| = n$. Edge $e_{ij} \in E$ is the connection from node i to j . $w(e_{ij})$ denotes the weighted value on edge e_{ij} indicating how close nodes i and j are. $w(e_{ij})$ can be estimated using various formulations in different scenarios. Let function $r(u_i, u_j)$ represent the degree of closeness between the nodes i and j ; $Adj(u_i)$ is the set of nodes neighbor to node u_i . We define

$$r(u_i, u_j) = \begin{cases} w(e_{ij}), & u_j \in Adj(u_i) \\ 0, & u_j \notin Adj(u_i) \end{cases},$$

$r(u_i, u_j)$ is normalized so that $\sum_{0 \leq j \leq n-1} r(u_i, u_j) = 1$.

To further represent the degree of closeness between any node in u_0 's global network and u_0 , we estimate the closeness score by using the Random Walk with Restart (RWR) model [16]. Previous work [10, 12, 14] in the CSCW field used traditional social network analysis techniques (e.g., shortest path, social radius, maximum flow,

etc.) to model the closeness among people. These models neglected the global structure of the social graph and multi-facet relationship between two nodes. In contrast, the relevance score defined by RWR not only captures the global structure of the graph [9] but also captures the multi-facet relationship between nodes [17]. The detail of calculating closeness score using RWR is presented in Sec. 4.4. Here, let r_i denote closeness score between u_i and user u_0 .

3 Models

We present two baseline models, i.e. FOM and LCM, and our proposed model FRM in this section. These models combine the above two factors in three different ways. Let A_q represent the nodes list ranked by Eq. 1 to query q ; and C_{u_0} the nodes list ranked by r_i , the closeness score to user u_0 .

3.1 Filtering Out Model (FOM)

A naive approach to combine these two factors is to consider them step by step. We propose two sub-models applying different sequences of considering expert authority factor (A) and closeness factor (C).

Filtering Out Model I (C-A). We first select top K individual nodes in C_{u_0} as expert candidates, then rank them using the probabilistic language model of expert finding. We denote this model as FOM I.

Filtering Out Model II (A-C). Correspondingly, we can also select top K users in A_q as expert candidates at first, then rank them according to their closeness scores to the user. We denote this model as FOM II.

Note that filtering out model FOM I is similar to those models used by researchers in the CSCW field, while FOM II is a complementary method to FOM I.

3.2 Linear Combination Model (LCM)

Another basic approach is to use linear combination. We treat these two different factors as separate scores, i.e. expert score and closeness score. To avoid the potential bias to either score, we simply use the inverse of ranks [8] of each candidate c_i in nodes set A_q and C_{u_0} instead,

$$S(c_i, q) = \lambda \times \frac{1}{rank_{C_{u_0}}(c_i)} + (1 - \lambda) \times \frac{1}{rank_{A_q}(c_i)},$$

where λ is a weighting parameter which describes to what extent we bias towards the closeness between expert candidates and the user in the expert finding process. $rank_{C_{u_0}}(c_i)$ denotes the rank position of c_i in the set C_{u_0} , while $rank_{A_q}(c_i)$ the rank position of c_i in set A_q .

3.3 Friend Recommendation Model (FRM)

The basic idea of FRM is to model the behavior how we utilize our social networks to find experts on some topic in our daily life. When looking for someone who has expertise on some specific topic for collaboration, people usually turn to their social networks (e.g., friendship, collegueship, etc.). A typical process is like: we first consider our “friends” in our personal networks as candidates, or further inquire them “who is an expert on topic X?” to get expert recommendations from them. Then our “friends” would consider their “friends” who might have expertise on this topic in their personal networks and recommend them to us. This recommendation process would be recursive, which means the query would pass along from our “friends” to “friends of friends” and on. The individual nodes in the network will recommend experts based on their own knowledge of their friend nodes. This knowledge consists of: (a) the knowledge of their friends’ expertise on the specific topic; and (b) the knowledge of how close their friend nodes are to them. In this paper, these two types of knowledge can be represented as two factors accordingly. Thus, the policy of each node to recommend her friends as experts is a combination of these two factors of her friend nodes in her personal network.

We thus formalize this model into two steps. In the first step, we model when a node in one’s global social network receives a query, how this node recommends experts based on her personal social network, which we name *local recommendation policy*. In the second step, we treat the recommendation process for experts in one’s global network as a *Markov recommendation process*, which is based on the local recommendation policy of each node in user’s global social network obtained in step 1. We describe them in detail in the following subsections respectively.

Local Recommendation Policy

Our proposed expert finding model is based on the recommendations of friend nodes in one’s social network. We model this problem in a natural and heuristic way within probabilistic framework. The problem can be stated as: Given a topic query q and a being queried node u_i , what is the probability of an expert candidate u_j ($u_j \in Adj(u_i)$) within u_i ’s personal social network being recommended by u_i . For each expert candidate u_j ($u_j \in Adj(u_i)$), we estimate $P(u_j|u_i, q)$, supposing that query q and node u_i are independent,

$$P(u_j|u_i, q) \propto P(u_j|u_i) \times P(u_j|q), \quad (2)$$

$P(u_j|q)$ can be calculated using Eq. 1. $P(u_j|u_i)$ represents the probability to what extent one person will recommend her “friends” based on the closeness between them. We simply define,

$$P(u_j|u_i) = \frac{r(u_i, u_j)}{\sum_{0 \leq j \leq n-1} r(u_i, u_j)}.$$

To a certain node u_i , for a specific query q , we attain a local recommendation policy by calculating the probability of each neighbor friend node $u_j \in Adj(u_i)$.

Markov Recommendation Process

As for user u_0 , the recommendation process will start from her neighbor friend nodes (level-1) and pass to her extended social network (level-2 to 6). We view this process as a Markov Process. FRM models friend recommendations as a stochastic process by treating each node’s recommendation as a state transition in Markov Chain. Each recommendation can be viewed as a state transition, which leads from one node to another with a certain probability. We use the Random Walk with Restart (RWR) to model the recommendation process occurred in user’s global social network. RWR is a technique that can be utilized to estimate the relevance score between two nodes in a weighted graph [16]. It is defined as

$$P_i = (1 - \alpha)A^T P_i + \alpha e_i, \tag{3}$$

where P_i is a $n \times 1$ ranking vector, P_{ij} is the relevance score of node j with respect to node i . A is an irreducible and aperiodic stochastic matrix which guarantees the convergence of random walk and get a stationary distribution. α ($0 \leq \alpha \leq 1$) is a restart probability which controls to what extent after each iteration the computation bias towards user u_i . e_i is a $n \times 1$ personalized vector in which i th element is 1 and others are 0. The P_{ij} in final solution P_i to Eq. 3 means the steady state probability of reaching node j from node i .

The transition probability distribution of every node in the social graph is its local recommendation policy distribution. For each node u_i in the global social network of user u_0 , it can get a local recommendation policy using Eq. 2 in step 1. Thus, we have a $n \times n$ transition probability matrix A . For $\forall A_{ij} \in A$, let

$$A_{ij} = P(u_j | u_i, q).$$

A_i is normalized so that $\sum_j A_{ij} = 1$ and augmented to attain an irreducible and aperiodic matrix A' ,

$$A' = (1 - d)\frac{E}{n} + dA^T,$$

where E is ee^T (e is a column vector of all 1’s) and thus E is a $n \times n$ square matrix of all 1’s. $1/n$ is the probability of randomly recommending to a particular node in the global social network. n is the total number of nodes in the social network graph. d is the damping factor which we fix to 0.95 in the remaining experiments. We substitute A' into Eq. 3, after i th recommendation we get,

$$P_0^{i+1} = (1 - \alpha)A' P_0^i + \alpha e_0, \tag{4}$$

where $P_0 = \{A_{00}, A_{01}, \dots, A_{0n-1}\}^T$ is the initial probability distribution with respect to u_0 . e_0 is a $n \times 1$ personalized vector whose 0th element is 1 and others are 0.

To attain the solution vector P_0' of Eq. 4, we use the power iteration method which produces the principal eigenvector with the eigenvalue of 1. FRM ranks expert candidate u_i by P_{0i}' .

4 Experiment Settings

4.1 Scenario and Data Set

Scenario: *In the academic field, researcher often has an information need for knowing who is an authority on some research field, and similarly, she also often looks for some experts as coauthors for her next paper.*

Coauthoring is one kind of collaboration. If we only consider the absolute expert authority of expert candidates, it would make the coauthoring cooperation hard to execute. Hence, we use this scenario as an experiment setting to evaluate our models and see how they work.

Data Set: DBLP¹ (Digital Bibliography & Library Project) is a computer science bibliography website. One disadvantage of DBLP data is that it only provides the title of each article without the abstract and index terms, which lead to the shortage of evidences for expert candidates' expertise. Therefore, we also fetched abstracts from other digital libraries (e.g., ACM, IEEE, Springer, etc.), when we crawled articles from DBLP, to expand the article information so as to provide abundant evidences for experts' expertise. For building a more concentrative social network of a researcher, we only crawled those articles published in conferences WWW, SIGIR, SIGKDD and CIKM, which are regarded as devoting to the same research fields, during years 2004~2009. We split them into two parts respectively: (a) training data (articles published during years 2004~2008), including 3180 articles and 5647 expert candidates, used for constructing users' social networks, extracting evidences for experts' expertise and calculating expert score for each expert candidate; and (b) testing data (articles published in year 2009), including 640 articles and 1673 expert candidates, used for extracting the information of testing users and providing the ground truth of coauthors selected by users in order to test the models' performances. For all articles in training and testing data sets, we remove stop words and perform stemming for all words for further experiments. We denote the training set and testing set with D_{train} and D_{test} respectively.

4.2 Modeling Authority

We use the second probabilistic model proposed by Balog et al. [1] to estimate a candidate's probability to be an expert on a given query. For a given topic query q and an expert candidate c , we estimate $P(c | q)$ by using the following formula,

$$P(c | q) \propto \sum_d \left(\prod_{t \in q} ((1 - \lambda) P(t | d) + \lambda P(t))^{n(t,q)} \right) \times f(d, c), \quad (5)$$

where t represents a term, and the query is represented by a set of terms. $n(t, q)$ is the number of times t occurs in q . λ is a smoothing parameter, set to 0.5 in the following experimental evaluations. $f(d, c)$ is the association of document d and expert candidate c . In our approach, let $[a_{first}, a_{second}, a_{third}, \dots, a_{last}]$ denote the

¹ <http://www.informatik.uni-trier.de/~ley/db/>

author signatures order of paper d . Expert scores of corresponding authors are defined as $[5, 3, 2, \dots, 1]$. The scores of other authors are all set to 1. We define $f(d, c)$ as follow,

$$f(d, c) = \frac{S_a(o_d(c))}{\sum_{d' \in D_{train}} S_a(o_{d'}(c))},$$

where $o_d(u_i)$ stands for the place of u_i of in the author signatures order in paper d . Function S_a represents the expert score of author in the specific place (e.g. $S_a(a_{first}) = 5$).

4.3 Modeling Closeness

We extract the user's co-authorship network from D_{train} and define closeness between authors according to the history of their co-authorship. We use the number of papers they coauthored and their appearance orders in author signatures as features, and empirically define the relation score between two authors as $S_r(a_{first}, a_{second}) = S_r(a_{second}, a_{first}) = 5$; $S_r(a_{first}, a_{third}) = S_r(a_{third}, a_{first}) = 3$. Other scores are all set to 1. After iterating all the papers in D_{train} , we obtain a closeness score between any two authors u_i and u_j . Hence, we define,

$$r(u_i, u_j) = \begin{cases} \frac{\sum_{d \in D_{train}} S_r(o_d(u_i), o_d(u_j))}{\sum_{u_j} \sum_{d \in D_{train}} S_r(o_d(u_i), o_d(u_j))}, & u_j \in Adj(u_i) \\ 0, & u_j \notin Adj(u_i) \end{cases}.$$

We use RWR to calculate the closeness score between any node in u_0 's global social network and u_0 . For $\forall A_{ij} \in A$, let $A_{ij} = r(u_i, u_j)$. After iterations by using Eq. 4, we get the stationary closeness score vector P_0 with respect to u_0 . Let $r_i = P_{0i}$ denote the closeness score between user u_i and u_0 . For the comparisons of three models, the parameters of RWR here are set to the same as those in FRM.

4.4 Evaluation Method and Metrics

An automatic method is used to evaluate our approaches in this paper. We group papers in D_{test} by authors and evaluate models' performances by simulating how a user inquires some queries. We treat the title of each paper that user authors in D_{test} as a topic query, and the coauthors appearing in the same paper as the ground truth that whom the user has chosen as coauthors. We prepare a subset from D_{test} for testing, i.e., we select top 200 users who have most published papers in D_{train} . In this case, an author has more sufficient information for providing expertise and building her personal network. There are totally 338 queries generated by these 200 users. We denote this subset as D_{200} . We adopt two metrics to evaluate the performances of above models, $P@n$, measuring the relevance of top n results in the recommendation list to a given query and $nDCG@n$, measuring the models' ranking performances. For $nDCG$, the relevance grades for authors in $[a_{first}, a_{second}, a_{third}, \dots, a_{last}]$ are $[5, 3, 2, \dots, 1]$ and those of other authors are all set to 1. The average number of authors in D_{test} is about 3.2, so we consider $P@1$, $P@3$ and $P@5$ as the precision

measure metrics. And $nDCG@2$ and $nDCG@5$ are chosen respectively to evaluate the quality of recommended expert list.

5 Experimental Results

In this section, we first inspect the performance of the traditional expert finding model (EM) using Eq. 5 for comparison and show the results in Table 1. The experimental results shown in this section are average results for all queries.

5.1 Filtering Out Model

For FOM, we present evaluation results for FOM I and FOM II. We respectively set K (described in Sec. 3.1) to 20, 50, 100, 150 and 200. The performances of these two models in $P@3$ and $nDCG@3$ are drawn in Fig. 1. From Fig. 1, for FOM I we witness as the value of K increases, the performance decreases. FOM I performs best when $K = 20$, which means that researchers seem to choose those coauthors they are familiar with, while for FOM II the performance gets better when K value increases and reaches its best when $K = 200$. It seems that although some experts have less expertise, they are still good coauthors for users to collaborate with as long as they are close to users. As the value of K becomes larger, with more expert candidates involved in consideration, FOM I tends to bias toward those authoritative experts while FOM II becomes to favor those candidates close to users. The performances of FOM I and FOM II when $K = 200$ also state that closeness is playing a more important part in this choosing coauthors scenario.

The results of FOM I and FOM II in all measure metrics when we set their best K values, respectively 20 and 200, are presented in Table 1. It is obvious that FOM I and FOM II outperform the pure expert finding model EM in all metrics, which confirms that social network information is helpful in finding appropriate experts for collaboration.

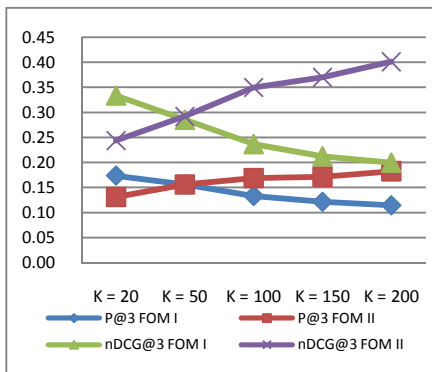


Fig. 1. The performances of FOM I and FOM II varying K value

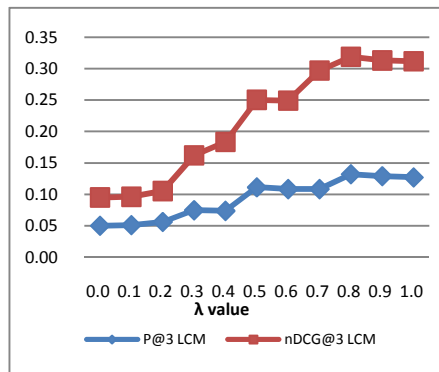


Fig. 2. The performance of LCM varying lambda value

5.2 Linear Combination Model

For LCM, we respectively evaluate the performance of the model by changing the value of closeness weighting parameter λ from 0.0 to 1.0 with increment of 0.1. Fig. 2 shows the performance of LCM when λ value varies.

From Fig. 2, we note that when λ value is low which indicates expert authority factor dominates, the performance of LCM is poor. As λ value increases, LCM performs better and reaches its best performance when $\lambda = 0.8$, which is consistent with previous finding in Sec. 5.1, that is when choosing coauthors, people are more likely to pick those candidates who are closer to them. For the remaining comparisons, we empirically set $\lambda = 0.8$. We present evaluation results in all metrics in Table 1. It shows that although LCM outperforms EM, it is dominated by FOM I and FOM II. Compared to others, LCM seems not suit this scenario. Although other models are more heuristic, filtering out technique is suitable in finding appropriate experts for collaboration.

5.3 Friend Recommendation Model

We first inspect the influence of the iteration number to the performance of FRM and when the model comes to converge. We tune the iteration number by fixing the restart parameter $\alpha = 0.2$. The results are shown in Fig. 3. Note that when the iteration number reaches to 2, the performance of FRM improves significantly. Since friend recommendation is modeled as a Markov Process, one iteration can be viewed as a propagation along the network to the next level. So, we can tell that only relying on the recommendations of friends in one’s personal network is not sufficient. As the iteration number increases, precision metric is slightly improved while $nDCG$ obtains more improvement. This indicates that when the recommendation process involves more nodes in the user’s global social network, FRM can generate a more accurate ranked expert list for the user. The global structure of social networks benefits finding appropriate experts for collaboration.

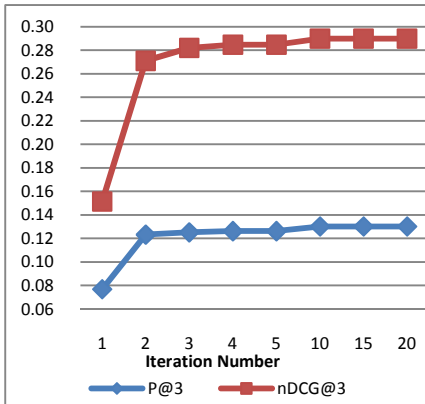


Fig. 3. The performance of FRM varying iteration number

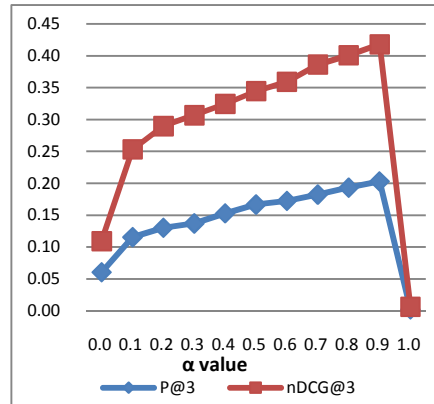


Fig. 4. The performance of FRM varying alpha value

Then we evaluate the influence of restart parameter α to our model by changing it from 0.0 to 1.0 with increment of 0.1. The iteration number is fixed to 15. Fig. 4 shows the results of $P@3$ and $nDCG@3$. We observe that the performance of FRM increases as the α value increases. When $\alpha = 1.0$, performance drops down steeply to 0, which can be explained by the fact that after each iteration P_0 is reset to e_0 then only the user herself is recommended. We further increase α from 0.90 to 0.99 with increment of 0.01 and witness that the performance of FRM still increases. Note that restart parameter α means the probability of returning to the user. A larger α value means FRM favors more to those nodes close to the user, resulting in a higher personalization for the user. In finding coauthors scenario, it shows that people prefer those nodes near them, especially those in their personal networks. However, we should keep in mind that the best α value might be different in other scenarios on other types of data sets, which we leave as future work.

The results of FRM in all measure metrics are presented in Table 1 by setting iteration number to 15 and α value to 0.9. Table 1 shows the experimental results of all models while they are in their almost best performances. With a quick scan of Table 1, we can note that for all metrics, FRM outperforms other models.

Table 1. Experimental results for all models using D_{200}

	$P@1$	$P@3$	$P@5$	$nDCG@3$	$nDCG@5$
EM	0.0503	0.0503	0.0379	0.0828	0.0949
FOM I	0.2012	0.1736	0.1420	0.3333	0.3749
FOM II	0.2041	0.1824	0.1450	0.4011	0.4197
LCM	0.1183	0.1322	0.1166	0.3188	0.3569
FRM	0.2574	0.2032	0.1509	0.4181	0.4366

6 Related Work

Many researches have been conducted in integrating social network information into expert finding. In the IR field, researchers utilize social networks as extra information sources to improve the authority of experts. Campbell et al. [5] have investigated the issue of expert finding in an email network. Link structure-based algorithms, such as PageRank [4] and HITS [11], can be used to analyze the relationships in a social network, which can improve the performance of expert finding. Jing Zhang et al. [18] proposed a propagation based approach in order to improve the accuracy of expert finding by taking relationship of each candidate into consideration. In the CSCW field, by using measurements of social distance in traditional social network analysis (e.g., shortest path, social radius, etc.), researchers took social relationships into the process of finding appropriate experts. ReferralWeb [10] constrained expert candidates in the user's social network by social radius and ranked them by their expertise on some query. Expertise Recommender [14] utilized the user's social network information to filter out those experts not in her social network. SmallBlue [12] reached

out to the user's social network within six degree to find and access expertise and information, and showed the user the shortest social path from her to those experts. Although research work in the CSCW field considered how to combine these two factors to find appropriate experts, their proposed approaches modeled closeness and combined related factors in simple ways as basic models FOM and LCM.

7 Conclusion and Future Work

This paper focuses on solving the problem of finding appropriate experts for user to efficiently collaborate with. We formally define two related factors in this task, i.e. expert authority and closeness to the user. And we consequently presented two baseline models, FOM and LCM and a novel model FRM. Evaluations of these models were performed on DBLP data set in a scenario that a user looks for coauthors in academic domain. By systematically evaluating these three models and through the comparisons, we found that: (a) closeness factor is important in choosing appropriate experts for collaboration; (b) RWR can effectively model friend recommendation process; and (c) the novel model FRM outperforms baseline models on all measure metrics in this experimental setting. In the future, we want to explore different types of closeness definitions by other social networks information (e.g., Facebook, LinkedIn, Twitter, etc.), and inspect these models further to find out how the closeness factor affects users when choosing appropriate experts for different kinds of collaborations, especially how FRM will perform.

References

1. Balog, K., Azzopardi, L., Rijke, M.D.: Formal models for expert finding in enterprise corpora. In: SIGIR (2006)
2. Bao, S., Duan, H., Zhou, Q., Xiong, M., Cao, Y., Yu, Y.: A Probabilistic Model for Fine-Grained Expert Search. In: ACL (2008)
3. Borgatti, S.P., Cross, R.: A Relational View of Information Seeking and Learning in Social Networks. *Management Science* (2003)
4. Brin, S., Page, L.: The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks* (1998)
5. Campbell, C.S., Maglio, P.P., Cozzi, A., Dom, B.: Expertise identification using email communications. In: CIKM (2003)
6. Cross, R., Parker, A.: *The Hidden Power of Social Networks*. Harvard Business School Press, Boston (2004)
7. Fang, H., Zhai, C.: Probabilistic Models for Expert Finding. In: Amati, G., Carpineto, C., Romano, G. (eds.) *ECIR 2007*. LNCS, vol. 4425, pp. 418–430. Springer, Heidelberg (2007)
8. Fox, E.A., Shaw, J.A.: Combination of Multiple Searches. In: TREC (1993)
9. He, J., Li, M., Zhang, H., Tong, H., Zhang, C.: Manifold-ranking based image retrieval. *ACM Multimedia* (2004)
10. Kautz, H.A., Selman, B., Shah, M.A.: Referral Web: Combining Social Networks and Collaborative Filtering. *Commun. ACM* (1997)
11. Kleinberg, J.M.: Authoritative Sources in a Hyperlinked Environment. In: SODA (1998)

12. Lin, C., Cao, N., Liu, S., Papadimitriou, S., Sun, J., Yan, X.: SmallBlue: Social Network Analysis for Expertise Search and Collective Intelligence. In: ICDE (2009)
13. Macdonald, C., Ounis, I.: Voting for candidates: adapting data fusion techniques for an expert search task. In: CIKM (2006)
14. McDonald, D.W., Ackerman, M.S.: Expertise recommender: a flexible recommendation system and architecture. In: CSCW (2000)
15. Milgram, S.: The small world problem. *Psychology Today* (1967)
16. Tong, H., Faloutsos, C., Pan, J.: Random walk with restart: fast solutions and applications. *Knowl. Inf. Syst.* (2008)
17. Tong, H., Faloutsos, C.: Center-piece subgraphs: problem definition and fast solutions. In: KDD (2006)
18. Zhang, J., Tang, J., Li, J.: Expert Finding in a Social Network. In: Kotagiri, R., Radha Krishna, P., Mohania, M., Nantajeewarawat, E. (eds.) DASFAA 2007. LNCS, vol. 4443, pp. 1066–1069. Springer, Heidelberg (2007)

Location Privacy Protection in the Presence of Users' Preferences*

Weiwei Ni, Jinwang Zheng, Zhihong Chong**, Shan Lu, and Lei Hu

Southeast University, Nanjing, China
{wni, zhengjw, chongzhihong, shine, hulei}@seu.edu.cn

Abstract. Location privacy receives considerable attentions in emerging location based services. Most current practice however fails to incorporate users' preferences. In this paper, we propose a privacy protection solution to allow users' preferences in the fundamental query of k nearest neighbors. Particularly, users are permitted to choose privacy preferences by specifying minimum inferred region. By leveraging Hilbert curve based transformation, the additional workload from users' preferences is alleviated. What's more, this transformation reduces time-expensive region queries in two dimensional space to range ones in one dimensional space. Therefore, the time efficiency, as well as communication efficiency, is greatly improved due to clustering properties of Hilbert curve. The empirical studies demonstrate our implementation delivers both flexibility for users' preferences and scalability for time and communication costs.

1 Introduction

Location-based services(LBS in short) in parallel with various applications of location-aware devices(e.g.,GPS devices) have gained tremendous popularity, spanning a wide spectrum from sensor networks [1] over online mapping services [2] to geospatial information systems [3], to name a few. Queries in LBS usually include enquiring locations and other information pertaining to so called points-of-interest(POI in short). Most of these spatial queries involve the query of the k nearest neighbors(k NN in short) such as the k nearest restaurants for a traveller [4]. While LBS enables a wide spectrum of location-based applications, they indeed threaten users' privacy as they force users to disclose their locations. For example, the query of the nearest gas station by a traveller on his trip demands the disclosure of his current location. Such location leakage may hinder applications susceptible to users' privacy. In this sense, queries over public data introduce additional challenges when users' privacy protection is concerned.

Existing solutions to avoiding location exposure fall into three categories, namely, spatial cloaking [6][7][8], space transformation [9][10] or location obstruction [11]. The common of these schemes trade-off among query performance,

* This work is supported by the National Natural Science Foundation of China under grant No. 61003057 and No. 60973023.

** This author is the corresponding author.

protection strength and query accuracy. Nevertheless, they in common fail to incorporate users' preferences into consideration in face of system's efficiency and scalability; different users with different concerns are imposed on the same privacy protection strength. Considering the same query of the nearest gas station initiated by Bob in two different scenarios where Bob is located near a university and where near a betting office, Bob prefers to be more protected when he dislikes someone to infer the possibility of his appearance in the betting office, while allowing weak protection strength even with the possibility of being inferred that he is near the university. This requirement motivates us to emphasize the importance of users' preferences.

Given the requirement that users' locations are not exposed to the server, the server delimits a region, denoted as *RCA*, that contains queried POIs for users and that guarantees users' privacy [6]. To state briefly, except the users' locations, all other information, including the *RCA* and its creation procedure, is public to attackers. This exposure leaves attackers the clues of bounding the range of the users' possible locations [6][11]. The minimum bound of the range derived by attackers is called the minimum inferred region, referred to as *MIR* in most work [7]. In this paper, we follow existing work to use *MIR* as the measurement of privacy strength. Meanwhile, we denote users' preferences using *MIR* and allow users to specify their preferences in queries. As a result, the two-fold roles of *RCA* due to its strong relation with user-specified *MIR* at user end invalidate existing solutions to creating *RCA* at server end where *MIR* is prior fixed at server end for all kinds of users, let alone heavy burden at the server end. In this paper, we propose a strategy, called *HilAnchor*, in a users' preferences-driven fashion, while alleviating server's burden. In particular, the *RCA* is created at client end by initiating a false query, promising user specified *MIR*; the overhead workload from *RCA* determined at client end can be alleviated efficiently by specialized data compression.

HilAnchor, while sharing the implementation of false query in location obstruction based solutions, makes itself distinguishable in taking users' preferences into account by permitting the user to choose the *MIR*. A query in *HilAnchor* consists of two rounds; at the first round, it generates a square from the returned answers to cover both the targeted POIs and the users' preferred inferred region; at the second round, users resend the square and receive all POIs in it. The targeted POIs are immediately pinpointed at client end. Paralleling with spatial transformation, the overhead workload from transmitting and processing square is alleviated by leveraging Hilbert curve to transform two dimensional space to one dimensional space. The clustering properties of Hilbert curve enable to encode the square into discrete ranges at client sides so that the communication between two ends is compressed. Meanwhile, the time-expensive region query is converted into range query at the server and B^+ -tree index structure further improves the time-efficiency. Our contributions are summarized as follows.

1. *HilAnchor* permits users to specify their preferred minimum inferred region. The answer is pinpointed from the candidates generated in terms of the specified *MIR*.

2. The location-based service does not compromise on its scalability when it provides privacy protection. The enhanced version *HilAnchor*⁺ of *HilAnchor* enables a thin server by pushing most workload down to client ends via Hilbert coding.
3. *HilAnchor*⁺ delivers accurate answers, excluding false locations included in traditional transformation-based methods.

1.1 Prior Work

Motivated by the privacy threats of location-detection devices, there have been a plethora of techniques to deal with these kinds of location-based privacy protection such as location obstruction [11], space transformation [10] [12] and spatial cloaking [6] [7] [13]. In location obstruction [11], a query along with a false location is first sent to the database server, and the database server keeps sending back the list of nearest objects to the reported false location until the received objects satisfies privacy and quality requirements. The lengthy interaction between two ends makes it difficult to allow users' preferences. In space transformation [10] [12], the locations of both data and queries are converted into another space through a trusted third party. The transformation maintains the spatial relationship among the data and queries to provide accuracy. This kind of solutions always have at least one of the following shortcomings: (1) Fall short in offering accuracy guarantee. Query result may contain false hints. (2) User's location privacy relies wholly on the security of transformation key and does not afford user's preference at all. In spatial cloaking [6] [7] [13], a privacy-aware query processor is embedded in the database server side to deal with the cloaked spatial area received either from a querying user [13] or from a trusted third party [6] [7]. Cloaking based solutions provide user preference to location privacy by transmitting a user defined profile including user location and expected minimum inferred area to the anonymizer. Anonymizer then expands user location into a cloaked region with expected area to act as final *MIR*, and sends the region to the server for retrieving candidate answers. In this way, complex server-side query processing is needed to determine *RCA* in terms of the given minimum inferred region, which affects system's performance seriously. It provides user preference in a brute-force way at cost of complex server-side query processing and poor scalability.

The rest of the paper is organized as follows. Section 2 discusses the user's option-driven framework *HilAnchor*. The thin-server enhanced version *HilAnchor*⁺ based on Hilbert transformation is presented in Section 3. Section 4 covers the experimental results of *HilAnchor* and *HilAnchor*⁺. Finally, Section 5 concludes and identifies research directions.

2 HilAnchor

We next start with the framework of *HilAnchor* to illustrate how *HilAnchor* works and then show the details of creating *RCA* as the key step of *HilAnchor* framework.

2.1 Framework of HilAnchor

HilAnchor processes a k NN query in two rounds, detailed in Fig. 1. In the first round, a user sends a false point p' of point p , called an anchor of point p , to server and receives k nearest neighbor answers, denoted as $NN(p')$, in terms of p' . In the second round, the client sends back RCA created from the returned answers. The server returns all POIs located inside the RCA . Finally, the actual result is pinpointed at the client end.

During these two rounds, RCA needs to meet two-fold requirements. First, it forces the server to return all POIs within it, including the target POIs. Second, it promises users' preferences to MIR . The difficulty of RCA creation stems from the latter requirement; it is possible for adversaries to shrink the inferred region within a big RCA , invalidating its privacy protection. This observation contradicts usual institutions of enlarging RCA in a brute-force way, let alone the increasing cost with large RCA . Therefore, the realization of *HilAnchor* framework becomes difficult when it aims to allow for user-specified MIR .

2.2 RCA Creation

For clarity reason of description, the two dimensional space we discuss is defined with Euclidean distance $d(\cdot)$ and the data set T contains all POIs at server. Recall the client end creates the RCA from the returned POIs for the anchor point. The creation is detailed into two phases as shown in Fig. 2. At the first phase, an initial region of a circle is created to cover the target POIs. At the second phase, the initial region is blurred to meet the MIR requirement.

Initial Region Creation. For given point p , p' is its anchor. Point $o \in NN(p')$ is the farthest POI returned by the server to p . The client builds the initial region by creating a circle $CC(p, p')$ shown in Fig. 2, centered at point p with radius from the center to the point o .

Theorem 1. *Given p , the area of the initial region is the minimum area that covers the k nearest POIs to p for any data set T .*

Proof. From the process of $CC(p, p')$ creation, it can be deduced that the initial region $CC(p, p')$ must cover $NN(p)$ and the maximum distance between p and

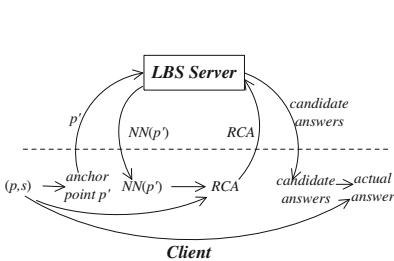


Fig. 1. Framework of HilAnchor

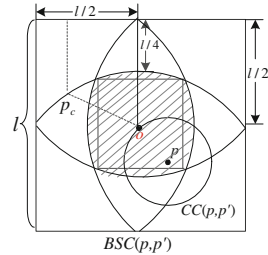


Fig. 2. Inferring region of RCA

POIs in $NN(p)$ is $d(p, o)$. Assume $o' \in T$ is an element in $NN(p)$ located outside the initial region $CC(p, p')$. For o locates just on the boundary of the circle denoted in Fig. 2, it must have $d(p, o') > d(p, o)$, which contradicts with the assumption that o' belongs to the POI set $NN(p)$ ‡

Theorem 1 indicates that any RCA must contain the initial region $CC(p, p')$ such that the target POIs are returned to users in the second round. It is naturally assumed that point o and the creation of RCA are public to adversaries. This assumption facilitates the inference of user’s location p , i.e., user’s location p is determined at the center of $CC(p, p')$. Hence, initial region $CC(p, p')$ fails to guarantee the specified MIR if it directly serves as RCA . Our strategy is to expand $CC(p, p')$ to a square, denoted as $BSC(p, p')$ in Fig. 2. To be precisely, o is set as the center of the square that covers the initial circle $CC(p, p')$. For $BSC(p, p')$, the following lemma is immediately obtained without proof.

Lemma 1. *Given the exposure of $BSC(p, p')$ and its creation algorithm, there are infinite number of $CC(p, p')$ that correspond to the $BSC(p, p')$, i.e., the probability of inferring $CC(p, p')$ for a query in a random way is zero.*

Further, the following theorem shows how $BSC(p, p')$ leads to MIR guarantee.

Theorem 2. *For a specified area s , if the side length of $BSC(p, p')$ is no less than $\ell_{min} = \max\{2.391\sqrt{s}, (\sqrt{2} + 2)d(p, o)\}$, the area of MIR is no less than s .*

Proof. As shown in Lemma 1 that given point o on the boundary of circle $CC(p, p')$, there are infinite number of circles centered at point p and containing o . It can be inferred that the centers for these possible circles must have equidistance from p to o and from p to the side of the concentric square inscribed to the circle within $BSC(p, p')$. For a given side, these centers form a parabola with point o as its focus and the side as an alignment. MIR must be embodied by four parabolas *w.r.t* four sides of $BSC(p, p')$, respectively, as the shaded shown in Fig. 2. The parabolic equations can be normalized as $y^2 = \ell x$, ℓ is the side length of $BSC(p, p')$. The inferred region of the shaded can be calculated, $\psi = 8 \int_0^{\frac{3-2\sqrt{2}}{4}\ell} \sqrt{\ell x} dx + (\sqrt{2} - 1)^2 \ell^2 \approx 0.175\ell^2$. It requires that $\psi \geq s$ for location privacy guarantee. Thus, we have $\ell \geq 2.391\sqrt{s}$. Meanwhile, to guarantee that $BSC(p, p')$ covers the initial region $CC(p, p')$, it requires $\ell \geq (\sqrt{2} + 2)d(p, o)$ therefore $\ell_{min} = \max\{2.391\sqrt{s}, (\sqrt{2} + 2)d(p, o)\}$. ‡

Algorithm 1. HilAnchor-Client(p, s)

// p is the user location and s the area of MIR

- 1: create the initial region CC ;
 - 2: expand CC to BSC ; //the side length of BSC satisfies Theorem 2
 - 3: send BSC back to the server;
 - 4: pinpoint answers from the returned;
-

The Algorithm 1 running at client ends presents details of process of our model at client. The square $BSC(p, p')$ serves as RCA and is sent back to the server

in line 3. The server returns all POIs inside $BSC(p, p')$ as candidate answers. Finally, the answers are pinpointed at client side.

The correctness of Algorithm 1 is guaranteed by Theorem 2. The users' preferences for protection strength are realized by parameter s . The problem arises that it is possible that the area of BSC created by $HilAnchor-Client(p, s)$ is not just closely larger than that of the initial region. The number of candidate POIs within the expanded BSC may be large, introducing heavy time cost overhead especially in two dimensional space at server side, as well as communication cost when they are sent back to client ends. To tackle this problem, we leverage the clustering property of Hilbert curve to devise the enhanced version $HilAnchor^+$ of $HilAnchor$ in the following section.

3 $HilAnchor^+$

Before delving into the details of how Hilbert curve enhances the power of $HilAnchor$, we sketch the paradigm of $HilAnchor^+$. The main differences between the two versions are: 1) the POIs at the server are encoded into Hilbert indexes(a.k.a. Hilbert cells) and 2) all queries to the server are correspondingly encoded into Hilbert indexes and all returns from server are decoded at client ends. Further, methods of processing encoded queries on encoded data are presented.

3.1 Hilbert Encoding under Privacy Constraint

To alleviate the overhead workload due to users' preferences, we explore Hilbert curve. In particular, a square, say $BSC(p, p')$ for example, in 2-D space is transformed to Hilbert indexes. Through this transformation, the time-expensive region query in 2-D space can be converted into range query in 1-D space. The Hilbert curve is used under privacy constraint. Both the service provider and the client users do not know parameters of the curve for privacy protection reason. The encoding and decoding functions at the users' ends are embedded in tamper-resistant devices without the third party's intervene.

Similar to work in [10], our enhancement requires an offline space encoding phase carried by a trusted third party. The curve parameters of SDK is determined and the whole space \mathfrak{R} is encoded into 2^{2N} Hilbert cells by the specified Hilbert curve H_2^N . As a result, all POIs are encoded into corresponding Hilbert values and stored in a look-up table LUT .

Correspondingly, the client submits the Hilbert encode $\tilde{h}(p')$ of the anchor p' to the server rather than anchor p' itself. The server returns the k nearest Hilbert values for POIs to $\tilde{h}(p')$. Subsequently, the client decodes the returned values of corresponding POIs and generates $BSC(p, p')$. Now, the problem boils down to encoding BSC of a square via Hilbert curve. For square S under Hilbert curve H_2^N , its Hilbert closure $HC(S)$ is the minimum set of Hilbert cells that cover S .

It is time and communication prohibitive to directly represent BSC of square with its Hilbert closure. Therefore, we resort to compressing the Hilbert

closure of a square. The intuition is to partition the cells in a Hilbert closure into consecutive ranges. These ranges include all Hilbert cells in $HC(S)$.

For the square S , let $BC(S) \subset HC(S)$, containing all cells located at the boundary of square S . Further, the cells in $BC(S)$ can be partitioned into three types, namely called in-cells, out-cells and inner-cells.

Definition 1. For square S , cell $u \in BC(S)$ is called an in-cell of S if the Hilbert curve enters S via u . Conversely, u is called an out-cell of S if the Hilbert curve leaves S via u .

Definition 2. Hilbert Closure Range. For given square S , sort its in-cells and out-cells in ascending order, resulting in a sequence of interleaved in-cells and out-cells. Each pair of two adjacent in-cell and out-cell forms a Hilbert range. The set of such ranges is called the Hilbert closure range of square S , denoted as $HCR(S)$.

Theorem 3. Given point $q \in \mathfrak{R}$, q belongs to square S if and only if there is a Hilbert range (I_i, O_i) in $HCR(S)$ such that $I_i \leq h(q) \leq O_i$.

Proof. The curve H_2^N traverses each cell once and only once and encodes the cell on the way consecutively [18]. For $S \subset \mathfrak{R}$, the curve inside S must be partitioned into a series of continuous curve segments, which starts from an in-cell of S and ends at an out-cell of S . These curve segments correspond to $HCR(S)$. Thus, Hilbert index of any cell in $HC(S)$ must be included by one of ranges in $HCR(S)$. For q belongs to square S , it must be indexed by some cell in $HC(S)$. Therefore, $h(q)$ must be included by a range in $HCR(S)$. \square

Theorem 3 verifies that Hilbert closure range of $BSC(p, p')$ must include all Hilbert indexes of target POIs w.r.t p . Therefore, the client can retrieve Hilbert indexes of target POIs by transmitting $HCR(BSC(p, p'))$ instead of all cells in $HC(BSC(p, p'))$. This would greatly reduce the query workload at server side and the communication cost.

Algorithm 2. HeC(S)

- 1: collect the Hilbert indexes of boundary cells into BC;
 - 2: identify in-cells and out-cells in BC; //By Theorem 4
 - 3: return all values of in-cells and out-cells in ascending order as $HCR(S)$;
-

While with unknown Hilbert curve due to the protection of privacy, it is non-trivial to decide the type of a cell in $BC(S)$, even when its Hilbert index is obtained. This application constraint makes the computation of $HCR(S)$ considerably difficult. A brute-force way is to calculate all Hilbert indexes of cells in $HC(S)$ to determine the ranges at client. Although the client can encode 2-D coordinates into its Hilbert index within several steps, to deal with so many cells is time-consuming. Fortunately, the following Lemma enables to decide the types of cells efficiently.

Theorem 4. For given square S , cell $u \in BC(S)$ and its outward adjacent cell u' , if the indexes of u and u' are consecutive, u must be in-cell or out-cell of S . Specifically, the index of u is larger than that of u' , u is an in-cell, otherwise it is an out-cell.

Proof. For the curve H_2^N traverses the whole space \mathfrak{R} and encodes each cell on the way in an incrementally consecutive way. Cell u is the boundary cell of S , if the curve leaves S from u , the next cell the curve visits must be just the outward adjacent cell of u . Similar conclusion can be drawn for in-cells. \square

Therefore, only those indexes of cells in $BC(S)$ and their outward adjacencies need computing. By now, based on Theorem 4, we can present the algorithms to compress the Hilbert encode of a square. The details are presented in algorithm $HeC(S)$.

3.2 Algorithm $HilAnchor^+$

This Section presents the client-side and server-side processing of $HilAnchor^+$.

Client-side Processing. Algorithm 3 summarizes the client-side process of $HilAnchor^+$ for a user to initiate a kNN query. In Line 1, the initial region is created by one handshake with the server accompanying encoding and

Algorithm 3. $HilAnchor^+$ -Client(p,s)

p is the user location and s the area of MIR

- 1: create the initial region CC ;
 - 2: expand CC to BSC ;
 - 3: compress BSC to HCR ;
 - 4: send HCR back to the server;
 - 5: pinpoint answers from the returned from the server;
-

decoding operations at client. Subsequently, $HCR(BSC)$ is determined by analyzing $BC(BSC)$ following Theorem 4. Line 3 sends $HCR(BSC)$ back as the encoded RCA to the server. The server returns all values in LUT inside ranges of HCR . Finally, the client can pinpoint answers from returned Hilbert indexes by decoding these indexes and comparing them with p . Differing from traditional transformation-based solutions, false-hints originating by Hilbert curve transformation will not appear in $HilAnchor^+$ for the predefinition of candidate answer region. The Hilbert value indexing accurate answer can be pinpointed at client in Line 4. In our solution, function $h^{-1}()$ returns the 2-D coordinates of the input Hilbert cell's center, therefore the distance between query answer and original 2-D coordinates of targeted POI will not exceed $\frac{\sqrt{2}}{2}$ times of the Hilbert cell extent.

Server-side Processing. In our framework $HilAnchor^+$, two handshakes exist between client and server. First, client sends Hilbert value of the anchor point to the server for retrieving k nearest Hilbert values of POIs to that of the anchor. Second, the client sends the generated Hilbert candidate ranges to the server and retrieves

all POIs whose Hilbert values locate inside the ranges. Both of these queries are carried on the 1-D table *LUT*, and belong to 1-D range query. To improve time-efficiency at server side, a B^+ -tree index is constructed on the *LUT* table to accelerate range queries. Therefore, the time efficiency of such range query is $O(\log_2^n)$, the symbol n denotes the total number of POIs at the server.

4 Empirical Evaluation

We compare *HilAnchor*⁺ with spatial cloaking approach, denoted as Casper [6] and the location obstruction method, denoted as *SpaceTwist* [11]. We implemented all algorithms using C++ on Intel Xeon 2.4GHz machine. The synthetic data sets were uniformly generated with 2000 to 30000 points and real data sets NE¹ with 3000 points. The coordinates of each points are normalized to the square 2D space with extent 100000 meters. The auxiliary structures R-tree and B^+ -tree with 1KB page size are optional. As for the location parameter p' , we choose it in two steps. The first step is to decide on an anchor distance $d(p, p')$ incorporating user's preference. In a second step, the anchor p' is set to a random location at distance $d(p, p')$ from p .

Scalability performance of each method is evaluated by submitting queries from a series of clients simultaneously to the same server. In each experiment, we use a workload with M query points generated randomly from different clients and measure the value of the following performance metrics: 1) average communication cost of M queries, in numbers of TCP/IP packets as adopted in [11]; 2) total time cost at server side and 3) average time cost at client side.

4.1 Comparing with Casper

Spatial cloaking based solutions commonly adopt k -anonymity based location model. Our privacy model does not require the knowledge of all users' real-time location distribution, which is hard to grasp for mobile users and snapshot query scenarios. Hence, k -anonymity based solution Casper and our solution work in different settings and offer different kinds of privacy guarantees. For comparison purpose, we implement solution Casper that generates the cloaked region, making its area the same to the area of *MIR* setting in *HilAnchor*⁺.

Fig. 3 depicts the scalability of *Casper* vs. *HilAnchor*⁺. Their performances w.r.t. the number M of querying users are detailed in Fig. 3(a). With increasing M , Casper consumes much more time. What's more, *HilAnchor*⁺ is a clear winner in terms of time cost shown in Fig. 3(b) at server side and communication overhead shown in Fig. 3(c). Since the anchor distance is fixed, the *RCA* and *HCR* are insensitive to the data size, *HilAnchor*⁺ is scalable to the data size.

When parameter k varies, *HilAnchor*⁺ shows its advantage in the time cost shown in Fig. 4(a) on real dataset NE. Due to the fact that the compression 1-D structure *HCR* leads to the scalability to the extent of *RCA*, the cost of *HilAnchor*⁺ is nearly independent of k . Fig. 4(b) and 4(c) show the comparisons

¹ Data source, <http://www.rtreportal.org>

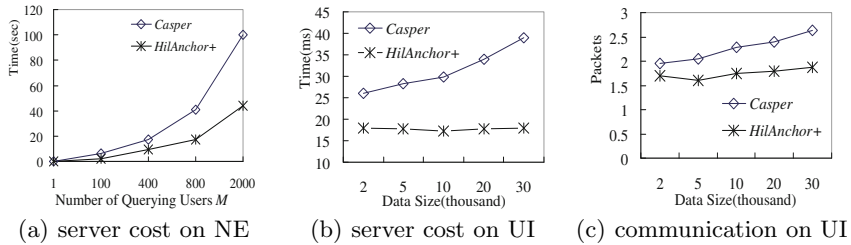


Fig. 3. Scalability $Vs M$ and Dataset Size, $k=1$, MIR is 1% of the data space

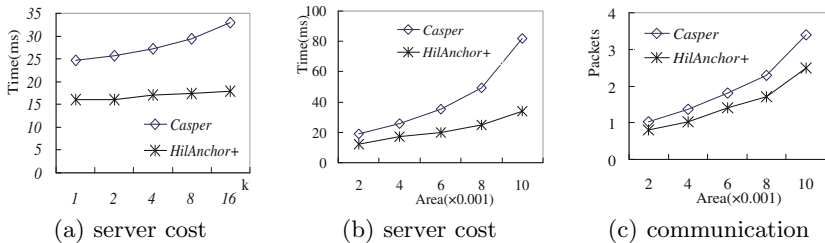


Fig. 4. Performance $Vs k$ and Area of MIR on NE dataset

of the server-side processing and communication cost in terms of MIR . The costs of both *Casper* and *HilAnchor*⁺ increase with enlarging extent of MIR . However, even for large MIR , the server-side processing time of *HilAnchor*⁺ increases quite more slowly than that of *Casper*. The underlying reason is that *Casper* falls short of allowing for constraints between RCA and MIR , and its region of RCA determined at server side expands sharply with increasing MIR .

From the above experiments, we conclude that *HilAnchor*⁺ exhibits high scalability with data size and the number of users with constraints of specified MIR .

4.2 Comparing with *SpaceTwist*

In this Section, we proceed to investigate the scalability of *HilAnchor*⁺ against *SpaceTwist*. *SpaceTwist* explores granular search strategy to retrieve data points from the server with a user-specified cell extent $\frac{\epsilon}{\sqrt{2}}$. It improves its efficiency at cost of query accuracy with error bound ϵ . We set ϵ to $\frac{\sqrt{2}}{2}$ times of Hilbert cell extent in *HilAnchor*⁺ so that they are compared with the same accuracy.

In Fig. 5(a) and Fig. 5(b), *HilAnchor*⁺ shows obvious advantage in terms of processing time for varying M , which is similar to the case of the comparison with *Casper*. Because of expensive queries in 2-D space other than 1-D query in *HilAnchor*⁺, *SpaceTwist* takes more time. Although the client-side process of *HilAnchor*⁺ is heavy, *HilAnchor*⁺ exploits only two-round handshakes rather than multi-rounds handshakes in *SpaceTwist*, which can effectively reduce the time at client waiting for response. Therefore, *HilAnchor*⁺ achieves both low

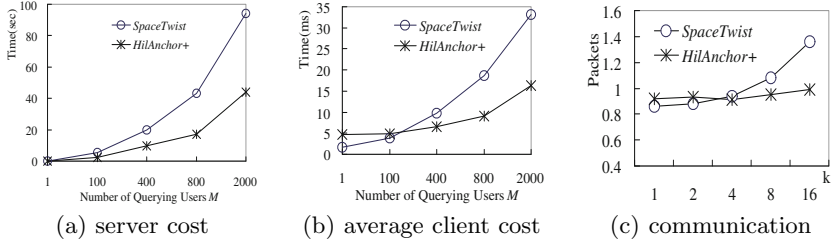


Fig. 5. Performance $Vs.$ M and k

server-side cost and low average client-side cost for a broad range of M , especially for larger M . Fig. 5(c) depict the performance with respect to parameter k . Due to its insensitivity to parameter k , $HilAnchor^+$ behaves approximately unchanged with increasing k .

Our solution shares strategies of the implementation of false query in location obstruction with $SpaceTwist$. Observe that these two solutions behave similarly in terms of client-side cost and communication overhead as shown in Fig. 6. Although a larger anchor distance in common promises a large RCA in $HilAnchor^+$ or a larger supply space in $SpaceTwist$, encoding structure HCR of $HilAnchor^+$ can compress RCA sharply. Thus, the performance gap between them is enlarged gradually with increasing anchor distance. As for server-side overhead, $SpaceTwist$ remains increasing much faster than $HilAnchor^+$. Finally, we compare the two solutions with respect to data size using synthetic UI datasets. As shown in Fig. 7 $HilAnchor^+$ scales well.

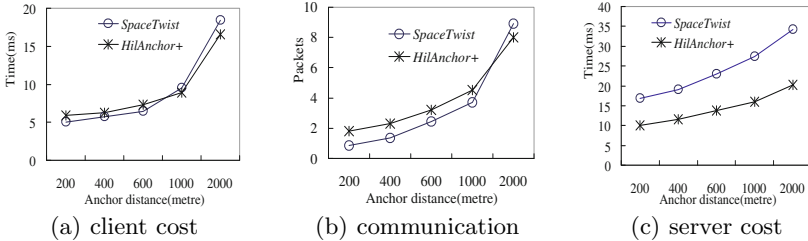


Fig. 6. Performance $Vs.$ Anchor Distance, $k=1$, MIR is 1% of the data space

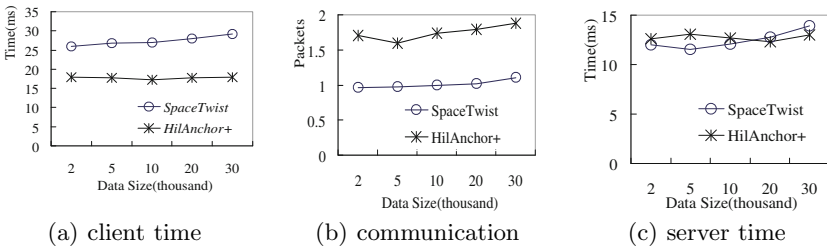


Fig. 7. Performance $Vs.$ Data Size, $k=1$, MIR is 1% of the data space, anchor distances are the same

5 Conclusion

This paper concerns the preference support for location-based service while guaranteeing user-defined minimum inferred region. Moreover, we propose a good tradeoff among location privacy preference, query accuracy and scalability performance by leveraging Hilbert curve based compression. Empirical studies with real-world and synthetic datasets demonstrate *HilAnchor*⁺ can provide flexible location privacy preference as well as accurate query results and light load on server end. It is expected to extend our propose to incorporate the influence of some non-reachable regions' existence and to support continuous queries.

References

1. Gruteser, M., Schelle, G., Jain, A., Han, R., Grunwald, D.: Privacy-aware location sensor networks. In: Proc. of the Workshop on Hot Topics in Operating Systems, HotOS (2003)
2. Beresford, A.R., Stajano, F.: Location privacy in pervasive computing. *IEEE Pervasive Computing* 2(1), 46–55 (2003)
3. Warrior, J., McHenry, E., McGee, K.: They know where you are. *IEEE Spectrum* 40(7), 20–25 (2003)
4. Roussopoulos, N., Kelly, S., Vincent, F.: Nearest neighbor queries. In: SIGMOD, pp. 71–79 (1995)
5. Bettini, C., Wang, X.S., Jajodia, S.: Protecting privacy against location-based personal identification. In: Jonker, W., Petković, M. (eds.) *SDM 2005*. LNCS, vol. 3674, pp. 185–199. Springer, Heidelberg (2005)
6. Mokbel, M.F., Chow, C.-Y., Aref, W.G.: The new casper: Query processing for location services without compromising privacy. In: *VLDB*, pp. 763–774 (2006)
7. Kalnis, P., Ghinita, G., Papadias, D.: Preventing location-based identity inference in anonymous spatial queries. In: *IEEE TKDE*, pp. 1719–1733 (2007)
8. Ghinita, G., Kalnis, P., Skiadopoulos, S.: PRIVE: anonymous location-based queries in distributed mobile systems. In: Proc. of Int. Conference on World Wide Web (WWW), pp. 371–380 (2007)
9. Indyk, P., Woodruff, D.: Polylogarithmic private approximations and efficient matching. In: Halevi, S., Rabin, T. (eds.) *TCC 2006*. LNCS, vol. 3876, pp. 245–264. Springer, Heidelberg (2006)
10. Khoshgozaran, A., Shahabi, C.: Blind evaluation of nearest neighbor queries using space transformation to preserve location privacy. In: Papadias, D., Zhang, D., Kollios, G. (eds.) *SSTD 2007*. LNCS, vol. 4605, pp. 239–257. Springer, Heidelberg (2007)
11. Yiu, M.L., Jensen, C.S., Huang, X., Lu, C.: SpaceTwist: Managing the trade-offs among location privacy, query performance, and query accuracy in mobile services. In: *ICDE*, pp. 366–375 (2008)
12. Ghinita, G., Kalnis, P., Khoshgozaran, A., Shahabi, C., Tan, K.-L.: Private queries in location based services: Anonymizers are not necessary. In: Proc. of the ACM International Conference on Management of Data, SIGMOD (2008)
13. Cheng, R., Zhang, Y., Bertino, E., Prabhakar, S.: Preserving user location privacy in mobile data management infrastructures. In: Proc. of Privacy Enhancing Technology Workshop (2006)

14. Wang, T., Liu, L.: Privacy-Aware Mobile Services over Road Networks. *PVLDB* 2(1), 1042–1053 (2009)
15. Wang, S., Agrawal, D., Abbadi, A.E.: Generalizing *pir* for practical private retrieval of public data. Technical Report 2009-16, Department of Computer Science, UCSB (2009)
16. Ghinita, G., Vicente, C.R., Shang, N., Bertino, E.: Privacy-preserving matching of spatial datasets with protection against background knowledge. In: Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems, San Jose, California, November 02-05 (2010)
17. Papadopoulos, S., Bakiras, S., Papadias, D.: Nearest neighbor search with strong location privacy. In: *VLDB* (2010)
18. Moon, B., Jagadish, H.v., Faloutsos, C., Saltz, J.H.: Analysis of the clustering properties of the Hilbert Space-Filling Curve. *TKDE*, 124–141 (2001)

Layered Graph Data Model for Data Management of DataSpace Support Platform

Dan Yang^{1,2}, Derong Shen¹, Tiezheng Nie¹, Ge Yu¹, and Yue Kou¹

¹ School of information Science&Engineering,
Northeastern University, Shenyang, 110004, China

² School of Software, University of Science and Technology,
LiaoNing, Anshan, 114051, China

asyangdan@163.com,
{shenderong, nietiezheng, yuge, kouyue}@ise.neu.edu.cn

Abstract. In order to effective management of heterogeneous data sources in dataspace and provide more high quality services, proposing a unified data model to represent all kinds of data in a simple and powerful way is the foundation of DataSpace Support Platform (DSSP). So we propose a novel layered graph Data Model (called lgDM) which includes Entity Data Graph (GD) and Entity Schema Graph (GS) to capture both associations among entities and associations among entity classes. Moreover we also propose an association mining strategy to try to incrementally find associations with less manual effort. We conduct experiments to evaluate the efficiency and effectiveness of our proposed data model.

Keywords: layered graph, data model, association, dataspace.

1 Introduction

Researches of dataspace [1, 2] receive considerable attention recently which is the extension of database management technique. A dataspace may contain all of the information relevant to a particular organization regardless of its format and location, and model a rich collection of relationships between data repositories. Dataspaces offer a pay-as-you-go approach to data management. Users (or administrators) of the system decide where and when it is worthwhile to invest more effort in identifying semantic relationships [4]. Currently one of the main challenges dataspace faces is to propose a data model which can satisfy the data management requirements, break the semantic barrier among data sources and lead to high-effective usage of data sources. Our goal is to provide a data model which can describes heterogeneous data sources and effectively capture their relationships in the dataspace environment. We propose a *layered graph Data Model (lgDM)* to capture both associations among entities and associations among entity classes entity association.

The main contributions of this paper are four-fold:

- We propose a layered graph Data Model (*lgDM*) which includes Entity Data Graph (G_D) and Entity Schema Graph (G_S). Then we give the algorithm of obtaining G_S from G_D .

- We propose a lifecycle of association building process and introduce the mechanism of Associations Constraints Validation (ACV) to improve efficiency and precision of association mining.
- We propose a novel three steps gradual refining association mining strategy which includes clustering entities, candidate entities generating and induction & reasoning based on heuristic & reasoning rules leveraging semantic repository and Knowledge Base.
- We present experiment evaluation to demonstrate the efficiency and effectiveness of our proposed data model.

The remainder of this paper is organized as follows. Section 2 introduces related work. In Section 3 our layered graph model is given. Section 4 gives association mining strategy in detail. Section 5 describes the experimental setup and the results. Section 6 summarizes the main contributions of the paper and our future work.

2 Related Work

The problem of data model is researched in many fields. For example entity-relationship (ER) graphs in relational databases are graph based knowledge representations. And knowledge base and ontology in RDF also adopt graph based data model. Data model in dataspace system faces many new challenges such as heterogenous of data; co-exist of structured, semi-structured and unstructured data; short of schema information and no mediated schema etc. Currently existing data model mostly are in Personal Dataspace System (PDS) such as iMeMex [6] and SEMEX [13]. iMeMex proposed iDM [12] data model for personal information management. iDM allows for the unified representation of all personal information like XML, relational data, file content, folder hierarchies, email, data streams and RSS feeds inside a single data model. But iDM is emphasis on describing structure aspect of data sources but pay less attention to semantic associations between data sources. SEMEX modeled the data from different data sources universally as a set of triples referred to triple base. Each triple is the form (instance, attribute, value) or (instance, association, instance). Thus a triple base describes a set of instances and associations. Though it captured the associations of data sources but it was in PDS environment and can not apply seamless to dataspace environment. [10] was related work in data Web integration which proposed a general graph-based data model and close to that of RDF(S) for web data sources. Our proposed data model is a general dataspace data model and capture semantic associations among data sources. Moreover it is a layered data model and provides the user with logical view on different level.

With respect to the association/relationship mining, the related works are mainly on two fields: social network analysis such as [7, 8] and in dataspace such as [3, 5, 9, 14]. Arnetminer [7] proposed heap-based Dijkstra algorithm to find people associations. But currently only two object relations are pre-defined. [8] developed an unsupervised model to estimate relationship strength from interaction activity and user similarity. But associations in dataspace are more complex and more diversified than people associations in social network. [3, 5] advocated a declarative approach to specifying associations among data items and each set of associations be defined by an *association trail*. But the *association trail* need be predefined by the system or

users. [9] proposed a concept *KnownBy* to model basic relationship between data items and users in Personal Dataspace. [14] adopted Apriori algorithm by mining the frequent item sets with association rules to discover the relationships among data resources in dataspace. Our association mining strategy is not confined in Personal Dataspace and combines induction based on heuristic&reasoning rules and data mining technique.

3 Overview of the Layered Graph Data Model

We treat the Entity (Object) as the smallest data unit to build our data model in dataspace. The layered graph Data Model (*lgDM*) is graph based and includes *Entity Data Graph* (denoted as G_D) and *Entity Schema Graph* (denoted as G_S). The G_D describes entities and captures associations among Entities; the G_S describes meta information of entity classes and captures associations among Entity classes. Both graphs exhibits semantics-bearing labels for nodes and edges and can thus be seen as a semantic graph, with nodes corresponding to entities and edge weights capturing the strengths of semantic relationships. Because data sources in dataspace are short of schema information or schema-less at first. The approach of dataspace is to build schema in a pay-as-you-go way. Our data model adopts the principle of pay-as-you-go to build schema information of Entity Class. We can get G_S from existing G_D . *lgDM* provides a layered logical view for the data in dataspace which captures both schema and data-level features of heterogeneous data sources in dataspace. It's useful for entity-relationship-oriented semantic keyword search, the searching results can be lists of entities or entity pairs instead of documents for users to extract relative information or facts.

3.1 Definitions

Firstly we give the following definitions used in our data model:

Definition 1. *Entity* is composed of a set of $\langle \text{attribute}, \text{value} \rangle$ pairs to describe a real world object. Where the *value* can be an atomic value (e.g., string, int etc.) or composite value and it has domain range e.g. age of a person can't be a negative integer. We allow a same attribute to appear several times (e.g. one person has more than one email address). An Entity example is as follows: *Tom*= {<name, 'Tom'>, <email, 'tom@yahoo.com'>, ... <affiliation, 'Google' >}

Definition 2. *Entity Class* is a primitive concept. It is type or class to which Entity instance belong. For example entity class from the academic domain contains four classes: {*Person, Author*}, {*Paper, Article*}, {*Conference, Meeting*}, {*Journal, Magazine*} from academic domain. Entity Class is composed of Entities instances.

Definition 3. *Association* is directional binary relation between two Entities or two Entity Classes. In the paper we use symbol ' $\xrightarrow{\text{associationName}}$ ' (a directed arrowed line with label of association name) to represents Association. Association between two entities is bidirectional namely if $e1 \xrightarrow{\text{associationName}} e2$ then accordingly $e2 \xrightarrow{\text{associationName}} e1$. For example *co-author* association between two persons, *sameTopic* association between

two articles. *authorOf* association between Entity Class *Person* and *Article*. *beAuthorOf* association between Entity Class *Article* and *Person*.

3.2 Entity Data Graph (G_D)

$G_D(N_D, E_D, L_D)$ is a directed graph with labels corresponding to associations of entities, where N_D is nodes set, E_D is edges set and L_D is labels set. Nodes in N_D are: (a) *Entities*, labeled with instance ID or unique access address (e.g. document file path); (b) *Attributes Values*, labeled with value of attribute. Edges in E_D are: (a) *Attribute Edges*, labeled with attributes of entity; (b) *Association Edges*, labeled with association between two instances. The triple format descriptions of them are as follows: $\langle \text{Entity}, \text{attribute}, \text{value} \rangle$, $\langle \text{Entity}, \text{Association}, \text{Entity} \rangle$. Labels in L_D are: (a) association labels on association edges. (b) *Attribute Labels* on attribute edges. We model each association as two edges: a forward edge and backward edge. Note that the number of association edges between two nodes maybe more than one to describe different associations between two entities. Fore example association between two persons maybe co-author, couple etc. Associations between two articles maybe are *citedBy* and *sameTopic*. An example of G_D with three persons entities ($p1 \sim p3$), five article entities ($a1 \sim a5$), two conference entities ($c1 \sim c2$) and one journal entity ($j1$) from four Entity Classes is showed in Fig.1 (we omissions the backward association edges for simplicity reason). Where ellipses nodes represent entities, rectangles nodes represent attribute values, unidirectional edges represent attributes, and directional edges represent associations. And the edges of dotted line are associations mined from the existed associations (see section 4.2).

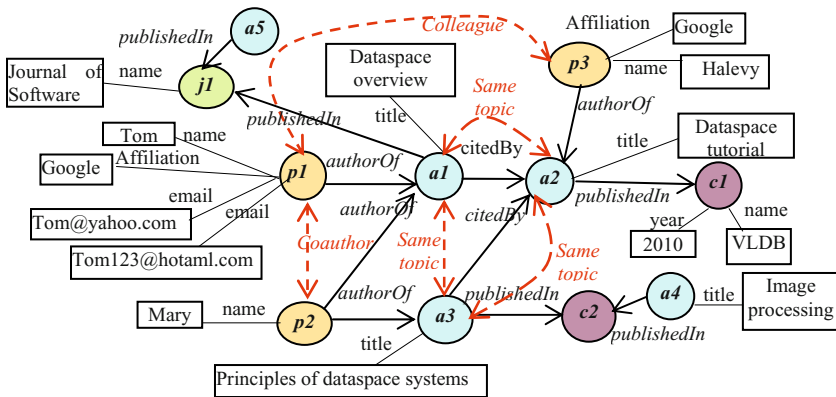


Fig. 1. Example of data graph G_D

3.3 Entity Schema Graph (G_S)

$G_S(N_S, E_S, L_S)$ is a directed labeled graph to describe meta information of Entity classes and capture associations of Entity Classes, where N_S is nodes set, E_S is edges set and L_S is labels set. Nodes in N_S are *Entity Classes* with Entity Class name and its meta information; Edges in E_S are *Association Edges* labeled with association between

two entity classes. And the triple format of association edge is \langle Entity Class, association, Entity Class \rangle . Labels in L_S are association labels on association edges. Schema graph summarizes the information contained in a data graph G_S . Fig. 2 shows the corresponding G_S of G_D in Fig. 1. Where node represents Entity Class and directional edges represent associations between entity classes. In the schema graph we add some dimensions of meta knowledge to facilitate user's querying for provenance, trust, temporal in dataspace. We define meta knowledge of each Entity Class as a triple \langle source, certainty degree, timestamp \rangle which describes the facts have been extracted from different sources, at different time-points, and with different degrees of extraction confidence. For example Entity Class person \langle source=www.dblp.com; certainty degree=0.8; timestamp=3/1/2011 \rangle .

We can obtain G_S from existing G_D . We construct G_S from G_D as the following steps: **Step 1.** For each node $n \in G_D$, the attribute values and their unidirectional edges are removed. **Step 2.** for each node $n \in G_D$, add entity class label for each node n in G_D . For each edge $e(e_i, e_j) \in G_D$, add to G_S an edge with same label from each entity class to associated entity class. **Step 3.** if e_i and e_j are from the same entity class then merges them to keep only one node, and add association label to the entity class itself. The algorithm is given in Algorithm 1.

Algorithm 1. G_D To G_S (G_D)

Input: $G_D(N_D, E_D, L_D)$

Output: $G_S(N_S, E_S, L_S)$

1. **Begin**

2. DFS or BFS of G_D //Deep-first Search or Breadth-first Search of G_D

3. **for** each node $\in N_D$ **do**

4. **If** \forall node \in Entities **then**

5. Add Entity Class label to the node;

6. **End If**

7. **If** \forall node \in Attribute values **then** //remove the nodes and connected attribute edges

8. Remove node;

9. Remove connected Attribute edge;

10. **End If**

11. **for** each edge $\langle n_i, n_j \rangle \in$ Association edge

12. **If** (Entity Class(n_i)=Entity Class(n_j)) **then**

13. merge n_i, n_j

14. add association edge to n/n_j itself

15. **End If**

16. **End for**

17. **End**

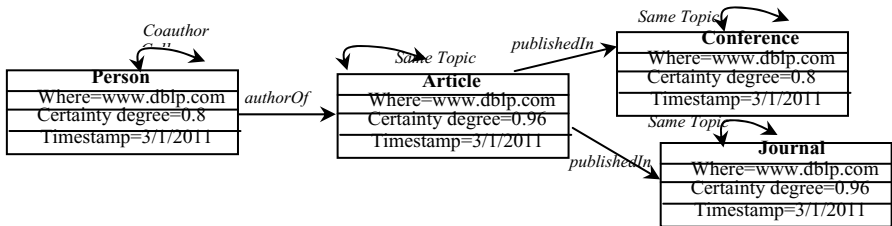


Fig. 2. Example of schema Graph G_S

4 Associations Building Process

In real world scenario there exist all kinds of associations among entities which are foundations of our semantic keyword query. And associations or relations among entities (objects) are also useful in solving other problems such as entity resolution (reference reconciliation). The challenge in this process is how to build and complete associations accurately with less manual effort. We define the lifecycle of associations building process as the following four phases:

- 1) **Preprocess phase.** In this phase some data cleaning jobs such as synonyms or abbreviations, data format heterogeneous problem and attribute structure heterogeneous problems and even the entity resolution problem are done. For example “conference” or “meeting” or “proceeding” are synonyms and have the same meaning. “SIGMOD” is the abbreviation of “Special Interest Group on Management of Data”. For example person’s name may be “name” or maybe “first Name” combine “last name”, in our data model we use a unique description and assume there is no such structure heterogeneous problems when mining associations.
- 2) **Initial phase.** We try to get associations automatically from multiple types of data sources. In initial phase associations can be obtained from the following ways. Firstly, some associations are pre-defined by some domain experts. Secondly, some associations are obtained from particular programs such as address book of emails or excel files etc. Thirdly, we can use some tools to extract associations from author part of text/ PDF files such as papers and thesis. Fourthly, associations can induced from relations of relational tables or deep Web tables directly. Fifthly, associations can extract from existing digital library such as DBLP for computer scientists, IMDB for movies. Finally associations can be extracted from semi-structured or unstructured data such as XML/HTML web page using wrappers.
- 3) **Complete phase.** In complete phase, more associations can be derived or mined from existing ones. The challenge is to incrementally find and build meaningful associations between entities. The detail of association mining strategy is presented in section 4.2.
- 4) **Maintenance phase.** When new data sources are added or new entities and associations between entities are added, G_D and G_S need to be updated accordingly to make them keep consistency. We refer to the process of updating the graphs’ nodes and associations as *graph maintenance*. We can get inspiration from [11] which automatically incorporating new sources in keyword search-based data integration. Details of graph maintenance are beyond the scope of this paper.

4.1 Association Constraint Validation

Associations Constraints Validation (ACV) is a powerful building block for high-precision harvesting of associations. Sometimes the associations extracted in initial phase or mined in complete phase maybe have errors due to many reasons (e.g. dirty data source or manual fault or false positive). For example an article is published in

one proceeding (e.g. SIGMOD 2010) and it can not be published in another proceeding (e.g. VLDB 2010) at the same time. So we introduce the mechanism of ACV in the lifecycle of associations building process. By doing ACV, many incorrect hypotheses or intermediate results can be excluded early to improve efficiency and precision.

Here we distinguish the following kinds of ACV.

- 1) **Type checking (Entity Class checking).** Binary association has an accordingly type signature. For example, $Mary \xrightarrow{hasAlmaMater} Paris$ can be immediately falsified because Paris is a city and not a university, and thus violates the type condition.
- 2) **Temporal Consistency checking.** For assessing and cleaning temporal hypotheses in an association repository, consistency checking plays a key role. For example, marriages of the same person must not overlap in time. Constraints may refer to the relative ordering of facts: if person $p1$ is doctoral advisor of person $p2$, then $p1$ must have graduated before $p2$ in time sequence. When reasoning about the validity of facts in the presence of such constraints, both base facts and temporal scopes need to be considered together. For example two articles publisher in the same conference can not have the *citedBy* association.
- 3) **Monotonous (acyclic) consistency checking.** Some associations among entities (the number of entity ≥ 3) can't form a cycle such as associations with monotonous semantic such as $\xrightarrow{lessThan}$, $\xrightarrow{moreThan}$. For example if articles $a1$, $a2$ and $a3$ have the associations $a1 \xrightarrow{citedBy} a2$, $a2 \xrightarrow{citedBy} a3$ and $a3 \xrightarrow{citedBy} a1$, it's wrong because the *citedBy* associations form a cycle and it's impossible and wrong in logic.

4.2 Associations Mining Strategy

We propose a three-step gradual refining association mining strategy which includes clustering entities, candidate entities generating and induction & reasoning based on heuristic & reasoning rules leveraging semantic repository and Knowledge Base which is showed in Fig. 3. The availability and leverage of Knowledge Base that know Entities Classes and many facts about them can boost the effectiveness of association mining. Relations of Entity Classes such as subclass/superclass connections-inclusion dependencies and hyponymy/hypernymy relations are stored in Knowledge Base. For example, "China chairman" is a subclass of "politicians". In our approach we distinguish entities as *Heterogeneous Entities* and *Homogeneous Entities*, and distinguish associations as *kernel association* and *expanded association*.

Definition 4. Homogeneous Entities (denoted as *HomoEs*) is set of entities of same Entity Class. For example Entity $p1$ and Entity $p2$ are *HomoEs* because they both are Entity Class *person*.

Definition 5. Heterogeneous Entities (denoted as *HeterEs*) is set of entities of different Entity Class. For example Entity $p1$ and Entity $a1$ are *HeterEs* because $p1$ is Entity Class *person* and $a1$ is Entity Class *article*.

Definition 6. Kernel Association is predefined by some domain experts or obtained from the extracted information which is stored in association repository. For example kernel associations in academic domain may include: $Paper \xrightarrow{citedBy} Paper$, $Person \xrightarrow{authorOf} Paper$, $Paper \xrightarrow{publishedIn} Journal$ and $Paper \xrightarrow{publishedIn} Conference$.

Definition 7. Expanded Association is derived from *kernel association* by data analyzing and mining *kernel associations* of entities. For example co-author, co-organization association of persons, *sameTopic* of articles. There are two kinds of *expanded associations*: one kind is associations among *HomoEs*. Another kind is associations among *HeterEs*.

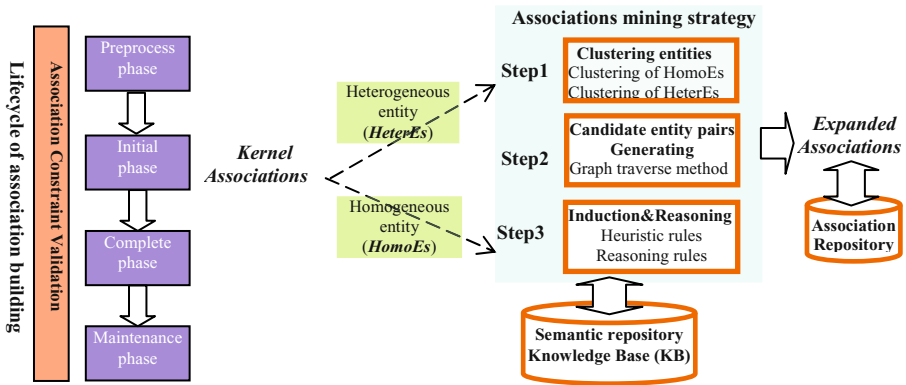


Fig. 3. Strategy of associations mining

Step 1. Clustering entities. In this step we adopt different clustering algorithms of data mining to cluster *HomoEs* and *HeterEs*.

Clustering of *HomoEs*. We use k-means clustering algorithm to cluster *HomoEs* based on their values on one attribute, and entities in each cluster have same or similar attribute values (based on edit distance or semantic distance) on one attribute, then any pair of entities in the same cluster have the *sameAttribute* association. For example in Fig. 1, given a cluster number $k=2$, clustering result of paper entity $a1, a2, a3, a4$ and $a5$ based on value of attribute *title* is $Cluster_1 \{a1, a2, a3, a5\}$, $Cluster_2 \{a4\}$. Article entities in cluster C_1 are similar on value (e.g., *dataspace*) of *title* attribute and have *sameTopic* association each other: $a1 \xrightarrow{sameTopic} a2$, $a3 \xrightarrow{sameTopic} a2$, $a1 \xrightarrow{sameTopic} a3$.

Clustering of *HeterEs*. We use k-medoids clustering algorithm to cluster *HeterEs* by transferring it into clustering problem of *HomoEs* records. The center entity of each cluster is called medoid, then any pair of entities in the same cluster have some association related with the medoid of the cluster. For example entities in Fig.1, given a cluster number $k=2$ and entity class of medoid is *Conference*, then the clustering result may be $Cluster_1 \{c2, p2, a3, a4\}$ and $Cluster_2 \{c1, p1, a1, a2, p3, j1, a5\}$, so the

HeterEs in $Cluster_1$ are related to conference $c2$ and the *HeterEs* in $Cluster_2$ are all related to conference $c1$. Another example, given a cluster number $k=3$ and entity class of medoid is *Person*, the clustering result may be $Cluster_1\{p1, j1, a1, a5\}$, $Cluster_2\{p3, a2, c1\}$, $Cluster_3\{p2, a4, a3, c2\}$, so the *HeterEs* in $Cluster_1$ are all related to person $p1$, likewise for $Cluster_2$ and $Cluster_3$.

Step 2. Candidate entity pairs generating. We traverse graph to generate candidate entities for induction&reasoning of next step from the result clusters of step 1. Because if two entity classes are less related in semantic then the strength of association between two entities of them maybe is very weak or the association is useless, so we only focus on finding the most ‘goodness’ *expanded associations* which satisfy *structure cost* and *semantic cost* at the same time. First given a walk length L , the two Entities e_i and e_j are reachable in the data graph (i.e., there is a path between e_i and e_j in graph and the length of the path is less than L). Second semantic support of the two entities in Knowledge Base or semantic repository are bigger than a threshold θ (defined by user or system). For example in Fig.1, given $L=3$, then $a5$ and $c1$ is not candidate entities because the path length from $a5$ to $c1$ is 4 which not satisfy the first condition. the semantic distance between two entities which is defined as the association strength between e_i and e_j .

Step 3. Induction & reasoning based on Heuristic&Reasoning Rules. We give some useful Heuristic rules from observations and intuition to mine *expanded associations* aiming at the candidate entities.

Heuristic rule 1. Same source rule. Given Entities $e1, e2, e3$ and their associations: if $e1 \xrightarrow{X} e3$ and $e2 \xrightarrow{X} e3$, where X is some association, then usually there is $e1 \xrightarrow{\text{someAssociation}} e2$. For example $p1 \xrightarrow{\text{authorOf}} a1$, and $p2 \xrightarrow{\text{authorOf}} a1$, then $p1$ and $p2$ have co-author association namely $p1 \xrightarrow{\text{co-author}} p2$.

Heuristic rule 2. Same target rule. Given Entities $e1, e2, e3$ and their associations: if $e1 \xrightarrow{X} e2$ and $e1 \xrightarrow{X} e3$, where X is some association, then usually there is $e2 \xrightarrow{\text{someAssociation}} e3$. For example, if the known associations are $p2 \xrightarrow{\text{authorOf}} a1$ and $p2 \xrightarrow{\text{authorOf}} a3$, then $a1$ and $a3$ have *sameTopic* or *similarTopic* association with high probability because it’s written by the same person.

Heuristic rule 3. Transitive rule. Given Entities $e1, e2, e3$ and their associations: if $e1 \xrightarrow{X} e2$ and $e2 \xrightarrow{Y} e3$, where X, Y are associations ($X=Y$ or $X \neq Y$), then there must be $e1 \xrightarrow{\text{someAssociation}} e3$. For example in Fig.1, $p3 \xrightarrow{\text{authorOf}} a2$ and $a2 \xrightarrow{\text{publishedIn}} c1$, then $p3 \xrightarrow{\text{e.g.isSpeakerIn}} c1$, here $X \neq Y$. Another example with $X=Y$, given three person entities $p1, p2, p3$ and their associations $p1 \xrightarrow{\text{isParentOf}} p2$, $p2 \xrightarrow{\text{isParentOf}} p3$, then $p1 \xrightarrow{\text{isGrandparentOf}} p3$.

Heuristic rule 4. Same attribute rule. Given Entity $e1\{(attribute_1, value_1), \dots, (attribute_n, value_n)\}$ and $e2\{(attribute_1, value_1), \dots, (attribute_n, value_n)\}$, where $e1 \in$ Entity class C and $e2 \in$ Entity class C , if both of them have the same or similar values (based on edit distance or semantic distance) on $attribute_i$: $e1(value_i) \approx e2(value_i)$ (symbol ' \approx ' denotes same or similar), then there must be $e1 \xrightarrow{sameAttribute} e2$. For example from Fig.1, article $a1\{(title, 'dataspace\ technique') \dots\}$ and article $a2\{(title, 'dataspace\ overview') \dots\}$, then $a1 \xrightarrow{sameTopic} a2$. Another example from Fig.1, person $p1\{(affiliation, Google) \dots\}$ and person $p3\{(affiliation, Google) \dots\}$, then $p1 \xrightarrow{colleague} p3$.

Associations are not existed dependently but are related to each other in some degree such as hierarchy relationships and other semantic relationships. So we define some *reasoning rules* to mine expanded associations leveraging semantic repository and Knowledge Base (KB). We use symbol ' $\Rightarrow_{semantic}$ ' to denote the semantic relationship between associations and symbol ' $\Rightarrow_{hierarchy}$ ' to denote the concept hierarchy relationship between associations. Currently we consider the following hierarchy relationships: subclassOf, subPropertyOf.

Reasoning rule 1. Semantic reasoning. Given two associations X and Y , if $X \Rightarrow_{semantic} Y$ and entities $e1 \xrightarrow{X} e2$ are known, then $e1 \xrightarrow{Y} e2$. For example, given *citedBy*, *sameTopic* associations have the relationship $citedBy \Rightarrow_{semantic} similarTopic$, and article $a1, a2$ have the association $a1 \xrightarrow{citedBy} a2$, then it is with high probability that $a1$ and $a2$ have *sameTopic* association namely $a1 \xrightarrow{similarTopic} a2$.

Reasoning rule 2. Hierarchy reasoning. Given two associations X and Y , if $X \Rightarrow_{hierarchy} Y$ and entities $e1 \xrightarrow{X} e2$ are known, then $e1 \xrightarrow{Y} e2$. For example, given *isFatherOf*, *isParentOf* associations have the relationship $isFatherOf \Rightarrow_{hierarchy} isParentOf$ (because *isFatherOf* is subclassOf *isParentOf*) and person $p1, p2$ have the association $p1 \xrightarrow{isFatherOf} p2$, then $p1$ and $p2$ have *isParentOf* association namely $a1 \xrightarrow{isParentOf} a2$.

Reasoning rule 3. Transitive reasoning. Known three entities $e1, e2, e3$ and their associations $e1 \xrightarrow{X} e2$ and $e2 \xrightarrow{X} e3$, where $X \in \{sameAS\ semantic\ e.g.\ sameTopic, sameConference\}$ then there is $e1 \xrightarrow{X} e3$. For example in Fig. 1 there are associations $a1 \xrightarrow{sameTopic} a2$ and $a2 \xrightarrow{sameTopic} a3$, then there is $a1 \xrightarrow{sameTopic} a3$.

5 Experiments

First we design two experiments to evaluate our data model. One is from the efficiency facet and one is from the effectiveness facet. Then we conduct experiments to compare precision of different association mining strategies.

5.1 Data Set

Our experiments use data from the DBLP data set. The data set contains 10532 entities (articles, persons, conferences, research organizations and journals). We conduct the experiments on a 3.16 GHz Pentium 4 machine with 4GB RAM and 500GB of disk.

5.2 Efficiency and Effectiveness Evaluation

We consider index lookup time and query response time to evaluate *lgDM*. To support keyword query we build two indexes: structure index that indexes attributes of entity and associations; value index that indexes attribute values of entities. From the above data set we select keywords randomly and the number of keywords is varied from 2 to 5 denoted as $N2\sim N5$. For each kind of $N2\sim N5$, we build 100 queries randomly. The average index lookup time and query response time is shown in Fig. 4.

We conduct experiments to find the effectiveness of associations to keyword query which showed in Fig. 5. The X-axis indicates the number of associations; the Y-axis indicates the number of entities returned from the keyword query. From Fig. 5 we can know that the result number increases with the number of associations. Experiment results show that association mining is key and important to semantic keyword query.

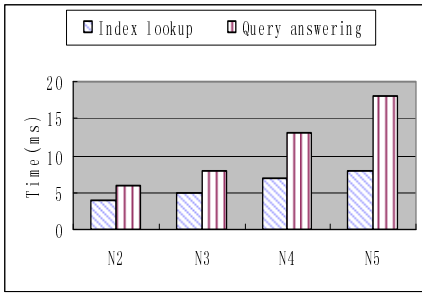


Fig. 4. Average index lookup and query response time

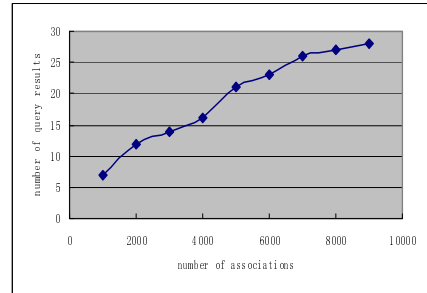


Fig. 5. Effectiveness of associations to keyword query

5.3 Comparison of Association Mining Strategies

We compare the precision of the following two association mining strategies:

- Strategy 1: only adopts two steps including clustering entities and induction & reasoning without step 2.
- Our strategy: three steps gradual refining strategy which includes clustering entities, candidate entities generating and induction&reasoning.

The experiment result is showed in Fig. 6. We can know from the Fig. 6 that our strategy is better than strategy 1 in precision, and the trend is obvious when the kernel associations are more.

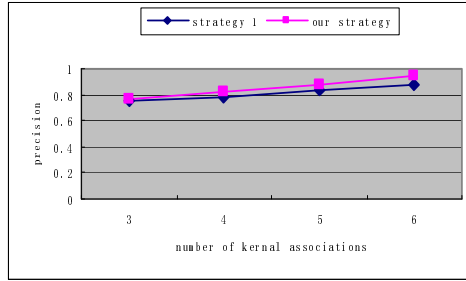


Fig. 6. Performance comparison of association mining strategies

6 Conclusion and Future Work

We propose a layered graph Data Model (*lgDM*) which introduces a layered logical view on top of the data sources that provides physical and logical abstract of dataspace information independently. *lgDM* can be extend and maintenance easily. And we also propose a lifecycle of association building process and introduced the concept of association constraint validation (ACV) and we propose a novel three steps gradual refining association mining strategy. We do experiments to evaluate effectiveness and efficiency of our data model and we also do experiments to compare different association mining strategies. Our next work will consider mining the associations or relationships among clusters (homogenous clusters and heterogeneous clusters) to satisfy keyword query of potential use.

Acknowledgments. This work is supported by the NSFC under Grant No. 60973021 and No. 61003059.

References

1. Michael, F., Alon, H., David, M.: From databases to dataspace: a new abstraction for information management. In: SIGMOD 2005 (2005)
2. Halevy, A., Franklin, M., Maier, D.: Principles of dataspace systems. In: Proc. of PODS (2006)
3. Marcos, A., Vaz, S., Jens, D., Lukas, B.: Intensional associations in dataspace. In: ICDE 2010 (2010)
4. Franklin, M.J., Halevy, A.Y., Maier, D.: A first tutorial on dataspace. In: PVLDB (2008)
5. Marcos, A., Vaz, S., Jens, D., Lukas, B.: Adding structure to web search with iTrails. In: ICDEW (2008)
6. Lukas, B., Jens-Peter, D., Olivier, R.G.: A dataspace odyssey: the iMeMex personal dataspace management system. In: 3rd Biennial Conference on Innovative Data System Research, pp. 114–119 (2007)
7. Tang, J., Zhang, J., Zhang, D., Yao, L., Zhu, C.: ArnetMiner: An expertise oriented search system for web community. In: The Sixth International Semantic Web Conference (2007)

8. Xiang, R., Neville, J., Rogati, M.: Modeling relationship strength in online social network. In: Proceedings of the 19th International Conference on World Wide Web, pp. 981–990 (2010)
9. Li, Y., Meng, X., Kou, Y.: An efficient Method for constructing personal dataspace. In: Sixth Web Information Systems and Applications Conference (2009)
10. Tran, T., Haofen, W., Peter, H.: Hermes: DataWeb search on a pay-as-you-go integration infrastructure. *Web Semantics: Sci. Serv. Agents World Wide Web* (2009)
11. Talukdar, P.P., Lves, Z.G., Pereira, F.: Automatically incorporating new sources in keyword Search-based data integraion. In: SIGMOD (2010)
12. Dittrich, J.-P., Salles, M.A.V.: iDM: A Unified and Versatile Data Model for Personal Dataspace Management. In: VLDB (2006)
13. Dong, X.: Providing Best-Effort Services in Dataspace Systems. Ph.D. Dissertation. Univ. of Washington (2007)
14. Dong, Y., Shen, D., Kou, Y.: Discovering relationships among data resources in dataspace. In: Sixth Web Information Systems and Applications Conference (2009)

A Study of RDB-Based RDF Data Management Techniques

Vahid Jalali, Mo Zhou, and Yuqing Wu

School of Informatics and Computing, Indiana University, Bloomington
{vjalalib,mozhou,yugwu}@indiana.edu

Abstract. RDF has gained great interest in both academia and industry as an important language to describe graph data. Several approaches have been proposed for storing and querying RDF data efficiently; each works best under certain circumstances, e.g. certain types of data and/or queries. However, there was lack of a thorough understanding of exactly what these circumstances are, as different data-sets and query sets are used in the empirical evaluations in the literature to highlight their proposed techniques. In this work, we capture the characteristics of data and queries that are critical to the RDF storage and query evaluation efficiency and provide a thorough analysis of the existing storage, indexing and query evaluation techniques based on these characteristics. We believe that our study not only can be used in evaluating both existing and emerging RDF data management techniques, but also lays the foundations for designing RDF benchmarks for more in-depth performance analysis of RDF data management systems.

Keywords: RDF, SPARQL, Storage, Index, Query Evaluation.

1 Introduction

Resource Description Framework (RDF) [3] which is a World Wide Web Consortium (W3C) recommended standard, represents data entities and their relationships in the Semantic Web. In RDF, each data entity has a Unique Resource Identifier (URI) and each relationship between two data entities is described via a triple within which the items take the roles of *subject(S)*, *predicate (or property)(P)* and *object(O)*. RDF Schema (RDFS) [6] further extends RDF to describe the semantic and structure of the data by introducing classes.

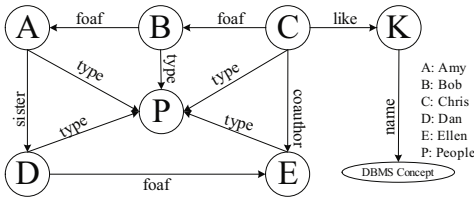


Fig. 1. Example RDF data

Fig. 1 shows a graph representation of a sample RDF data and its schema that describes people, books and their relationship in social networks. The arch “Amy type Person” indicates that *Amy* is an instance of the class named “Person”.

SPARQL [9] is an RDF query language recommended by W3C to

facilitate users to retrieve meaningful information from RDF data. A SPARQL query consists of graph pattern(s) that can identify the nodes/edges of interest through graph pattern matching against the RDF data. The following query retrieves a person who is a friend of A, and his/her relationship with B is the same as D's relationship with E.

```
SELECT ?person WHERE {?person foaf A . ?person ?p B. D ?p E}
```

To keep pace with the ever-increasing volume of semantic web data and the needs of answering complicated queries on such data, various RDF storage methods were proposed to improve the RDF data storage and query evaluation efficiency. Majority of these approaches rely on existing relational database (RDB) management systems, among which the most notable techniques are Triple Store [7], Vertical Partitioning [4] and Property Table [19]. Indexing methods were also investigated, including MAP [10], Hexastore [18] and TripleT [8].

It is the general wisdom that engineering designs all have their advantages and drawbacks; they are superb for certain circumstances but less so for others. Benchmarks, which consist of both data set and query set, are designed to help users determine the strength and limitations of a data management system or a specific data management technique. In the context of RDF, many efforts [14,15] have been made in this regard. However the question “what characteristics of data and query highlight the advantages and drawbacks of a storage method” is yet to be answered, as existing benchmarks are not yet covering key characteristics of RDF data and queries, especially those that bring challenges to RDF storage and query evaluation techniques.

We set out to conduct such a thorough analysis, focusing on the type of data and queries that are not well covered (e.g. the sample query above) by the existing benchmarks and studies. In particular,

- We introduce and analyze a set of key characteristics of RDF data for analyzing the space efficiency of RDF storage methods.
- We classify SPARQL queries based on roles of the variables, and identify the challenges brought by data storage and query evaluation techniques.
- We design extensive empirical study to investigate and compare existing RDF data management techniques, and prove that the data and query characteristics we identified are indeed critical.
- We analyze existing RDF benchmarks based on the data and query characteristics we propose, and suggest improvement in RDF benchmark designing.

2 Storage Efficiency

In this section, we study the space efficiency, i.e. the amount of space required for storing RDF data, of different RDB-based RDF data storage methods, specifically nonindex-based and index-based methods.

Definition 1. *Given an RDF document D in triple format, and a data storage or indexing method \mathcal{M} that is designed to store D , the storage efficiency of \mathcal{M}*

with respect to document D is $SE_{\mathcal{M}}^D = \frac{S_{\mathcal{M}}(D)}{S(D)}$, where $S(D)$ represents the size of the text document D in triple format, and $S_{\mathcal{M}}(D)$ represents the space required for storing D using method \mathcal{M} .

Obviously, the smaller the value of $SE_{\mathcal{M}}^D$, the better method \mathcal{M} is for storing/indexing document D . In the rest of this section, we study the most prominent RDF data storage and indexing techniques in the literature, identify key characteristics of RDF data that determine the storage requirement, and propose formulas for calculating $S_{\mathcal{M}}(D)$ for each method. Then, we will report the result of our impartial study on the storage efficiency of these techniques, w.r.t. RDF documents with various combinational configurations on the key characteristics.

2.1 RDF Data Storage Methods

The most prominent RDF data storage methods are Triple Store (TS) [7], Vertical Partitioning (VP) [4] and Property Table (PT) [19]. Fig. 2 illustrates how these three methods store some fragments of the RDF data whose graph representation was shown in Fig. 1.

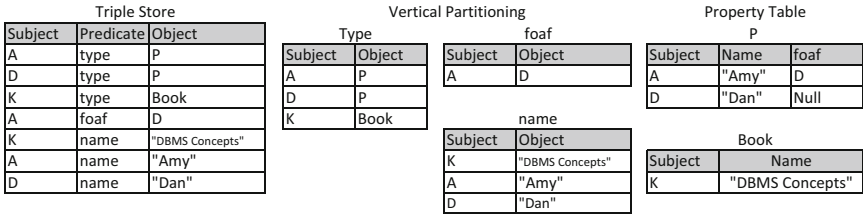


Fig. 2. The Sample Data Stored in TS, VP and PT

The space requirements of these storage methods are shown in the table on the right. We use $|D|$ to represent the total number of triples in D and L the average size of all values in D , while the other data characteristics and how they impact the storage efficiency of a data storage method will be discussed in details next.

	Data	Schema Overhead
TS	$3 \times D \times L$	O_{TS}
VP	$2 \times D \times L$	$ pred(D) \times O_{VP}$
PT	$(D - lft(D)) \times L + 3 \times lft(D) \times L$	$ cls(D) \times O_{PT} + O_{TS}$

In **Triple Store** [7], all triples are stored in a single table with three columns (S, P, O). Therefore, the space requirement is determined by the number of tuples in that table, e.g. the number of triples in the RDF data ($|D|$), and the size of each value to be stored. The only overhead (O_{TS}) is for storing the schema of the triple store table in the system catalog.

In **Vertical Partitioning** [4], a separate table is created for each distinct predicate to store all triples that feature this predicate. As all triples in each table share the same predicate, the predicate is omitted and only the values of the subject and object roles are stored. Therefore, the space requirement is determined by the number of tuples in these tables, as well as the number of tables, which is the number of unique predicates in the RDF data ($|pred(D)|$).

In **Property Table** [19], a separate table is created for each distinct predicate to store all triples that feature this predicate. As all triples in each table share the same predicate, the predicate is omitted and only the values of the subject and object roles are stored. Therefore, the space requirement is determined by the number of tuples in these tables, as well as the number of tables, which is the number of unique predicates in the RDF data ($|pred(D)|$).

The overhead (O_{VP}) is incurred by storing the schema of each VP table, which increases as $|pred(D)|$ increases.

In **Property Table [19]**, RDF data are stored in traditional relational tables whose schema carry the semantics of the residential data. There are two types of PTs that provide distinct ways in shredding RDF triples into tables: (1) *Clustered Property Table*, in which RDF triples with the subjects sharing the same set of predicates are clustered into a property table, generated manually, or using a dynamic lattice based method [17]; and (2) *Property-class Table*, in which a table is created for each class and stores information about all members of that class, with the attributes being the single-value predicates of these members. In both implementations, a leftover table with three columns (S, P, O) is created for storing the triples not belonging to any other tables. In the former method, the schema of the property tables highly depends on the clustering algorithms, which varies based on the implementations. Thus we focus on the latter method in which the property tables are determined once the schema information is given. In the rest of the paper, when we refer to Property Table method, we are indeed referring to the property-class table method. In contrast to vertical partitioning and triple store, it is not straightforward to decide the schema of the tables used for storing RDF data in property tables.

The space requirement for PT method consists of two parts: the space required for storing the property tables and the space required for storing the leftover table, each of which, similar to our discussion of the TS and VP methods, consists of a data component and a schema component. The number of property tables is the number of classes ($|cls(D)|$). $|lft(D)|$ represents the number of triples that cannot be placed in any property table but have to be placed in the leftover table. The overhead for each property table is denoted by O_{PT} , while the overhead of the leftover table is the same as a triple store table, O_{TS} .

Assuming the space required for storing the information of each attribute in the system catalog is the same, denoted C , and the overhead for storing the information of a table is the same, denoted O , then, the overhead for storing the schema information, referred to as O_{TS} , O_{VP} and O_{PT} in the table above, can be further depicted using the formula on the right. Here, $avgPred(D)$ represents the average number of single-value predicates of each class. Note that in the PT method, besides the attributes that correspond to the single-value predicates of each class, an additional attribute is introduced to store the URIs of the subjects.

$O_{TS} = O + 3 \times C$ $O_{VP} = O + 2 \times C$ $O_{PT} = O + (avgPred(D) + 1) \times C$
--

2.2 Index-Based Storage Methods

To facilitate efficient query answering, multiple indexing techniques were proposed, including MAP [10], Hexastore [18] and TripleT [8], whose structures are illustrated in Fig. 3. Indeed the index-based methods proposed are not merely designs of indices but alternative ways for storing RDF data for efficient data accessing based on indexing techniques. Hence, we call them *index-based storage methods*.

Before we discuss the space requirement of the index-based storage methods, we review the space requirement of a traditional clustered B⁺-tree index. Given the number of unique values of the index key N_k , the data size of the index key S_k , the size of a pointer S_{ptr} , and the size a page S_P (assume that S_{ptr} and S_P are fixed), the

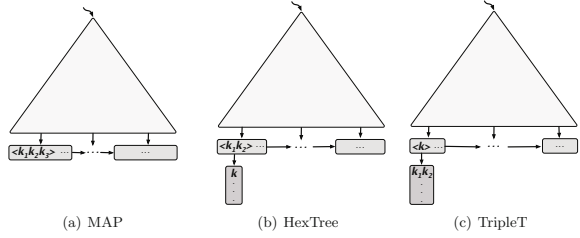


Fig. 3. RDF Indices [8]

space requirements of the B⁺-tree is $S_{BT}(N_k, S_k) = S_P \times \sum_{i=1}^{\log_f N_k} f^i$, where the fan-out f is computed as $f = \frac{S_P}{S_k + S_{ptr}}$.

The space requirement of each index-based storage method consists of two parts: (1) the space requirement of the clustered B⁺-tree indices; and(2) the space requirement of the payload pointed by the leaf nodes in B⁺-tree indices. We summarize these in the table below.

	Payload	Index
MAP	0	$6 \times S_{BT}(D , 3 \times L)$
Hexastore	$\sum_{role=\{S,P,O\}} \pi_{role} D \times L$	$6 \times \sum_{role=\{SP,SO,PO\}} S_{BT}(\pi_{role} D , 2 \times L)$
TripleT	$3 \times val(D) \times L$	$S_{BT}(val(D) , L)$

MAP [10] builds six clustered B⁺-tree indices on six RDF triple store tables, each clustered on one permutation of S, P and O, i.e. SPO, SOP, PSO, POS, OSP, OPS. As a result, all RDF triples are stored six times on the leaf nodes of these indices. On each such replication a B⁺-tree index is built, with $N_k=|D|$ and $S_k=3 \times L$. Please note that there is no additional payload as all three roles are indexed. More recently, MAP method was enhanced by indexing over aggregated functions and by using index compression techniques [13].

Hexastore [18] improves the storage efficiency of MAP by reducing the space requirement of the index part and the duplication of the RDF data. Instead of indexing all three roles, Hexastore builds six clustered B⁺-tree indices on two out of three roles of RDF triples, i.e. SP, PS, SO, OS, PO, OP, while the two indices on symmetric roles, e.g. SP and PS, share the same payload that contains the distinct values of the other role, in this case O. In the worst case, the RDF triples are duplicated five times.

Rather than creating six B⁺-tree indices, **TripleT** [8] uses only one B⁺-tree to index all distinct values in an RDF document, across all roles. Each leaf node in this B⁺-tree has a payload that is split into three buckets, S, P and O, and each bucket holds the list of related atoms for the other two roles. Therefore the space requirement of TripleT is the sum of (1) the overall payload, in which each triple in the RDF data is stored 3 times, one under its subject’s value, one

under its predicate’s value, and one under its object’s value; and (2) the size of a single B^+ -tree index.

2.3 RDF Data Characteristics

Based on the analysis above, we identify the following as key factors that can be used to describe the characteristics of an RDF data D and to evaluate and measure the space efficiency of RDF data storage and indexing techniques.

1. $|D|$: the total number of triples in D ;
2. $|pred(D)|$: the number of unique properties in D ;
3. $|val(D)|$: the number of unique values in D ;
4. $|cls(D)|$: the number of classes in D ;
5. $avgPred(D)$: the average number of properties belonging to the same class in D ; and
6. $|lft(D)|$: the number of triples whose subjects do not belong to any class or whose predicates are multi-value predicates.

As the values of (2), (3), (4) and (6) partially depend on the value of (1), indeed, it is the ratio of these values to $|D|$ that truly describes the data distribution of an RDF data. Also note that there is correlation between $|D|$, $|pred(D)|$, $|cls(D)|$ and $avgPred(D)$. Even though there is no direct functional relationship between them, but given three, a tight up/lower bound is set on the value the fourth can take.

2.4 Data Characteristics in RDF Benchmarks

We have identified 6 key factors of RDF data that have significant impact on the storage efficiency of RDF data storage and indexing techniques. We believe insightful comparison of such techniques should be conducted on data-sets in which all the key factors vary.

The data characteristics that are covered by the existing benchmarks that are used in the research work of RDF data storage and indexing is summarized in the table on the right.

Benchmark	$ pred(D) $	$ val(D) $	$ D $	Used in
Barton [5]	285	19M	50M	[4][11][15][18]
LUBM [2]	18	1M	7M	[18]
Yago [16]	93	34M	40M	[11]
LibraryThing [1]	338824	9M	36M	[11]

As the schema information is frequently absent from these benchmarks, we do not summarize the schema related data characteristics, namely $|cls(D)|$, $avgPred(D)$ and $|lft(D)|$, in the table.

2.5 Empirical Analysis

To provide a thorough understanding of the storage methods, and answer the question about exactly what type of RDF data each storage method is best/worst at, we generate and conduct experiments on synthetic data sets, big and small, with various combination on the key factors we identified in Sec. 2.3. The trend we observe are the same. In this section, we present our results on the comparison

based on multiple data sets with a fixed number of triples (100,000 triples to be specific), but vary on other characteristics.

In order to better understand the correlation between $|pred(D)|$ and storage efficiency of RDF repository that highlights the advantage and drawback of the PT method, we design our RDF data set such that all triples fit in property tables and the leftover table is empty. Since a leftover table is nothing but a triple store table of the residential RDF triples, the storage efficiency of the PT methods on data that yield non-empty leftover table can be easily estimated by integrating the analysis and observations of both PT and TS methods.

We use MySQL to implement the storage methods discussed in Sec. 2.1 and Sec. 2.2. Specifically, we use relational tables and B⁺-tree indices to implement the three index-based storage methods discussed in Sec. 2.2, by storing the RDF triples in their proper clustering order in relational tables and create B⁺-tree indices on top of these tables, hence the key concepts and features of the original methods are loyally preserved.

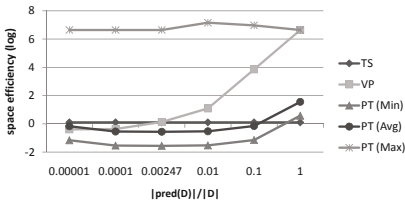


Fig. 4. Space Efficiency Comparison: Data Storage Methods

Data Storage Methods. The impact of the number of unique predicates ($|pred(D)|$) on the space efficiency of the data storage methods is shown in Fig. 4. As the value of $|pred(D)|$ is strongly correlated to the number of triples in an RDF document, we use the ratio, $\frac{|pred(D)|}{|D|}$ as the parameter. Please note that logarithmic scale is used on the y-axis, due to the large difference exhibited by different methods.

Reflecting our analysis presented in Sec. 2.1, TS is indifferent to $|pred(D)|$.

SE_{VP}^D is heavily affected by $|pred(D)|$ because the larger $|pred(D)|$ is, the greater the overhead for storing the schema info of the tables would be. VP, originally designed to improve the space efficiency of TS, can end up to be not efficient, even very inefficient, when the overhead is driven up by large number of unique predicates, cancelling out the saving of not storing the predicate value in those tables.

The impact of $|pred(D)|$ varies on PT, in which other factors play more important roles on the space efficiency. In Fig. 4,

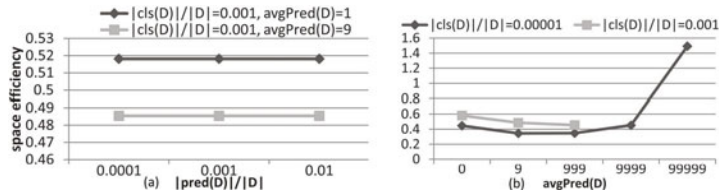


Fig. 5. Storage Efficiency Analysis: Property Table

$PT(Max)$, $PT(Avg)$ and $PT(Min)$ represent the maximum, average and minimum storage efficiency we obtained on various RDF data sets that share the same $|pred(D)|$. We then investigate the impact of other factors, including the number of classes ($|cls(D)|$) and average number of predicates per class ($avgPred(D)$), on PT method.

As seen in Fig. 5(a), indeed $|pred(D)|$ does not have any direct impact on SE_{PT}^D . It only defines what the values of $avgPred(D)$ and $|cls(D)|$ may be.

For RDF documents with the same $|pred(D)|$, SE_{VP}^D are different when $avgPred(D)$ and $|cls(D)|$ are different. When $|D|$ and $|cls(D)|$ are fixed, the direct impact of $avgPred(D)$ on SE_{PT}^D is illustrated in Fig. 5(b). The curve reflects the trade-off between the save in data storage by introducing wider property tables and the overhead in storing the schema info of all the columns in the wider tables.

Index-based Storage Methods. We store RDF data of various $|val(D)|/|D|$ ratio using the three index-based storage methods. Our experimental results, as shown in Fig. 6 confirm our analysis. As summarized in Sec. 2.2, space efficiency wise, the difference between MAP and Hexastore lies in the size of the index tree and index leaf nodes. Hence, Hexastore is always more space efficient than MAP. TripleT distinguishes itself from MAP and Hexastore by indexing only unique values. Therefore, the unique number of values in the RDF data is the dominating factor of the space efficiency of TripleT. In addition, as it stores each triple only three times, in most cases it is more efficient than MAP and Hexastore. However, it becomes less efficient when the number of unique values increases.

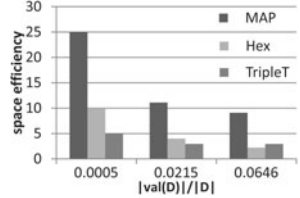


Fig. 6. Storage Efficiency Comparison: Indexing

3 Query Evaluation

As a matter of fact, data storage and indexing techniques are invented to facilitate efficient query evaluation. Hence, besides space efficiency, query efficiency is of ultimate importance.

3.1 Query Patterns

SPARQL [9], recommended by W3C, is the de facto standard RDF query language. A SPARQL query consists of one or more graph patterns, the evaluation of which is based on graph pattern matching against the RDF data graph. Let \mathcal{L} be a finite set of literals, \mathcal{U} a finite set of URIs and \mathcal{V} a finite set of variables. Then an *RDF triple* is in the set $\mathcal{U} \times \mathcal{U} \times (\mathcal{U} \cup \mathcal{L})$ and a *simple triple pattern* (STP) in a SPARQL query is in the set $(\mathcal{U} \cup \mathcal{V}) \times (\mathcal{U} \cup \mathcal{V}) \times (\mathcal{U} \cup \mathcal{V} \cup \mathcal{L})$. Please note that a variable can appear in the role of subject, predicate, or object.

We list 10 different simple triple patterns based on the number and roles of variables in

s,type, ?o	s, p, ?o	s, ?p, o	?s, p, o	?s, type, o
?s,type,?o	?s, p, ?o	?s, ?p, o	s, ?p, ?o	?s, ?p, ?o

a pattern. Specifically we distinguish “type” from other predicates, because matching the pattern $(?s, type, o)$ is very different from $(?s, p, o)$ in Property Table as the knowledge of the class that $?s$ belongs to can determine the property table to search in, while the knowledge of the constant p can not have the same filtering effect on the optimization process.

A *graph pattern* (GP) is a non-empty set of STPs. We define a connected graph pattern recursively as follows.

Definition 2. A graph pattern with single STP is connected. If two connected graph patterns, g_1 and g_2 , share a common variable or URI, we say that the graph pattern $g_3 = g_1 \cup g_2$ is also connected.

Please note that our classifications of whether a graph pattern is connected merely depends on whether the STPs share variables or URIs, not whether the graph pattern, when represented as a graph, is a connected graph. For instance, in our classification, $\{s1, ?p, o1. s2, ?p, o2\}$ is a connected graph pattern.

We call a connected graph pattern *single joint graph pattern* (SJGP) if it has exactly two STPs. Based on the roles that the shared variable takes in these STPs, we can identify six types of joins.

Definition 3. Given a SJGP, $g = \{stp_1, stp_2\}$, where $stp_1 = (s_1, p_1, o_1)$ and $stp_2 = (s_2, p_2, o_2)$. We say that g is formed by S-S join if $s_1, s_2 \in \mathcal{V}$ and $s_1 = s_2$. Similarly we can define O-O, S-O, P-P, S-P, and O-P joins.

Join type	Example
S-S	?s p ₁ o ₁ . ?s p ₂ o ₂
O-P	s ₁ p ₁ ?x. s ₂ ?x o ₂
O-O	s ₁ p ₁ ?o. s ₂ p ₂ ?o
P-P	s ₁ ?p o ₁ . s ₂ ?p o ₂
S-O	?x p ₁ o ₁ . s ₂ p ₂ ?x
S-P	?x p ₁ o ₁ . s ₂ ?x o ₂

A graph pattern may consist of multiple STPs, multiple variables and multiple types of joins. We call them Complex Join Patterns (CJP). To better understand how CJPs can be evaluated by different storage methods, it would be beneficial to first understand how many types of CJPs there are.

[12] defined chain shape patterns (CSP) as a set of STPs linked together via S-O joins and star shape pattern (SSP) as a set of STPs linked together via S-S joins. They proved that all data storage approaches do not favor queries with CSP and PT favors queries with SSP.

However, CSP and SSP as defined in [12] represent only a very small fragment of SPARQL queries. In this paper we propose a more sophisticated classification based on the positions of variables in a graph pattern, which extends the concepts of CSP and SSP by (1) considering all positions of variables, i.e. S, P and O; and (2) considering all types of joins besides S-S join in SSP and S-O join in CSP.

Definition 4. Given a connected graph pattern gp that consists of more than two STPs, we say that gp is:

- an Extended Chain-shaped Pattern (ECP) if no more than two STPs in gp share a common variable;
- an Extended Star-shaped Pattern (ESP) if all STPs in gp share at least one common variable;
- a Hybrid Pattern (HP) if gp does not fall into the above categories.

The graph pattern in the query we presented in Sec. II, $\{?person foaf A . ?person ?p ?person. D ?p E \}$, is an ECP, featuring a S-S join and a P-P join.

3.2 Query Patterns in Benchmarks

The design of the query set in a benchmark plays a critical role in testing how efficiently a data management system can handle various types of queries. We summarize the query patterns (STP, SJP and CJP) that are covered by existing benchmarks.

Our observations are:

- (1) Queries with variables in the predicate role are not well explored.

When the lone query with ?p in Barton was used, it was applied to an RDF data with only 28 unique predicates [4]. (2) The queries in each benchmark cover at most three different SJPs, and none of them feature any join that involves a variable in the predicate role. (3) The queries in these benchmarks are as complex as SSP and CSP, but none of the benchmark features any extended join patterns we defined, which, as shown in our motivating example in Sec. 1 is a typical query that appear in real life applications.

Benchmark	STP	SJP	CJP	Used in
Barton [5]	?s type o, ?s type ?o, ?s p o, ?s p ?o, ?s ?p ?o	S-S, S-O	SSP, CSP	[4][11][15][18]
LUBM [2]	?s p o, ?s p ?o	S-O, O-O	None	[18]
Yago [16]	?s type o, ?s p o, ?s p ?o, ?s ?p ?o	S-S, S-O, O-O	SSP, CSP	[11]
LibraryThing [1]	?s p o, ?s p ?o	S-S, S-O, O-O	SSP	[11]

3.3 Empirical Study

To better understand how the RDF data storage and index-based storage techniques proposed in the literature stand up to the challenges of the important types of query patterns, we designed a set of queries to stress the systems.

The data characteristics of our synthesized data-set is shown on the right. This data set is chosen

D	pred(D)	val(D)	cls(D)	avgPred(D)	lft(D)
10M	900	1.7M	100	10	0

as it does not favor one data storage and/or index-based storage method over another, in terms of space efficiency. |lft(D)| is 0, in order to weed out the impact of the leftover table when we evaluate the query evaluation performance of PT.

We have tested on large number of queries, STPs, SJPs, and CJPs, of various value/join selectivity and result cardinality. We pick the queries on the right to illustrate our analysis and observations. The class a query belongs to is reflected in its name. Query patterns and the result cardinalities of these queries are also provided.

Query	Pattern	Result
STP1	?s p o	1K
STP2	?s type o	1K
STP3	s ?p o	1K
STP4	s ?p o	22
STP5	?s ?p o	1K
SJP1	?x p1 o1. ?x p2 o2	1K
SJP2	s1 p1 ?x. s2 ?x o1	1K
ESP1	s1 ?x o1. s2 ?x o2. s3 ?x o3	100
ECP1	s1 p1 ?x. ?x ?y o1 . s2 ?y o2	100

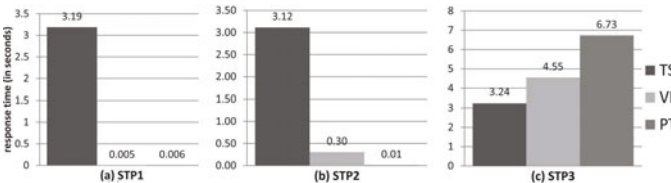


Fig. 7. STP - Data Storage Methods

Simple Triple Patterns (STP).

We first study how different data storage methods fair answering STP queries. We focus on the missing pieces in the literature and

investigate the impact of the certainty of the values in the predicate role on the evaluation of STPs. As shown in Fig. 7(a), for STP1, VP outperforms TS and PT, since in VP, the search is limited to only the table corresponding to the given predicate, while the search in TS involves the full TS table, and the search in PT involves multiple tables, corresponding to classes that feature the given predicate. In comparison, PT performs better when only one property table can be identified, i.e. STP2, with the combination of “type” in the predicate role and a constant in the object role, as shown in Fig. 7(b). This impact is magnified when a variable is on the predicate role, i.e. STP3, as illustrated in Fig. 7(c). For this type of queries, TS outperforms VP and PT thanks to its simple schema, as in both VP and PT, all tables have to be searched to answer the query.

We then study how the indices facilitate the evaluation of STP queries. TripleT is penalized when it evaluates STPs with constants on two roles, i.e. STP3, as

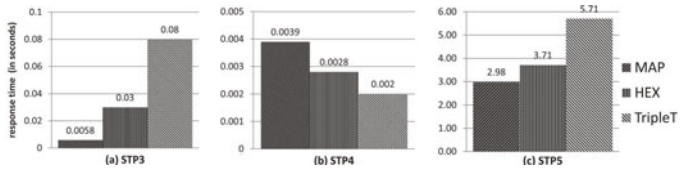


Fig. 8. STP - Index-Based Storage Methods

it only indexes on one value, rendering many pointers to follow into its complicated payload after searching in the B⁺-tree structure, while MAP and Hexastore are both capable of immediately locate the query results. The dramatic difference is shown in Fig. 8(a). When the constants given in the STP are rare, i.e. STP4, the benefit of the light-weight B⁺-tree index of TripleT ensures that it outperforms both MAP and Hexastore, as shown in Fig. 8(b). The penalty is not as severe when there is only one constant, no matter what role it takes in the STP, i.e. STP5, as shown in Fig. 8(c).

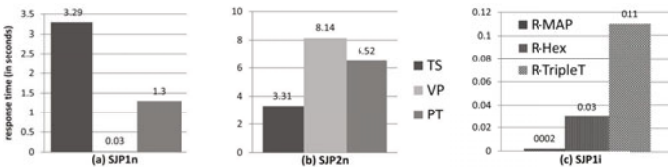


Fig. 9. Query Performance: SJP

Simple Join Patterns (SJP). For SJP queries, when there are no other variables in the STPs, the dominating factors for query efficiency are

indeed those which dictate the efficiency for answering each single STP query, as can be observed from the comparison between the evaluation of SJP1 (Fig. 9(a)), which features an S-S join, and SJP2 (Fig. 9(b)), which features a P-O join. Index-based storage methods are critical for improving the performance of SJP queries, as MAP, Hexastore and TripleT are all indifferent to the role of the variable in an STP and INLJ (index nested loop join) is frequently a good choice when indices are available. The improvement can be seen by comparing Fig. 9(a) and Fig. 9(c) (please note the difference in the scales on the y-axis).

Complex Join Patterns (CJP).

As to the CJP queries, we focus on the extended star and chain shaped patterns (ESP and ECP) we identified, especially the patterns that were not studied in the literature before, for example, ESP1, a star pattern that join on predicate role, and ECP2, a chain shaped pattern involving a P-P join.

As can be observed from Fig. 10(a), on ESP1, VP outperforms TS and PT. The reason is that when the predicate is unknown, in both TS and PT, all data has to be searched then joined. However, in VP, to yield the final results, triples in one table only need to join with other triples in the same table. In other words, the vertical partitioning serves as a pre-hashing. As it could be seen in Fig. 10(b), VP and PT slightly outperform TS, as they both take advantage of the known predicate to narrow down the search in evaluating one STP. However, due to the uncertainty introduced by the variable in the predicate roles, their advantage over TS is not as obvious as if the query is a CSP with features only S-O joins. As the index-based storage methods are indifferent to the the roles of the variable, adding more STPs and more joins only intensify what we have observed in the SJP cases.

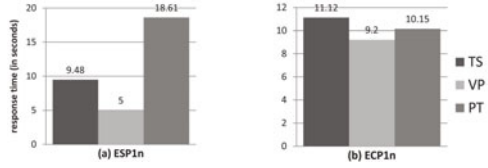


Fig. 10. Query Performance: CJP

4 Conclusion

Based on the in-depth study of the RDB-based RDF data storage and indexing methods, we introduce a set of key data characteristics based on which we compare the storage efficiency of these methods. We study SPARQL queries, introduce new ways to classify patterns based on the locations of the variables, and compare and analyze the query evaluation efficiency of the storage methods, focusing on the query patterns that were not in the RDF benchmarks and not reported in the literature.

Our empirical evaluation testify the law of engineering design: there is no one-size-fit-all method — each method is superb only for certain types of data and certain types of queries. Our study also illustrates the insufficiency of existing RDF benchmarks for providing thorough and in-depth measurement of RDF data management and query evaluation techniques. We believe that our analysis and findings presented in this paper will serve as a guideline for designing better RDF benchmarks, which is indeed what we plan to pursue in the wake of completing this project.

References

1. Librarything data-set, <http://www.librarything.com/>
2. LUBM data-set, <http://swat.cse.lehigh.edu/projects/lubm/>

3. Resource Description Framework (RDF). Model and Syntax Specification. Technical report, W3C
4. Abadi, D., et al.: Scalable Semantic Web Data Management Using Vertical Partitioning. In: VLDB, pp. 411–422 (2007)
5. Abadi, D., et al.: Using The Barton Libraries Dataset As An RDF Benchmark. Technical Report MIT-CSAIL-TR-2007-036, MIT (2007)
6. Brickley, D., et al.: Resource Description Framework (RDF) Schema Specification. W3C Recommendation (2000)
7. Broekstra, J., et al.: Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, pp. 54–68. Springer, Heidelberg (2002)
8. Fletcher, G., et al.: Scalable Indexing of RDF Graphs for Efficient Join Processing. In: CIKM, pp. 1513–1516 (2009)
9. Furche, T., et al.: RDF Querying: Language Constructs and Evaluation Methods Compared. In: Barahona, P., Bry, F., Franconi, E., Henze, N., Sattler, U. (eds.) Reasoning Web 2006. LNCS, vol. 4126, pp. 1–52. Springer, Heidelberg (2006)
10. Harth, A., et al.: Optimized Index Structures for Querying RDF from the Web. In: LA-WEB, p. 71 (2005)
11. Neumann, T., et al.: RDF-3X: a RISC-style Engine for RDF. Proc. VLDB Endow. 1(1), 647–659 (2008)
12. Neumann, T., et al.: Scalable Join Processing on Very Large RDF Graphs. In: SIGMOD, pp. 627–640 (2009)
13. Neumann, T., et al.: The RDF-3X engine for scalable management of RDF data. VLDB J. 19(1), 91–113 (2010)
14. Schmidt, M., et al.: SP 2 Bench: A SPARQL performance benchmark. In: ICDE, pp. 222–233 (2009)
15. Sidirourgos, L., et al.: Column-store Support for RDF Data Management: Not all Swans are White. Proc. VLDB Endow. 1, 1553–1563
16. Suchanek, F., et al.: YAGO: A Core of Semantic Knowledge - Unifying WordNet and Wikipedia. In: WWW, pp. 697–706 (2007)
17. Wang, Y., et al.: FlexTable: Using a Dynamic Relation Model to Store RDF Data. In: Kitagawa, H., Ishikawa, Y., Li, Q., Watanabe, C. (eds.) DASFAA 2010. LNCS, vol. 5981, pp. 580–594. Springer, Heidelberg (2010)
18. Weiss, C., et al.: Hexastore: Sextuple Indexing for Semantic Web Data Management. Proc. VLDB Endow. 1, 1008–1019 (2008)
19. Wilkinson, K., et al.: Jena Property Table Implementation. In: SSWS (2006)

Requirement-Based Query and Update Scheduling in Real-Time Data Warehouses*

Fangling Leng, Yubin Bao, Ge Yu, Jingang Shi, and Xiaoyan Cai

School of Information Science & Engineering,
Northeastern University, Shenyang 110819, P.R. China
lengfangling@mail.neu.edu.cn

Abstract. A typical real-time data warehouse continually receives read-only queries from users and write-only updates from a variety of external sources. Queries may conflict with updates due to the resource competition and high loads. Moreover, users expect short response time for queries and low staleness for the query results. This makes it challenging to satisfy the two requirements simultaneously. This paper proposes a requirement-based querying and updating scheduling algorithm (RQUS) which allows users to express their real needs for their queries by specifying the acceptable response time delay and the acceptable result staleness when queries are submitted. RQUS dynamically adjusts the work mode of the system according to the changing requirements of users in order to allocate system resource to queries or updates and then prioritizes the query or update queue according to the work mode. And a freshness monitor is adopted to monitor the execution state of updating tasks in order to maintain the global table incrementally. Experimental results show that RQUS algorithm performs better than the three traditional scheduling algorithms with the changing user requirements overall.

1 Introduction

Real-time data warehouses, RTDWH for short, emerge due to the requirements of up-to-date data in enterprises for real-time decision support. RTDWH uses the push-based data integration. That is to say, it is automatically captured, transformed into unified format, and then loaded into RTDWH whenever new data is produced by OLTP from a variety of external sources. Thus, continuous flow of read-only queries and write-only updates may conflict with each other. Moreover, in the applications of RTDWH, users expect short response time for their queries and low staleness. However, it may be extremely difficult to meet the two criterion simultaneously especially in the period of high load. But, we have observed that some users can accept response time delay to obtain lower result staleness and others can accept result staleness in order to get result faster.

* Supported by the National Natural Science Foundation of China under Grant No.61033007 and the Basic Scientific Research Foundation of the Ministry of Education under Grant No.N100304005.

User preferences have been discussed in [12]. In [1], users need to assign the profit values to express their preferences. But the profit is hard to be decided. Users are allowed to express their relative preferences instead of real needs about Quality of Service and Quality of Data and the sum of Quality of Service and Quality of Data is one [2]. Thus, these can't reflect the real needs about response time and staleness. In RTDWH, query tasks consist of short and long ones. In [3], the deadline is not considered. [4] proposes and discusses several distributed query scheduling policies that directly consider the available cache contents by employing distributed multidimensional indexing structures and an exponential moving average approach to predicting cache contents, but not considers the updating. These give raise to a big problem of how to schedule the two kinds of tasks reasonably that compete for resources and address the scheduling between long and short query tasks in order to satisfy the real requirements of users well. This is exactly the problem we study in this paper.

The main contributions of our work can be summarized as follows:

- (1) We introduce the concepts of acceptable response time delay and acceptable result staleness in order to allow users express their real needs about the query tasks they submitted.
- (2) We use the freshness monitor to monitor the execution state of update tasks and to modify the partition freshness of the global table when an update task on a partition is executed completely in order to reduce the complexity of recalculating the partition freshness.
- (3) We develop a two-level scheduling algorithm with the real requirements of users. It not only solves the problem of prioritizing the scheduling query and update tasks, but also uses absolute deadline to schedule short and long query task to reduce the deadline miss ratio effectively.

To evaluate the performance of RQUS, we conducted a set of experiments using TPC-DS schema. Then we compare RQUS to three traditional scheduling algorithms according to four metrics. The three traditional scheduling algorithms include: First in Fist out (FIFO), FIFO Update High (FIFO-UH) and FIFO Query High (FIFO-QH).

The paper is organized as follows. Section 2 discusses the related work. We describe the system model in Section 3. The two-level scheduling algorithm RQUS is introduced in detail in Section 4. Then, in Section 5, we describe experimental setup and present our experimental results. Finally, we conclude the paper in Section 6.

2 Related Work

The topic of scheduling algorithms has been discussed extensively in the research community and a variety of works exist. Scheduling algorithms include query scheduling, update scheduling as well as query and update scheduling. And Qu et al. [15] proposed an adaptive algorithm called QUTS in data-intensive web applications. QUTS allows users to express their references for the expected

QoS and QoD of their queries by assigning the potential economic benefit and the number of unapplied updates. But it is hard for users who lack economic approaches to assign right profit for queries. Although unapplied updates exist corresponding to a query, they don't lead to staleness if they aren't in the range of query results. However, RQUS we propose in this paper refer to the two-level scheduling structure of QUTS.

Thiele et al. proposed a workload scheduling WINE in RTDWH, where the user requirements were specified for each query according to a simple vote in [2]. It simply considered the demand trend of users about the two transactions instead of the real needs of users. So, in fact, it can't meet the requirements of the users well.

The importance of query and update tasks is considered by PBBS, which is a priority-based balance scheduling, proposed by Shi et al. [6]. PBBS sets priorities to query and update tasks separately, that is, the importance of the two types of tasks isn't related to each other. However, in real-time data warehouses, the priority of an update task is determined by query tasks due to the quality of query results is affected by unapplied updates that belong to a table the query accesses. In this paper, we set the priorities of update tasks according to the related priorities of query tasks.

There are a lot of researches that discuss scheduling of queries or updates [3,7,8]. However, in RTDWH, the scheduling of queries and updates both need to be considered because users demand on the response time and result staleness. Hard real-time scheduling is studied extensively [9], but it isn't suitable for RTDWH because tasks that miss deadlines can't be discarded.

3 System Model

In this paper, we consider a real-time data warehouse that receives write-only update tasks and read-only query tasks. Fig.1 illustrates the system model of

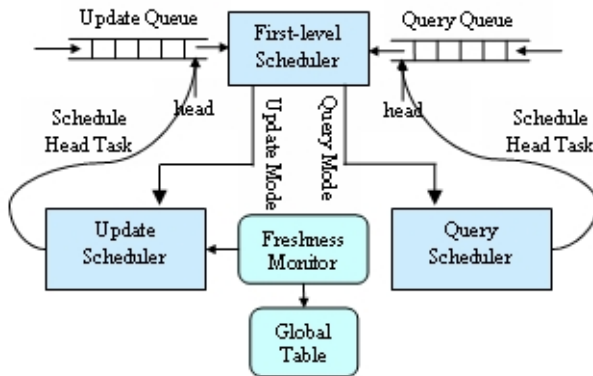


Fig. 1. System model

our proposed requirement-based query and update scheduling algorithm called for RQUS. The system consists of five kinds of components: two task queues (update queue and query queue), a first-level balance scheduler, two second-level schedulers, a freshness monitor, and a global table.

3.1 System Components

The two task queues receive write-only update tasks and read-only query tasks respectively. The update queue is denoted as UQ and the query queue is denoted as QQ. The position of the head task of each queue is 0. The first-level balance scheduler is responsible for deciding the execution mode of the current system. According to the result of the first-level balance scheduling, system enters the second-level scheduling stage. If the system mode is query, the query scheduler will run. It schedules the head query task to execute because query queue is sorted when new query task arrives. If the system mode is update, the update scheduler will run. It prioritizes, sorts the update queue and then schedules the head update task to execute. The global table stores the freshness information of each partition. The freshness monitor is used to monitor the execution of update tasks and modify the global table when update tasks are processed completely in order to avoid recalculating the freshness of the partition.

3.2 Task Model

In our system, there are two kinds of tasks: update tasks and query tasks. Update and query tasks are submitted to the system respectively and then enter the update queue and query queue separately. A task will be deleted when it is executed completely.

To represent the arrival time, each update task comes with a timestamp ($t_a(u_i)$) and each query task is associated with a timestamp ($t_a(q_i)$), too. In our system, in order to allow users to express their real needs, each query task is provided with two parameters, acceptable response time delay Δt_{rl} and acceptable result staleness Δs , when it is submitted by users. Then our system can schedule tasks according to these parameters so that it can satisfy the requirements of users well.

3.3 Performance Metrics

In order to measure the performance of our scheduling algorithm, we assume a set of update tasks and query tasks in a period of time and the performance is measured during the execution of tasks. Furthermore, we introduce three metrics for it. They can be classified into two classes: Quality of Service (QoS) metric and Quality of Data (QoD) metric. They are described as follows.

- 1) QoS metric
 - a) Deadline miss ratio (DMR)

DMR denotes the ratio of the number of query tasks that miss their deadlines to the number of query tasks that have already been executed. The calculation formula is as follows:

$$DMR = \frac{|Q_{dm}|}{|Q_e|} \quad (1)$$

Where, Q_{dm} is the set of query tasks that miss their deadlines, Q_e represents the set of query tasks that have already been executed, $|Q_{dm}|$ is the number of Q_{dm} and $|Q_e|$ is the number of Q_e .

2) QoD metric

a) Unacceptable result staleness ratio (URSR)

To describe the satisfaction degree of the result staleness, we introduce the URSR metric. It is the ratio of the number of queries of which the result staleness is unacceptable to the number of queries that have already been executed. The calculation formula is as follows:

$$URSR = \frac{|Q_{urs}|}{|Q_e|} \quad (2)$$

Where, Q_{urs} is the set of query tasks of which the result staleness is unacceptable, Q_e represents the set of query tasks that have already been executed, $|Q_{urs}|$ is the number of Q_{urs} and $|Q_e|$ is the number of Q_e .

b) Average Staleness (AS)

We firstly give the definition of freshness of the partition p_i as the maximum timestamp of all records in p_i denoted as $F(p_i)$ (similar definition has appeared in the real-time stream warehouse literature [7]). In our paper, we calculate data staleness according to the partition unit. We define the staleness of a partition p_i as the difference between the maximum arrival timestamp of each unapplied update on p_i and the freshness of p_i :

$$S(p_i) = \max(t_a(u_j), p_i) - F(p_i) \quad (3)$$

Where, $S(p_i)$ is the staleness of the partition p_i and $\max(t_a(u_j), p_i)$ is the maximum arrival timestamp of each unapplied update on p_i . Because a query only concerns about the update tasks that arrive before it, so the staleness of a partition p_i related to a query q is changed as follows:

$$S(p_i, q) = \max_{t_a(u_j) \leq t_a(p_i)} (t_a(u_j), p_i) - F(p_i) \quad (4)$$

For a query task q , the staleness $S(q)$ is the maximum staleness of all partitions p_q that affect query results:

$$S(q) = \max_{p_i \in p_q} (S(p_i, q)) \quad (5)$$

We use the average staleness of query results AS in current system to describe the impact of our scheduling algorithm. Let Q_e be the set of already processed query tasks in the system and $|Q_e|$ be the number of Q_e . AS is calculated as follows:

$$AS = \sum_{q_i \in Q_e} \frac{S(q_i)}{|Q_e|} \quad (6)$$

4 Scheduling Algorithm

In this section, we will illustrate our scheduling algorithm RQUS. It consists of the first-level scheduling that is to support fair schema according to the user requirements and the second-level scheduling that schedule tasks to execute after a set of prioritizing and sorting. In the second-level scheduling, the head query task will be scheduled to execute due to a sorted query queue if the mode is query and if the mode is update, update queue will be prioritized and sorted and then scheduling the head task to be processed according to the result of the first-level scheduling. In the application of real-time data warehouses, users may change their requirements with the time going, that is, they have different needs for their queries about the response time delay and result staleness at different times. RQUS can perform very well in this context of changing user requirements due to the adjustability.

4.1 First-Level Scheduling

The purpose of the first-level scheduling is to determine the execution mode of current system. More precisely, system is in the state of update or query. As previously motioned, users express their requirements according to assigning the acceptable response time delay Δt_{rl} and acceptable result staleness of the queries Δs they submitted. In the first-level scheduling, we use response delay ratio R_{rl} and result staleness ratio R_S to show that which kind of tasks deviates from the user to a larger extent. The system will enter query scheduling if $R_S \leq R_{rl}$ or update scheduling if $R_S > R_{rl}$. Moreover, R_S and R_{rl} are recalculated whenever an update or query task is processed completely. Then we introduce response delay ratio and result staleness ratio respectively.

1) Response delay ratio (R_{rl})

The response delay ratio denotes the sum of ratios between real response time delay t_{rl} and acceptable response time delay Δt_{rl} . In this paper, we assume that the smallest time unit is 1. Due to the acceptable zero latency, we should add 1 to denominator. Then R_{rl} is

$$R_{rl} = \sum_{q_i \in Q_e} \frac{t_{rl}(q_i)}{\Delta t_{rl}(q_i) + 1} \quad (7)$$

In the above formula, $t_{rl}(q_i)$ is the difference between the scheduled time $t_s(q_i)$ and the arrival time $t_a(q_i)$ of a query q_i . Then more detailed computation method of R_{rl} is

$$R_{rl} = \sum_{q_i \in Q_e} \frac{t_s(q_i) - t_a(q_i)}{\Delta t_{rl}(q_i) + 1} \quad (8)$$

2) Result staleness ratio (R_S)

The result staleness ratio is the sum of ratios between the real result staleness s and the acceptable result staleness Δs . Similar to acceptable zero latency, acceptable zero result staleness may also exist. The calculation formula is

$$R_s = \sum_{q_i \in Q_e} \frac{s(q_i)}{\Delta s(q_i) + 1} \quad (9)$$

4.2 Second-Level Scheduling

According to the result of the first-level scheduling, the execution mode of the current system is determined. The system schedules query queue or update queue.

1) Query Scheduling

The priority is determined by the distance between the current moment and the deadline in EDF. EDF uses relative deadline and doesn't consider the execution time of tasks. A task meets its deadline once it is scheduled to be executed before its deadline.

However, in real-time data warehouses, short and long query tasks exist at the same time. If a long query task with higher priority before several short query tasks with lower priorities is scheduled, the short query tasks may miss their deadline that the users specify. In our query scheduling algorithm, we use absolute deadline (AD) to determine the priorities of tasks. This scheduling algorithm can effectively solve the scheduling problem between short and long queries with deadline and get a lower DMR than EDF. For a query q_i with execution time $t_e(q_i)$, AD is calculated as follows:

$$AD(q_i) = t_a(q_i) + \Delta t_{rl}(q_i) + t_e(q_i) \quad (10)$$

With AD, the task misses its deadline if the difference between the time that the task is processed completely and its AD is greater than zero. Our query scheduling algorithm avoids starvation effectively because the AD of each query is a period of finite time. Before a new query task enters the query queue, system firstly computes the AD in order to get the priority and then inserts it into the right position in the order of the priority. The relative order of query tasks

doesn't change only when a new query task is inserted. Thus our query scheduling algorithm not only gets low DMR, but also reduces the complexity effectively.

2) Update Scheduling

The priority of an update task is related to two parameters: the acceptable result staleness and the locations of the queries that are affected by the update task. As mentioned before, an update task has correlation with a query task if the partitions that the update task affects and the partitions that the query task accesses have a same partition at least.

There are many update tasks influencing the query result. But every update affects the query to a different degree. Due to the acceptable result staleness assigned by users, some unapplied update tasks are accepted by users. As a result, the update tasks with large timestamps have higher priorities. Thus the order of the execution of update tasks on a partition is deviated from their arrival timestamps. Therefore, the priorities of update tasks are calculated according to the partition unit. Update tasks on a partition have the same priority and is scheduled in the increasing order of the arrival timestamps. In a word, update tasks are scheduled in decreasing order of priorities and in increasing order of arrival timestamps. To define the priority of a partition, we introduce the concept of staleness distance dis between the partition p_j and q_i :

$$dis(p_j, q_i) = s(p_j) - \Delta s(q_i) \quad (11)$$

Next, we give the definition of dis penalty function.

$$M_i = \begin{cases} 0 & , \quad dis \leq 0 \\ dis & , \quad dis > 0 \end{cases} \quad (12)$$

That is, the freshness of a partition satisfies the user requirements if $dis \leq 0$ and the freshness of a partition does not satisfy the user requirements if $dis > 0$. Now, the priority of a partition p is calculated as follows:

$$P(p) = \sum_{(p_{q_i} \cap p \neq \emptyset) \cap (q_i \in Q_p)} \frac{M_i}{loc_{q_i} + 1} \quad (13)$$

Where, Q_p is the set of query tasks accessing the partition p and loc_{q_i} is the location of a query task q_i which belongs to Q_p .

In a word, we firstly calculate the priority of each partition to get the priority of each update task, and then sort update queue by the priority and arrival timestamp of each update task in our update scheduling algorithm.

5 Experiments

We have done a set of experiments to compare the performance of our proposed requirement-based query and update scheduling algorithm RQUS to the performance of others scheduling algorithms mentioned before, including FIFO, FIFO-UH, FIFO-QH. Results show that RQUS outperforms all baseline algorithms in the entire spectrum of user requirements and tasks.

5.1 Experimental Setup

All experiments are carried out on a 3.16GHz Intel(R) Core(TM)2 Duo CPU with 4GB RAM under Windows XP and ORACLE 11g DBMS. We implemented RQUS and the FIFO, FIFO-UH, FIFO-QH scheduling algorithms using Java 1.6. Query and update SQL statements is from TPC-DS decision support benchmark (TPC-DS) [10]. We generate a set of 200 queries and 200 updates and they are rewritten due to using partitions in this paper. Query execution time ranges from 100 to 1,000 milliseconds and update execution time ranges from 20 to 50 milliseconds. The acceptable response time delay ranges from 100 to 1500 milliseconds and the acceptable result staleness ranges from 10 to 100 milliseconds. We use the real execution time for queries. The system includes three loads: high, medium and low. The arrival numbers of queries in a second under high, medium and low loads are two, five and ten, respectively. The arrival numbers of updates in a second under high, medium and low loads are five, eight and twelve, respectively.

5.2 Performance Comparison

In Fig.2, we compare the four scheduling algorithms using the deadline miss ratio (DMR) metric. We see that RQUS performs better than others under different workloads. It is easily to understand that RQUS outperforms FIFO and FIFO-UH. Although FIFO-QH favors queries than updates, it doesn't consider that long and short queries exist simultaneously and queries are with deadlines. RQUS considers both. Moreover, the execution time of an update is so much shorter than that of a query. RQUS performs a little better than FIFO-QH under different loads.

Fig. 3 illustrates the AS metrics of the four algorithms under different loads. The AS of FIFO-UH is zero because it favors updates than queries all the time. That is to say, there are no updates when a query task is being scheduled to

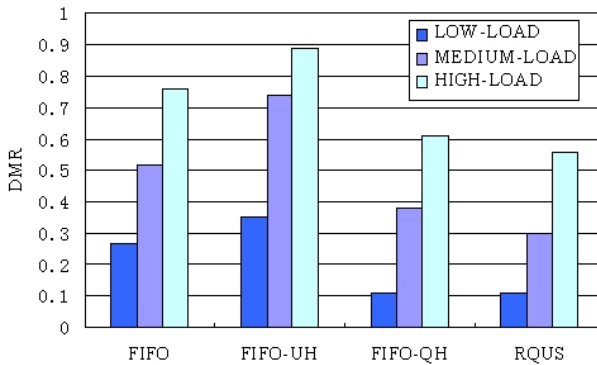


Fig. 2. DMR performance

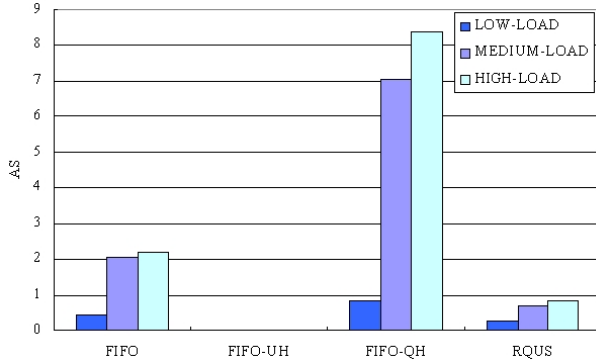


Fig. 3. AS performance

be executed. The AS of RQUS is second only to that of FIFO-UH. FIFO performs as well as RQUS under the low load. But FIFO performs much worse than RQUS under medium and high loads because FIFO doesn't consider the acceptable query result staleness of users. RQUS will execute update tasks when the acceptable query result staleness is being unsatisfied. The AS of FIFO-QH is bigger and bigger and FIFO-QH performs worst than others with the increasing load. It can be seen that RQUS can improve the satisfaction of users with the result staleness effectively.

Fig.4 shows the URSR metrics for all the four scheduling algorithms under different loads. It is similar to Fig.2. FIFO-UH is always performing better than others due to the inherent characteristic. The URSR metric of RQUS is performs a little better than that of FIFO under the low load. However, with the increasing load, RQUS outperforms FIFO greatly and FIFO is stale. RQUS performs a little worse than FIFO-UH and RQUS is stale when the load is to an extent.

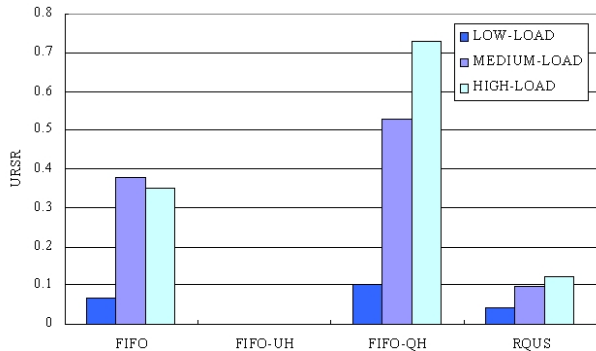


Fig. 4. URSR performance

From the above, we can conclude that RQUS reduce the DMR and URSR of queries effectively. That is to say, RQUS can balance the two metric (QoS and QoD) under all kinds of loads. RQUS performs a little better than FIFO-QH with the QoS metric and a little worse than FIFO-UH with the QoD metric. RQUS performs much better than FIFO that doesn't favor any kind of tasks.

6 Conclusions

In this paper we propose a requirement-based query and update scheduling algorithm RQUS which allows users to specify the real needs about the query tasks by assign acceptable response time delay and acceptable result staleness. RQUS uses a global table to store the freshness of each partition and modifies it every time. An update task is executed completely in order to reduce the complexity of recalculating the freshness of every partition. The priorities of update tasks are calculated according to the partition unit so that update tasks in one partition are scheduled in FCFS mode. We use four metrics to compare RQUS to the three competing scheduling algorithms. The results show that RQUS outperforms others in the overall performance. Moreover, RQUS not only adjusts itself with the changing requirements of users and satisfies the requirements of users very well.

References

1. Qu, H.M., Labrinidis, A.: Preference-Aware Query and Update Scheduling in Web-databases. In: ICDE, pp. 356–365 (April 2007), doi:10.1109/ICDE.2007.367881
2. Thiele, M., Fischer, U., Lehner, W.: Partition-based Workload Scheduling in Living Data Warehouse Environments. In: DOLAP, pp. 57–64 (November 2007), doi: 10.1145/1317331.1317342
3. Mehta, A., Gupta, C., Wang, S., Dayal, U.: rFEED: A Mixed Workload Scheduler for Enterprise Data Warehouses. In: ICDE, pp. 1455–1458 (April 2009), doi:10.1109/ICDE.2009.66
4. Nama, B., Shinb, M., Andrader, H., Sussman, A.: Multiple query scheduling for distributed semantic caches. *Journal of Parallel and Distributed Computing* 70(5), 598–611 (2010)
5. Qu, H.M., Labrinidis, A., Mosse, D.: UNIT: User-centric Transaction Management in Web-Database Systems. In: Proc. ICDE Data Engineering (ICDE 2006), pp. 1–10 (April 2006), doi:10.1109/ICDE.2006.166
6. Shi, J.G., Bao, Y.B., Leng, F.L., Yu, G.: Priority-based Balance Scheduling in Real-Time Data Warehouse. In: HIS, vol. 3, pp. 301–306 (2009), doi:10.1109/HIS.2009.275
7. Golab, L., Johnson, T., Shkapenyuk, V.: Scheduling Updates in a Real-Time Stream Warehouse. In: ICDE, pp. 1207–1210 (April 2009), doi:10.1109/ICDE.2009.202
8. Labrinidis, A., Roussopoulos, N.: Update Propagation Strategies for Improving the Quality of Data on the Web. In: VLDB, pp. 391–400 (June 2001)
9. Burns, A.: Scheduling hard real-time systems: a review. *Software Engineering Journal* 6, 116–128 (1991)
10. Othayoth, R., Poess, M.: The Making of TPC-DS. In: Proc. VLDB, pp. 1049–1058. ACM Press, New York (September 2006)

Answering Subgraph Queries over Large Graphs

Weiguo Zheng, Lei Zou *, and Dongyan Zhao

Peking University, Beijing, China

{zhengweiguo, zoulei, zdy}@icst.pku.edu.cn

Abstract. Recently, subgraph queries over large graph-structured data have attracted lots of attentions. Most of the recent algorithms proposed to solve this problem apply the structural features of graphs to construct the index, such as path, tree and subgraph. However, there is no a solid theory foundation of which structure is the best one to construct the index. What is more, the cost of mining these structures is rather expensive. In this paper, we present a high performance graph query algorithm, SMS, based on the simple yet effective neighborhood structure. To further improve the query performance, a graph partition solution is proposed and the efficient codes of vertices and blocks are carefully designed. Extensive experimental studies demonstrate the effectiveness and scalability of our algorithm in the issue of subgraph queries on large graph-structured data.

1 Introduction

In the era of Web data explosion, it is very important to organize and manage massive data efficiently. Well-established relational databases work well in the areas they are designed for, such as Enterprise Resource Planning (ERP) and Management information system (MIS), but they have a serious shortcoming, that is “schema before data”. Furthermore, it is quite expensive for RDBMS-based systems to handle updates over schemas. However, in Web data management, such as social network analysis, the schema is always changing. In this case, graph is a good model to solve this problem, because graph is powerful in expression with its structural features. Furthermore, also due to its flexibility, graph provides a good solution for Web data mashup. Thus, graph data management has attracted lots of attentions in different areas, such as bioinformatics, social networks, and semantic data management. It is a key problem to develop efficient and effective techniques to manage, process, and analyze graph data. Among these techniques, subgraph query is an interesting, fundamental, and important task.

Recently, many techniques processing subgraph query over large graph databases have been proposed [16, 17, 22]. There are two scenarios of subgraph queries, one of which is searching a query graph over a large number of small graphs, and the other one is searching a query graph over a large graph. Note that, this work focuses on the latter scenario.

Many subgraph isomorphism algorithms have been proposed, such as Ullmann [15] and VF2 algorithm [3]. However, these algorithms cannot work well in large graph databases. As we know, subgraph isomorphism is a classical NP-complete problem. In

* Corresponding author.

order to improve the query performance, most of existing works adopt the “filter-and-refine” framework. Specifically, some indexes are built in offline processing. At run time, based on these indexes, an efficient filtering algorithm is used to reduce the search space. Finally, subgraph isomorphism algorithm is used to find final answers.

Most index-based solutions propose to utilize some frequent structural features, such as paths, trees or subgraphs, to construct the indexes [14,16,17,2,6,5,21]. However, this kind of methods lose their advantages in large graphs, since frequent structural pattern mining over a large graph is still an open question in data mining community. In order to address this issue, some neighborhood-based solutions have been proposed. For example, Zhao and Han propose SPath algorithm [20], which utilizes shortest paths around the vertex as basic index units. A key problem of SPath lies in the inefficiency of offline processing. Furthermore, it is quite expensive to perform updates over these indexes.

In this paper, for each vertex, we propose a simple yet effective strategy to encode the neighborhood structure around the vertex. Specifically, we consider the labels and the degrees of the neighbors. Based on the codes, we combine the filtering and verification together to speed up query processing. Furthermore, in order to address the scalability of our method, we propose a graph partition solution. We partition a large graph G into n blocks G_i , $i = 1, \dots, n$. For each block G_i , we also design a code, based on which, many blocks can be filtered out safely. In order to find matches across multi-blocks, we build some carefully-designed indexes.

To summarize, in this work, we make the following contributions:

1. Based on the neighborhood of one vertex, we propose a simple yet effective code for each vertex.
2. We propose an efficient subgraph search algorithm, which combines the filtering and refining process together.
3. In order to answer a subgraph query over a very large graph, we propose a graph partition-based solution.
4. Extensive experiments confirm the superiority over existing algorithms.

The rest of the paper is organized as follows. Section 2 introduces the related work. Section 3 defines the problem definition. Section 4 introduces the subgraph query algorithm SMS. The improved algorithm SMSP based on graph partition is proposed in Section 5. Section 6 reports the experimental results on real and synthetic datasets. Finally Section 7 gives the conclusion.

2 Related Work

There are mainly two embranchments of subgraph query. One of them is exact subgraph query, which means all the vertices and edges are matched exactly. The other one is approximate subgraph query, which usually concerns the structure information and allows some of the vertices or edges not be matched exactly [19,4,8,11]. However, we focus on the exact subgraph query in this paper.

In the exact subgraph query, there are two categories of related techniques: non-feature-based index and feature-based index. The algorithm Ullmann [15] to solve the problem of subgraph isomorphism is in the first branch. However, it is very expensive

when the graph is large. Some other algorithms were proposed [3,10], of which VF2 [3] is relatively efficient. The first step in VF2 algorithm is to compute the candidate set $P(s)$ of vertices. In this step, only the edge information is considered, thus the size of $P(s)$ is too large. In the verification phase, the feasibility rules are not efficient enough to verify the candidates either. The authors of Closure-tree [7] propose pseudo subgraph isomorphism using the strategy of checking the existence of semi-perfect matching between the query graph and database graphs. Noticed that the task of finding semi-perfect matching is very cost. So this algorithm will not work effectively for a large graph.

Recently, many index-based subgraph query algorithms have been proposed. Most of these algorithms utilize the paths, trees or subgraphs to construct the indexes [18,13,22,20]. Haichuan Shang develops an algorithm QuickSI [13] to test subgraph isomorphism. In the filtering phase, the features of prefix tree are considered to accommodate this algorithm. Shijie Zhang proposes the algorithm GADDI [18] based on the idea of frequent substructures. The authors of NOVA [22] construct the indexes based on the neighborhood label distribution of vertices. Peixiang Zhao investigates the SPath [20] algorithm, which utilizes shortest paths around the vertex as basic index units. Due to the complicated indexes, the index building cost of GADDI, Nova and SPath are all very expensive.

3 Problem Definition

In this section, we formally define our problem in this paper.

Definition 1. A labeled graph G is defined as $G = \{V, E, \sum_V, \sum_E, F_G\}$, where V is the set of vertices, E is the set of edges, \sum_V is the set of vertex labels, \sum_E is the set of edge label, and F_G is the mapping function that maps the vertices and edges to their labels respectively.

Definition 2. A graph $G = \{V, E, \sum_V, \sum_E, F_G\}$ is isomorphic to another graph $G' = \{V', E', \sum_{V'}, \sum_{E'}, F_{G'}\}$, denoted by $G \approx G'$, if and only if there exists a bijection function $g: V(G) \rightarrow V'(G')$ s.t.

$$1) \forall v \in V(G), F_G(v) = F_{G'}(g(v)); 2) \forall v_1, v_2 \in V(G), \overrightarrow{v_1 v_2} \in E \Leftrightarrow \overrightarrow{g(v_1)g(v_2)} \in E'$$

Given two graphs Q and G , Q is subgraph isomorphic to G , denoted as $Q \subseteq G$, if Q is isomorphic to at least one subgraph G' of G , and G' is a match of Q in G .

Definition 3. (Problem Statement) Given a large data graph G and a query graph Q , where $|V(Q)| \ll |V(G)|$, the problem that we conduct in this paper is defined as to find all matches of Q in G , where matches are defined in Definition 2

For example, in Figure 1, Q is a query graph, and G is a database graph. One match of Q in G is denoted as dotted lines in Figure 1. In this paper, we develop an algorithm to find all the subgraph matches of Q in G . For the purposes of presentation, we only discuss our method in undirected vertex-labeled graphs. Note that, it is very easy to extend our methods to directed and weighted labeled graph without a loss of generality.

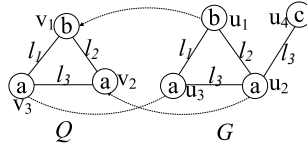


Fig. 1. An example of matching between Q and G

4 Subgraph Search Algorithm

In this section, based on the neighborhood structure, we first introduce a simple yet effective *vertex code* for each vertex. Based on the codes, we then propose a novel subgraph isomorphism algorithm.

4.1 Vertex Codes

As we know, subgraph isomorphism is a NP-complete problem. The key problem is the large search space. Let us recall Ullmann algorithm. Initially, a vertex v in query Q can match to any vertex u with the same label in graph G . Based on the neighborhood’s structure around the vertex, some vertices u can be pruned, even though they have the same labels as v in Q . Although SPath also proposes to use shortest paths around one vertex as basic index units [20], it is very expensive to build the index. Furthermore, the index has to be re-built, if any update happens to graph G . Thus, in this work, we propose to use some simple yet effective structures to encode each vertex.

Definition 4. Vertex Code. Given a vertex u in graph G , its vertex code is defined as $C(u) = [L(u), NLS(u) = \{\{l_i, (d_{i1}, \dots, d_{im})\}\}]$, where $L(u)$ is the label of vertex u , l_i is a kind of label of neighbor vertices of u , d_{i1}, \dots, d_{im} is the degree list of u' neighbor vertices with label l_i and $d_{i1} \geq d_{i2} \geq \dots \geq d_{im}$.

Definition 5. PreSequence. Given two sequences of numbers in decreasing order, $S = \{s_1, s_2, s_3, \dots, s_m\}$ and $T = \{t_1, t_2, t_3, \dots, t_n\}$, where s_i ($1 \leq i \leq m$) and t_j ($1 \leq j \leq n$) are both integers. S is called a PreSequence of T if and only if 1) $n \leq m$; and 2) $\forall s_i \in S, |\{t_j | t_j \geq s_i\}| \geq i$.

Definition 6. Subnode. Given two vertices v and u in query Q and graph G , respectively. v is a subnode of u if and only if 1) $L(v) = L(u)$; and 2) $\forall l_i \in NLS(v), \exists l_j \in NLS(u)$ and $l_i = l_j$ and the degree list $\{d_{i1}, \dots, d_{im}\}$ is PreSequence of the degree list $\{d_{j1}, \dots, d_{jn}\}$.

If v is a subnode of u , we can also say u is supernode of v , which is denoted as $v \sqsubseteq u$.

Based on *Vertex Code*, the indexes of graph G (denoted as LVG) is constructed as follows: the id and vertex code of each vertex.

For example, in Figure 2 the index structures for G are: $LVG = \{[1, a, [a, (3); b, (3); c, (4, 2)]]; [2, c, [a, (4); c, (4)]]; [3, c, [a, (4, 3); b, (3); (c, 2)]]; [4, a, [a, (4); b, (2); c, (4)]]; [5, b, [a, (4, 3)]]; [6, a, [a, (4, 3); b, (3, 2)]]; [7, b, [a, (4, 4); c, (4)]]\}$.

Similarly, we can also encode vertices by vertex codes into LVQ . For example, given a query Q in Figure 2 $LVQ = \{[1, c, [a, (3); b, (2); c, (2)]]; [2, c, [a, (3); c, (3)]]\}$.

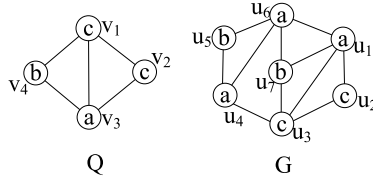


Fig. 2. A query graph Q and a database graph G

$[3, a, [b, (2); c, (3, 2)]]; [4, b, [a, (3); c, (3)]]$. Based on the indexes, we can easy find that u_1 is the supernode of v_3 .

Lemma 1. *Given two graphs Q and G , if there is a subgraph G' (in G) that matches to Q , $\forall v \in V(Q), \exists u \in V(G), v \sqsubseteq u$.*

Proof. Based on the problem definition, it is obviously proved.

With the vertex codes, it is easy to test whether a vertex v is a Subnode of another vertex u . For example, u_1 is the supernode of v_3 but u_4 and u_6 not, so the candidate vertex of v_3 in G is only u_1 .

4.2 Framework of the Algorithm

With the proper filtering approach introduced in the former subsection, the framework of our algorithm is presented as follows: first, encode the database graph G and query graph Q based on [4]. Then traverse each vertex v of Q in breadth first search and filter the vertices of G to get the candidate vertex set of v . During the process of traversing vertices in Q , the operation of verification in the depth first search is conducted. That is also to say, the filter-and-refine is combined together, which will be presented in details in the following subsection.

4.3 Subgraph Query Algorithm

Definition 7. Match Sequence. *Given two graphs Q and G , match sequence between Q and G is a sequence of matched vertex pairs $MS(n) = \{(v_{i_1}, u_{j_1}), \dots, (v_{i_n}, u_{j_n})\}$, where $u_{j_k} (1 \leq k \leq n)$ is the vertex in G that matches with the vertex v_{i_k} in Q . If the size of MS is equal to the size of Q , Q is a subgraph of G .*

In the query Algorithm [1], the vertex pairs of Match Sequence are stored in two vectors, which are denoted as SMQ and SMG respectively. The first step in query process is to select a vertex as shown in Algorithm [1]. If we select a vertex whose supernode set is least in size the outer loop decreases, especially when the size of the other candidate sets is larger. Similarly, to put the neighbors of current vertex into the match vector SMQ , the vertex v whose degree is least should be chosen first, and then select the vertex whose degree is least in the rest neighbor vertices. At last, all the vertices will be selected.

Algorithm 1. Subgraph Match preSequence (SMS) Algorithm

Require: **Input:** LVG, LVQ, G and Q .

Output: Matches of Q over G

```

1: Select the id of 1st vertex in LVQ into SMQ, push its neighbors' ids into the vector.
2: for each  $v \in SMQ$  do
3:   compute the candidate vertices  $C(v)$  for  $v$ .
4:   for each  $u \in C(v)$  do
5:     quickly verify the current match.
6:     if  $u$  matches  $v$  then
7:       push  $u$  into SMG.
8:       for each  $v' \in$  neighbors of  $v$  do
9:         if  $v'$  does not exist in SMQ then
10:          put  $v'$  into SMQ.
11:       SMS(LVG, LVQ, G, Q).
```

Definition 8. Prior Vertex. *In the matching sequence, the prior vertex of v is the earliest selected vertex in the neighbor vertices of v .*

In the matching sequence, each vertex except the first one has a prior vertex. In the matching process, to get all the candidate vertices of v , first we find all the supernodes, which are denoted as set S_1 . Then find the vertex matched with the prior vertex of v , and get its neighbor vertices denoted as set S_2 . Last the candidate vertices of v is the intersection of S_1 and S_2 .

Noticed that no matter how effective the filtering ability is, it is unavoidable to conduct the verification operation. So designing an effective and efficient verification approach is rather important and necessary.

If Q is subgraph isomorphic to G , for each subgraph of Q must be subgraph isomorphic to the corresponding subgraph of G . This is similar to the Apriori property in the frequent pattern mining [11]. That is also to say, If Q is subgraph isomorphic to G , there must be a match sequence at least. For each subset vertices of the match sequence, the vertex in Q must be subnode of the corresponding vertex in G . This has been proved in Lemma 1. However, there is another important theorem as presented in Lemma 2.

Definition 9. Sequence Number Set. *Given two graphs Q and G , as to the match sequence $MS(n)$, Sequence Number Set of the next vertex v to be matched is a set of number, denoted as $SNS(v)$, and its size is equal to the degree of v . For each neighbor v' of the vertex v , if v' is in the $MS(n)$, the sequence number of v' in the $MS(n)$ is pushed into $SNS(v)$, or the value $(n+j+1)$ is pushed into $SNS(v)$, where j is the number of neighbors of v that are not in $MS(n)$ currently.*

Lemma 2. *In the process of finding the match sequences between Q and G , the pair of new vertices to be matched are v and v' in Q and G respectively. If $SNS(v) \not\subseteq SNS(v')$, v' does not match v in the sequence, or they will match.*

Proof. If current match sequence $MS(k)=\{(v_t, u_s), (v_{t+1}, u_{s+1}) \dots (v_{t+k}, u_{s+k})\}$. When trying to find next match pair, the next candidate vertex in Q is v_{k+1} , and the next candidate vertex in G is u_{k+1} . If $SNS(v_{k+1}) \not\subseteq SNS(u_{k+1})$, there must be

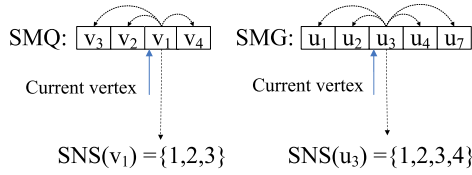


Fig. 3. SNS of current vertices in Q and G

some numbers which are not in $SNS(u_{k+1})$. Supposing the corresponding vertices are $\{v_{m1}, v_{m2} \dots v_{mp}\}$, then the edges $(v_{m1}, v_{k+1}), (v_{m2}, v_{k+1}) \dots (v_{mp}, v_{k+1})$ cannot be matched in G , so v_{k+1} is not matched with u_{k+1} .

Lemma 3. For each pair of vertices in each subsequence of a sequence S of vertices between Q and G , if the vertex in Q is a subnode of the vertex in G , the sequence S is a match sequence of Q and G .

Proof. If each subsequence of S could succeed, we can assign a sequence from 1 to the length of S , and this sequence assigned must succeed. Then Lemma3 is deduced. Since Lemma2 could be proved, so does Lemma 3.

For example, in Figure 3 supposing that two vertices have been matched currently, $MS(2)=\{(v_3, u_1), (v_2, u_2)\}$, if the next vertex to be matched in Q is v_1 , a candidate vertex of v_1 in G is u_3 . As is shown in Figure 3, $SNS(v_1)=\{1,2,3\}$, $SNS(u_3)=\{1,2,3,4\}$, $SNS(v_1)$ is a subset of $SNS(u_3)$, so v_1 can match u_3 temporarily.

If all the vertices in Q are included in the MS, the match sequence is a solution. In the match process, if the degree of a vertex is larger, the more edges will be test at a time. That is also to say graph-at-a-time is more cost-effective than traditional edge-at-a-time query processing. So this verification is rather efficient and effective for the good properties.

5 Subgraph Query Based on Partition

Considering the problem of subgraph isomorphism is NP-Complete, isomorphism test will be costly and inefficient in time and space cost if the size of database graph is very large. However, if the size of the graph is small or the graph is sparse, isomorphism test will be much easier.

Intuitively, we can partition the database graph into some small subgraphs. However, there are two major challenges: what the number of the partitioned subgraphs is better and dealing with matches crossing in multi-blocks.

5.1 Offline Processing

First, we partition the database graph into N blocks with METIS[9]. Noticed that if the size of each block is too small, the number of subgraphs and crossing edges are both increasing. Based on our experiments, when the size of the subgraph is 10 to 100 times

as large as the query graph, the query performance is better. In order to improve the query performance, we propose the block codes for each block to filter out some blocks that do not contain any subgraph that matches the query graph Q .

Definition 10. Block Codes. Given a block B of G , its block code is defined as $LLB = \{l_1, m_1, (d_1, d_2, \dots, d_{m_1}); \dots; l_n, m_n, (d_1, d_2, \dots, d_{m_n})\}$, where l is the label, m is the number of vertices with label l and d_i is the degree of i th node which has the label l .

The same code LLQ is also used for Q .

Lemma 4. Given two graphs Q and G , if Q is a subgraph of G if and only if $\forall l \in LLQ, \exists l \in LLG$ and the corresponding degree list LLQ must be the PreSequence of that in LLG , where PreSequence is defined in Definition 5.

Proof. It is easy to be proved according to the definition of the subgraph isomorphism.

In figure 4, the database graph is partitioned into three blocks (P_1, P_2, P_3) . $LLQ = \{a, 1, (2); b, 1, (2); c, 2, (3, 3)\}$, $LLP_1 = \{a, 1, (3); b, 1, (3); c, 2, (4, 3); d, 1, (3)\}$, $LLP_2 = \{a, 2, (3, 2); b, 1, (3); c, 1, (4); e, 1, (4)\}$, $LLP_3 = \{a, 1, (4); b, 1, (2); c, 2, (3, 2); e, 1, (3)\}$. To find the matches without this strategy, we have to try to traverse every partitioned subgraph. For example, when traversing P_1 , one candidate match sequence is “ $u_4-u_3-u_2$ ”, in that case the algorithm does not return until after trying to match the last vertex with label “ c ” in Q . However, if employing Lemma 4, we can filter P_2 and P_3 safely. So before dropping into recursion deeply, we can conclude that Q cannot match any subgraph in P_2 and P_3 . Undoubtedly, this will make our algorithm more efficient and effective.

There is another big challenge that dealing with the possible matches crossing several blocks. It is easy to get the idea that extending every block over x hops so that every two blocks have a overlap. However, the real database graph is dense, so the extended blocks will be as large as the database graph due to the “Six Degrees of Separation” [12].

To reduce the crossing edges, the algorithm of “Min-Cut” is employed as mentioned before. Noticed that after partition, if we only consider the crossing edges, the graph will be sparse as shown in Figure 7. Thus each partition edge with two labels can be viewed as a special “vertex”. The inverse index “LVE” similar to the previous LVG(or LVQ) can be constructed.

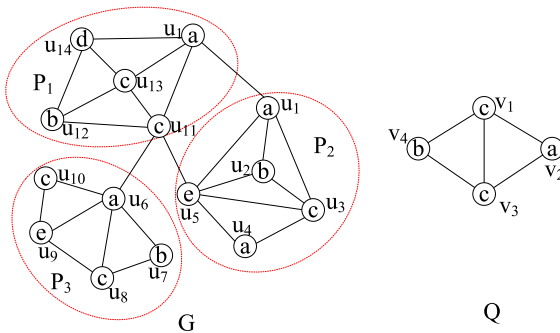


Fig. 4. The partitioned graph G and the query graph Q

Definition 11. Given a crossing edge “ u_1u_2 ”, its crossing edge code is defined as $LVE=[(l_1, l_2)\{u_1, u_2, NLS(u_1), NLS(u_2)\}]$, where l_1 and l_2 are the corresponding labels of u_1 and u_2 , $NLS(u_1)$ and $NLS(u_2)$ are the neighbor degree lists of u_1 and u_2 respectively, which are the same as that in definition 4.

For example, the “LVE” of crossing edge “ u_6u_1 ” is $[(a, c)\{u_6, u_1, [b(2), c(2, 2), e(3)], [a(4), b(4), d(3), e(5)]\}]$. It is obvious that the prune ability of LVE will be much more evident, because it stores more information. What is more, if a database graph contain 100,000 vertices, 500,000 edges and 200 different labels, the size of candidate set of each kind vertex label is 500 on average. But if we combine two vertices as a special “vertex” and the the number of crossing edges is 300,000, the size of candidate set of each kind “vertex” label is 7.5 on average. Undoubtedly, the search space be reduced greatly.

5.2 Online Query Based On Partition

With the indexes constructed in former subsection, we conduct the match process in this subsection. The framework of improved algorithm with partition is shown in Algorithm 2.

First, we find the matches in each block, the process of which is the same as that in SMS. What need to be noted is that the size of the “G” is much smaller relatively. So the time consumed in this phase is far less than that in the whole database graph.

Algorithm 2. Subgraph Match preSequence Based on Partition (SMSP) Algorithm

- 1: For each block calls Algorithm 1
 - 2: **for** each edge $e \in Q$ **do**
 - 3: find the crossing edges that match e in G .
 - 4: put the two vertices of e into SMQ.
 - 5: call Algorithm 1
-

Second, we find the possible matches through the crossing edges. In order to avoid traversing a possible match many times, a lexicographic order should be assigned. The match process is similar to SMS. The main difference lies in the match sequence of Q . Here, the match spreads in two directions in the first step. For each edge e in Q , we select a edge e' from crossing edges which matches e and push their vertices into SMQ and SMG respectively. And then the neighbors of vertices of e are pushed into SMQ. The rest process is the same as SMS.

6 Experimental Evaluation

In this section, we evaluate the query performance of our algorithms SMS and SMSP over both synthetic and real data sets, which prove that our approaches outperform some state-of-the-art algorithms well by more than one order of magnitude.

6.1 Experiment Preparation

The codes of GADDI[18] and Nova[22] are provided by authors. Furthermore, we implemented another algorithm SPath according to [20]. In our algorithm, we use the software metis to partition the database graph. All these algorithms compared were implemented using standard C++. The experiments are conducted on a P4 2.0GHz machine with 2Gbytes RAM running Linux.

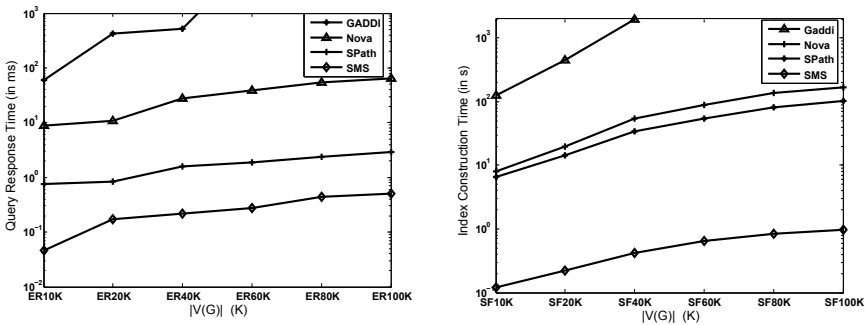
Synthetic Data Sets. We construct two data sets using two classical random graph models, Erdos Renyi model and Scale-Free model. The two data sets are denoted as ER and SF respectively. The number of vertices in ER and SF both vary from 10K to 100K. And the number of vertex labels of all the database graphs is 250.

Real Data Sets. We use a human protein interaction network(HPRD) and a RDF data(Yago) as the real data sets. HPRD consists of 9, 460 vertices,37, 000 edges and 307 generated vertex labels with the GO term description.Actually, Yago is a RDF graph consists of 368, 587 vertices, 543, 815 edges and 45, 450 vertex labels, where vertices, edges and labels corresponds to subjects(or objects), properties and the classes of subjects(or objects) respectively. The edge labels and direction are ignored in our experiments.

6.2 Experimental Results

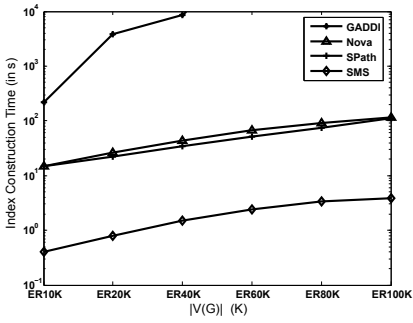
In this sub-section, we compare our algorithm SMS with GADDI [18], Nova [22] and SPath [20]. And then study the query performance of SMSP.

Experiment 1. Performance of SMS versus $|V(G)|$. Figure 5(a) and Figure 5(b) summarize the time consumed by the four methods,where the corresponding query graph is 10-vertex graph and size of the data graph of ER and SF varies from 10k to 100k. Noted that the query respond time of SMS is less than 1 second even though the size of G is 100k. What is more,the index construction time of our algorithm is also less than the other three algorithms as shown in Figure 6(a) and Figure 6(b). Undoubtedly,it proves our algorithm has a good scalability as $|V(G)|$ increasing.

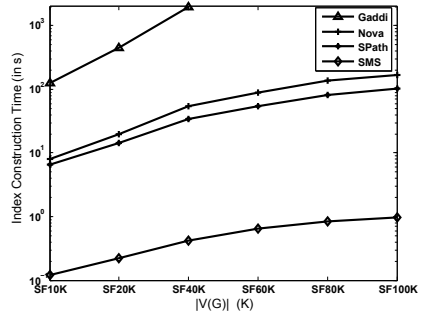


(a) Query Response Time (in milliseconds) over ER (b) Query Response Time (in milliseconds) over SF Graphs

Fig. 5. Performance VS. $|V(G)|$ in Query Response

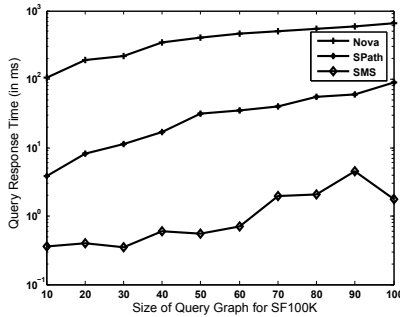


(a) Index Building Time (in seconds) over ER Graphs

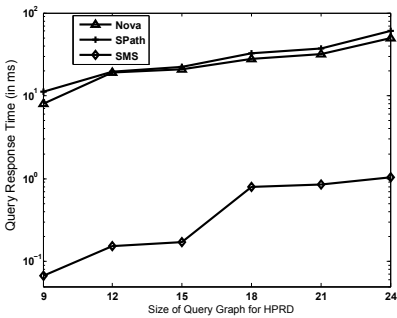


(b) Index Building Time (in seconds) over SF Graphs

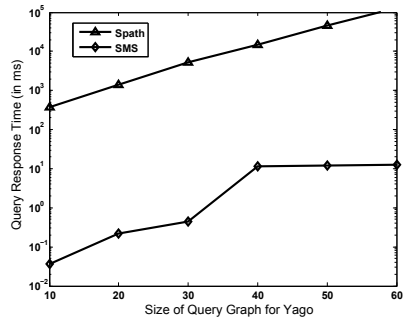
Fig. 6. Performance VS. $|V(G)|$ in Index Building



(a) Query Response Time (in milliseconds) Over SF100K



(b) Query Response Time (in milliseconds) Over HPRD



(c) Query Response Time (in milliseconds) Over Yago

Fig. 7. Performance VS. $|V(Q)|$ Over SF100K, HPRD, Yago

Experiment 2. Performance of SMS versus $|V(Q)|$. To further study the scalability and efficiency of our methods, we fix the database graph to be SF100k, HPRD, Yago respectively. And then, vary the the size of query graph. The corresponding results are shown in Figure 7(a), Figure 7(b) and Figure 7(c).

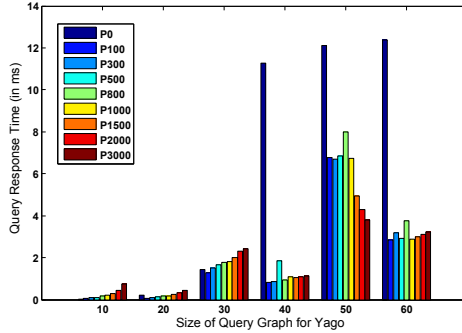


Fig. 8. Performance SMSP VS. SMS

In HPRD data set, SMS takes 0.25 seconds to construct the index. To finish the index construction Nova and GADDI need about 20 seconds and 600 seconds respectively. In Yago data set, SMS takes about 4.22 seconds to finish the index process, which is also far faster than SPath.

Experiment 3. Performance of SMSP versus partition number. In this experiment, we use the data set of Yago and vary the number of partition blocks from 100 to 3000. The query response time is shown in Figure 8, where P_n means partition the database graph into n parts. Specially, P_0 means no partition, namely SMS.

Noted that, when the query graph is small (such as the size of Q is 10, 20 or 30), the SMS is faster than SMSP no matter how many parts the database graph partitioned into. While when the size of the query graph is larger relatively, the superiority of SMSP appears, but it is very difficult to determine how many blocks the database graph partitioned into is best.

To sum up, it is obvious that the algorithm shows high efficiency and good scalability in all these data sets. Generally speaking, our algorithm is orders of magnitude faster than the compared existing three algorithms. When the query graph is larger, SMSP outperforms SMS well, but the optimal number of blocks needs to be studied in further step.

7 Conclusions

In this paper, we consider the query graph problem on large graph-structured data. Most of the recent algorithms construct the index based on the structure information of the graph, which are expensive due to the mining of the structure feature. We carefully design vertex code based on the information of each vertex and its neighbors. What is more, we propose the strategy of partitioning the large graph to improve the query performance. No matter what the index is constructed based on and how effective the pruning method is, the efficient verification strategy is important and critical. The algorithm presented in this paper is very effective and efficient both in time and memory for its simple index.

Acknowledgments. This work was supported by NSFC under Grant No. 61003009 and RFDP under Grant No. 20100001120029.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: VLDB, pp. 487–499 (1994)
2. Cheng, J., Ke, Y., Ng, W., Lu, A.: *fg*-index: Towards verification-free query processing on graph databases. In: SIGMOD (2007)
3. Cordella, L.P., Foggia, P., Sansone, C., Vento, M.: A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* 26(10), 1367–1372 (2004)
4. Dost, B., Shlomi, T., Gupta, N., Ruppin, E., Bafna, V., Sharan, R.: Qnet: A tool for querying protein interaction networks. In: Speed, T., Huang, H. (eds.) RECOMB 2007. LNCS (LNBI), vol. 4453, pp. 1–15. Springer, Heidelberg (2007)
5. Williams, J.H.D.W., Wang, W.: Graph database indexing using structured graph decomposition. In: ICDE (2007)
6. Jiang, P.Y.H., Wang, H., Zhou, S.: Gstring: A novel approach for efficient search in graph databases. In: ICDE (2007)
7. He, H., Singh, A.K.: Closure-tree: An index structure for graph queries. In: ICDE (2006)
8. Jiang, H., Wang, H., Yu, P.S., Zhou, S.: Gstring: A novel approach for efficient search in graph databases. In: ICDE (2007)
9. Karypis, G., Kumar, V.: Analysis of multilevel graph partitioning. In: SC (1995)
10. Liu, J., Lee, Y.T.: A graph-based method for face identification from a single 2d line drawing. *IEEE Trans. Pattern Anal. Mach. Intell.* 23(10) (2001)
11. Mandreoli, F., Martoglia, R., Villani, G., Penzo, W.: Flexible query answering on graph-modeled data. In: EDBT, pp. 216–227 (2009)
12. Milgram, S.: The small-world problem. In: PT, vol. 1, pp. 61–67 (1967)
13. Shang, H., Zhang, Y., Lin, X., Yu, J.X.: Taming verification hardness: an efficient algorithm for testing subgraph isomorphism. *PVLDB* 1(1) (2008)
14. Shasha, D., Wang, J.T.-L., Giugno, R.: Algorithmics and applications of tree and graph searching. In: PODS (2002)
15. Ullmann, J.R.: An algorithm for subgraph isomorphism. *J. ACM* 23(1) (1976)
16. Yan, X., Yu, P.S., Han, J.: Graph indexing: A frequent structure-based approach. In: SIGMOD (2004)
17. Zhang, S., Hu, M., Yang, J.: Treepi: A novel graph indexing method. In: ICDE (2007)
18. Zhang, S., Li, S., Yang, J.: Gaddi: distance index based subgraph matching in biological networks. In: EDBT, pp. 192–203 (2009)
19. Zhang, S., Yang, J., Jin, W.: Sapper: Subgraph indexing and approximate matching in large graphs. *PVLDB* 3(1), 1185–1194 (2010)
20. Zhao, P., Han, J.: On graph query optimization in large networks. In: VLDB (2010)
21. Zhao, P., Yu, J.X., Yu, P.S.: Graph indexing: Tree + delta \geq graph. In: VLDB (2007)
22. Zhu, K., Zhang, Y., Lin, X., Zhu, G., Wang, W.: Nova: A novel and efficient framework for finding subgraph isomorphism mappings in large graphs. In: Kitagawa, H., Ishikawa, Y., Li, Q., Watanabe, C. (eds.) DASFAA 2010. LNCS, vol. 5981, pp. 140–154. Springer, Heidelberg (2010)

Finding Relevant Papers Based on Citation Relations

Yicong Liang¹, Qing Li¹, and Tiejun Qian^{2,3}

¹ Department of Computer Science,
City University of Hong Kong, Hong Kong, China

² State Key Laboratory of Software Engineering,
Wuhan University, Wuhan, China

³ State Key Laboratory for Novel Software Technology,
Nanjing University, Nanjing, China

yicoliang2@student.cityu.edu.hk, itqli@cityu.edu.hk
qty@whu.edu.cn

Abstract. With the tremendous amount of research publications, recommending relevant papers to researchers to fulfill their information need becomes a significant problem. The major challenge to be tackled by our work is that given a target paper, how to effectively recommend a set of relevant papers from an existing citation network. In this paper, we propose a novel method to address the problem by incorporating various citation relations for a proper set of papers, which are more relevant but with a very limited size. The proposed method has two unique properties. Firstly, a metric called *Local Relation Strength* is defined to measure the dependency between cited and citing papers. Secondly, a model called *Global Relation Strength* is proposed to capture the relevance between two papers in the whole citation graph. We evaluate our proposed model on a real-world publication dataset and conduct an extensive comparison with the state-of-the-art baseline methods. The experimental results demonstrate that our method can have a promising improvement over the state-of-the-art techniques.

Keywords: Paper Relevance, Citation Relation, Citation Network.

1 Introduction

Since the overwhelming number of publications makes researchers difficult to survey all possibly relevant papers in a field, paper recommendation systems are developed to identify relevant information or knowledge that meets the demands of individual users. For example, when beginning to work in a new research problem, researchers are usually interested in quickly understanding existing literatures related to that problem, including finding relevant papers. Paper recommendation systems aim at recommending relevant papers to researchers with respect to their individual demands. In this paper, we deal with the problem of finding relevant papers related to a given paper.

In order to make proper recommendations, one important problem is how to measure the relevance among papers. Content analysis (e.g. text similarity computation), social network (e.g. coauthorship), citation analysis (e.g. citation count, cocitation¹),

¹ Cocitation indicates that two papers are jointly cited in one or more subsequent papers.

cocoupling²), or some combinations of them have been adopted. [12] shows that simple content analysis did not perform well in paper recommendation, however, citation feature played a major role in finding relevant papers with high quality. There exist some drawbacks in finding relevant papers by using traditional techniques, e.g. cocitation and cocoupling. For example, since some recently published papers receive few citations, it is difficult to find relevant papers by using cocitation. Similarly, cocoupling is incapable of finding relevant papers published in the early stages of the development of a research topic, because these papers only share a small number of references in common with others. Liben-Nowell et al. took advantage of the global graph structure in terms of Katz graph distance [6] to measure the relevance between nodes in a graph. However, a link in a citation graph is more than just a “simple link”, and the underlying semantics among links should be taken into account in measuring the relevance among papers. In our model, semantics among links in the citation graph are captured by their specific citation relations.

Generally speaking, there are three main cases of relevant papers: (1) Some relevant papers are recent and topically related, e.g. focusing on dealing with a similar research problem. (2) Some relevant papers are seminal works or survey articles. Even though these papers share a weaker topical similarity, they provide the context and background knowledge about the field to researchers. (3) Some relevant papers are related to some basic algorithms and standard methods, which can provide fundamental theory and tools which are helpful for researchers to solve their particular research problems. In order to find relevant papers that can cover the above cases, we incorporate various citation relations in our recommendation mechanism, since the cases of relevance are similar to the motivations for citations which can be indicated by citation relations. To the best of our knowledge, this is the first attempt to tackle the problem of finding relevant papers in this manner.

Our contributions in this work include the following:

- We propose a model to capture the dependencies between cited and citing papers by incorporating the three types of citation relations.
- We develop a method by using graph distance and weight of links to measure the relevance between two papers in a citation graph.
- We conduct experiment of studies with a real-world publication dataset, the results of which show that our proposed method outperforms the state-of-the-art methods for finding relevant papers.

The rest of this paper is organized as follows: Section 2 reviews some related literatures. We formalize our model and proposed method in Section 3. In Section 4, we conduct an extensive experiment with a real-world publication dataset, and then present and analyze the experimental results conducted on both offline and expert evaluation. Finally, we conclude our work in Section 5.

² Cocoupling indicates that two papers share one or more of the same cited papers in their reference lists.

2 Related Work

2.1 Paper Relatedness in Citation Graph

Citation information has been used to compute the relatedness among academic papers. As known to all, traditional link-based methods measure the relevance by focusing on neighbors. For example, two noteworthy methods utilizing the direct cited and citing papers are co-citation [11] and bibliographic coupling (or called cocoupling) [3]. Lawrence et al. [4] proposed a similarity measure called Common Citation \times Inverse Document Frequency (CCIDF) which was based on common citations to measure the relatedness between papers. Lu et al. [7] proposed a metric called HITS Vector-based which took advantage of local neighborhoods of nodes in the citation graph to quantify similarity of papers. Liben-Nowell et al. [6] proposed to use Katz distance to measure the proximity among nodes. Lu et al. in [7] pointed out that, since CCIDF focuses on direct references and citations alone, CCIDF inherits the drawbacks from co-citation and co-coupling. A citation link is more than just a “simple link”, yet HITS Vector-based and Katz distance ignore the link semantics behind the citation link, which nevertheless can be useful to measure paper relevance.

2.2 Paper Recommendation

Since collaboration filtering (CF) is a popular technique in recommendation system, McNee et al. [8] proposed two algorithms User-Item CF and Item-Item CF by integrating CF into the domain of research papers. They adopted several ways to map the citation graph onto a collaborative filtering user-item rating matrix. In particular, there are two ways of mapping, described as follows: (1) a paper would represent a ‘user’ in the matrix and a citation would represent an ‘item’ and each paper would vote for its reference papers. (2) both the ‘users’ and ‘items’ in the rating matrix are citations and a rating between two papers would be some measure of cocitation. On one hand, User-Item CF [8] is based on the first way of mapping and makes recommendations by comparing the similarity of reference papers related the target paper and candidate paper. On the other hand, Item-Item CF [8] is based on the second way of mapping and makes recommendations by comparing the similarity of cited papers related to the target paper and candidate paper. It can be easy to find that User-Item CF depends on cocoupling metric [3] while Item-Item CF depends on cocitation metric [11]. In other words, the two CF methods proposed in [8] have the similar drawback with cocitation and cocoupling in nature.

Ekstrand et al. [1] proposed several methods for augmenting CF and content-based filtering algorithms with measures of the influence of a paper within the web of citations to recommend relevant papers to users. But the major difference between their work and ours is that, Ekstrand et al. focus on recommending a reading list to a target user who is interested in a particular topic, while our focus is put on recommending a list of papers which are relevant to a given paper. Similar to [1], Sugiyama et al. [13] also aimed at recommending papers to a user based on the user’s recent research interest. Ratprasartporn et al. [10] proposed a framework by incorporating ontology hierarchy as contexts to estimate relatedness between two papers. However, their proposed framework is highly dependent on domain-specific ontology hierarchy (i.e. Gene Ontology).

Strohman et al. [12] discovered that *Katz* graph distance measure [6] was good at finding relevant academic papers. However, *Katz* ignore the weight in the link in terms of the dependency between the cited and citing papers, yet utilizing the dependency in the citation graph can contribute to finding closely related papers.

3 Problem Formulation

3.1 Preliminary

In this section, we introduce a few important notations and concepts related to our problem of finding relevant papers. A paper p includes the time stamp of p (denoted by T_p), a set of reference papers cited by p (denoted by R_p), and another set of papers citing p (denoted by Q_p). For a reference p_i in R_p , we denote “ p cites p_i with a citation relation” (denoted by τ)” as $p \xrightarrow{\tau} p_i$. Several existing works have already discussed how to make use of the citation context [4] to automatically categorize such citation relations [9][14][15]. In this work, we adopt the technique from [9] to classify the citation relations into three major categories, named as *Based-on*, *Comparable* and *General*. For convenience, a pair of cited paper and citing paper are always denoted as p_{citee} and p_{citer} in the rest of this paper.

In the following, we shortly describe these three kinds of citation relation as follows. A *Based-on* relation is a relation when p_{citer} is based on p_{citee} to some extent. For example, the technique proposed in p_{citer} is based on the technique proposed in p_{citee} . A *Comparable* relation is a relation when p_{citee} has been compared to p_{citer} in terms of differences or resemblances. For example, p_{citer} and p_{citee} are solving a similar research problem but basically use different methods. The last type of citation relation, *General*, is neither *Based-on* nor *Comparable*. For example, p_{citer} introduces some background information about its research problem by citing p_{citee} . More detailed descriptions of these three types of citation relations can be found from [9].

3.2 Citation Link

Strohman et al. in [12] pointed out that, to find relevant papers with high quality, using the property of graph structure plays a more important role than simply incorporating text similarity. In this paper, we not only utilize graph structure but also different kinds of citation relations for the task of finding relevant papers. A citation graph is an acyclic directed graph, $G = \langle V, E \rangle$, where V is a set of papers, and E is a set of citation links(see in Definition 1).

Definition 1 (Citation Link). A citation link $\varepsilon = \langle p_{citer}, p_{citee}, \tau \rangle$, while τ is the citation relation. In this paper, the categories of citation relation τ include τ_B (Based-on), τ_C (Comparable) and τ_G (General).

³ In this paper, we assume that a pair of cited and citing papers only contains one kind of citation relation.

⁴ Citation context is a bag of words surrounding the citation position.

We consider that there exists some “dependency” between the cited and citing papers while Tang et al. [14] believe that if two papers describe a similar content, then they may have a high dependency. However, some pairs of cited and citing papers which vary a lot in content may still have high dependency. For example, some key parts of a solution proposed in the citing paper are based on the cited paper. Furthermore, different types of citation relations help to quantify the dependency between a pair of cited and citing papers [2]. Given a paper p and its reference papers R_p , it can be considered that p has some dependency on p_i ($p_i \in R_p$) in some sense. In this paper, we define a model called **Local Relation Strength (LRS)** (to be discussed in Section 3.3) to quantify the dependency between p_{citer} and p_{citee} , denoted as $LRS(p_{citer} \rightarrow p_{citee})$.

3.3 Local Relation Strength

Our proposed model LRS estimating dependency between a pair of citing and cited papers is based on the categories of citation relations and the surrounding citation environment. As mentioned in [29], some particular types of citation relations are more important than others. For example, in [9], *Comparable* relation is considered as more important than the other two types of relations. Besides following the above assumption, in this work, we further assume *Based-on* relation carries more importance than *General* relation. We use a predefined parameter w_τ to represent the importance of the three types of citation relations. The conditions upon these three parameters of citation relations are listed below:

$$\begin{cases} w_C + w_B + w_G = 1 \\ w_C > w_B > w_G \end{cases} \quad (1)$$

where w_C , w_B and w_G are corresponding to *Comparable*, *Based-on* and *General* respectively. Our experiment results suggest that treating *Comparable* and *Based-on* as more important than *General* (i.e setting more weight to w_C and w_B) is more effective and efficient to find relevant papers.

Given a citation link ε with $p_{citer} \xrightarrow{\tau} p_{citee}$, intuitively, the corresponding citation environment with respect to ε can be reflected by $R_{p_{citer}}$ and $Q_{p_{citee}}$. Furthermore, if the types of citation relation are taken into account, then the citation environment also includes the papers which cite p_{citee} with τ (denoted by $Q_{p_{citee}}^\tau$). However, the papers which are cited by p_{citer} with τ are not taken into consideration for measuring the dependency between p_{citer} and p_{citee} , because the numbers of references vary a lot in different papers. For example, due to the space constraint, the authors may cut some references, while more space becomes available, the authors may add more references. In other words, we should eliminate the unsuitable effect in terms of the insignificant addition/deletion of references on dependency measurement. p_{citee} may be cited by more than one papers with the same citation relation τ (i.e. $Q_{p_{citee}}^\tau$), but not all the citing papers in $Q_{p_{citee}}^\tau$ have the same dependency with p_{citee} . In a word, two citation links in different surrounding citation environment may carry different dependency even though they are related to the same type of citation relation.

We adopt the following criteria to determine LRS: (i) the larger ratio of papers which cite p_{citee} with τ , the higher LRS is; (ii) the closer the timestamp between p_{citer} and

p_{citee} , the higher LRS is. The assumptions behind these criteria are as follows: (i) A citation relation type can reflect a specific citing purpose of p_{citer} when citing p_{citee} (in this paper, for simplicity, we assume that each type of citation relations is related to one specific citing purpose). For a particular purpose, e.g. p_{citer} cites both p_{citee1} and p_{citee2} with the same τ , if p_{citee1} has a larger ratio of citations for that citing purpose than p_{citee2} does, then p_{citer} have a stronger dependency with p_{citee1} than p_{citee2} . The reason for using ratio instead of count is that some recent published papers may receive a small number of citations, which leads $|Q_{p_{citee}}^\tau|$ to become small as well. (ii) Authors are in general selective in citing relevant papers, e.g. to prefer citing recently published papers. We assume that a pair of citing and cited papers with a closer timestamp have a stronger dependency than those with a farther timestamp. We use $d(T_{p_{citer}}, T_{p_{citee}})$ to denote the temporal distance between two papers and adopt the time decay function $e^{-d(\cdot)}$ in the dependency computation. As mentioned the category of citation relation can affect on measuring dependency, so w_τ representing the importance of τ is adopted in the calculation of LRS. Finally, we define LRS as follows:

$$LRS(p_{citer} \rightarrow p_{citee}) = w_\tau * \frac{|Q_{p_{citee}}^\tau|}{|Q_{p_{citee}}|} * e^{-\frac{d(T_{p_{citer}}, T_{p_{citee}})}{d_M}} \quad (2)$$

where d_M denotes the largest temporal distance among all pairs of p_{citer} and p_{citee} .

3.4 Paper Relevance Measurement

To recommend related papers of p , only using direct citation relations for recommendation may miss some highly related papers. For example, some ‘‘indirectly cited papers’’ of p , such as papers not directly cited by p but cited by papers in R_p , can be suitable candidates for recommendation as well. In order to retrieve such ‘‘indirectly cited’’ papers, the Katz graph distance measure can be used to fulfill this task [12]. Given a graph, Katz [6] defines the relevance between two nodes x and y as follows:

$$relevance(x, y) = \sum_{l=1}^{\infty} \eta^l \cdot |\theta_{x,y}^{<l>}| \quad (3)$$

where η is a decay parameter between 0 and 1, $|\theta_{x,y}^{<l>}|$ is the set of all l -hops paths from x to y . Since Katz measures the relevance between nodes by considering the number of paths between x and y , and the number of hops in each path, Eq. 3 can be equivalently defined as follows [5]:

$$relevance(x, y) = \sum_{\theta_i \in \vartheta} \eta^{|\theta_i|} \quad (4)$$

where ϑ is a set of all paths between x and y . Given two nodes, there may be many paths between them in a graph. We consider that it is enough to use short paths to find relevant nodes, because long paths contribute less weights than short paths in measuring the relevance, and longer paths may even bring in noise. Therefore, we constrain the length of path to be 3. However, Katz treats every link as equally important in the graph, which means that it ignores the existing dependency in a citation link. In our work, we use LRS to represent the dependency (denoted as λ_{xy}) in a citation link between a pair of citing

and cited papers x and y . Finally, we propose a model called *Global Relation Strength (GRS)* that measures the relevance between any two papers by combining Katz graph distance and the dependency in a citation link, defined as follows:

$$GRS(p_1 \mapsto p_2) = \sum_{\theta_i \in \vartheta} [\eta^{|\theta_i|} \cdot \frac{\sum_{e_{mn} \in \theta_i} \lambda_{mn}}{|\theta_i|}] \quad (5)$$

From Eq. 4 and Eq. 5 we can see that Katz is a special case of *GRS* where the weights in all edges are treated as 1. When incorporating dependency in relevance computation, we opt to use the average weight rather than product of weight because if we use the later choice, the path length will penalize the score twice, i.e. being penalized by η and λ since both η and λ are in the range of $(0, 1)$.

3.5 Extracting Relevant Candidates

Based on our proposed *GRS* as a new metric to measure the relatedness between two papers, we now move on to the task of extracting relevant candidates for a target paper p . First, we use depth-first-search (DFS) technique to extract a set of papers (denoted as D_{prior}) whose distance is less than K -hop⁵ from the source node p . Since in a citation link, the timestamp of cited paper is earlier than timestamp of citing paper, $\forall p_i \in D_{prior}$, T_{p_i} is earlier than T_p . However, since not all the timestamps of related papers should be earlier than the timestamp of the input paper, recommending D_{prior} alone is incomplete. Hence, again, we use DFS to extract another set of papers ($D_{posterior}$) whose distance is less than K to the target node p . Finally, we combine $D = D_{prior} \cup D_{posterior}$, rank D by *GRS*, and then return top- M results as the candidates.

4 Experiment

4.1 Experimental Setup

The dataset used to evaluate our proposed model is a collection of journal, conference and workshop articles from ACL Anthology Network⁶ (AAN). The AAN dataset consists of 12409 papers and 61527 citation links where these papers are published between 1965 and 2009. References to papers not in AAN dataset were ignored in the citation graph construction. The reason for choosing AAN in our experiment is that, to the best of our knowledge, the public AAN dataset provides both the citation structure and citation context information which helps to classify the citation relations by using [9].

In order to find a set of relevant papers related to a given paper p , a scoring function is used to estimate the relevance between p and other papers. The top- M relevant papers are returned as recommendations according to the scoring function. As mentioned in [12] that citation features play a significant role in finding relevant papers, the baselines used to compare with our proposed method mainly focus on citation features. In

⁵ In our experiment, we set K to 3.

⁶ <http://clair.si.umich.edu/clair/anthology/index.cgi>

particular, in our experiment, the scoring functions related to the five different baselines include the following: (1) **Cocitation** [11]; (2) **Cocoupling** [3]; (3) **CCIDF** [4]; (4) **HITS Vector-based** [7]; (5) **Katz**[6].

We conduct our evaluations in two different frameworks (i.e. offline evaluation and expert evaluation) to evaluate the performance of our proposed method as well as the above baselines. In the offline evaluation, we evaluate the ability of our method to recommend papers known to be related to a selected paper. In the expert evaluation, we use human subjects to evaluate the effectiveness of our proposed method for retrieving relevant papers.

4.2 Offline Evaluation

Metrics

Following [8,12], we adopt a popular technique called *All-But-One* as our offline evaluation method. In particular, given an input paper p , first we extract the adjacent papers A_p (including R_p and Q_p) around p . Then we randomly choose 1/10 papers in A_p as test nodes denoted as T_p ($T_p \subset A_p$), and remove the citation links between p and each node in T_p , and use the remaining papers for training. We evaluate the performance by examining how many test nodes occur in the result list after removing the citation links. In our experiments, three evaluation metrics are used, i.e., F1-score, normalized discounted cumulative gain (NDCG) and mean reciprocal rank (MRR). In our experiment, given a retrieved result list D (with M returned papers), how to calculate the F1-score, NDCG and MRR can be referred in [5]. We select the papers published between 2003 and 2004 (about 1700 papers) as a test set and evaluate the recommendation performance based on this test set.

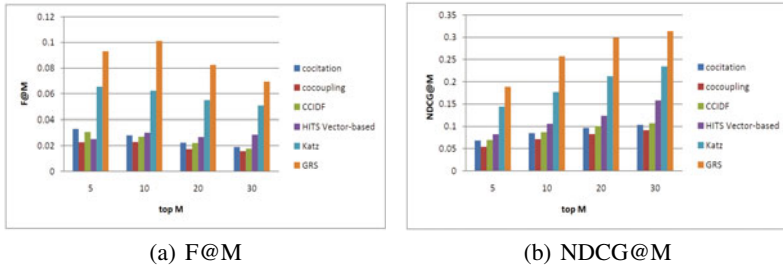
Evaluation Results

Following [6], the decay parameter η is set to 0.005. We first tune different combinations of the three citation relation weights in the following experiment. Following [9], *Comparable* is treated as the most important among the citation relations, and we further assume *Based-on* carries more importance than *General*. Due to the space limitation, we do not show the details of tuning results here and interested readers can refer to [5]. According to the experiment results in [5], we can find that when $w_C = 0.65$, $w_B = 0.2$, $w_G = 0.15$, GRS has the best performance in terms of F@10 (0.1011). In the following experiments, we set $w_C = 0.65$, $w_B = 0.2$ and $w_G = 0.15$.

The result of F@10, NDCG@10 and MRR related to the baselines as well as ours are shown in Table 1. From Table 1, we can see that HITS Vector-based, Katz and GRS which incorporate path distances can achieve better performance than other baselines that focus on neighbor structure alone. The reason is that cocitation, cocoupling and CCIDF only focus on a “narrow region” (i.e. neighbors around nodes) while HITS Vector-based, Katz and GRS look into a “broad region” (i.e. global path structure among nodes). Comparing GRS with Katz, GRS achieves 62.0% of F@10, 45.4% of NDCG@10, and of 29.6% MRR improvement. The results suggest that using link semantics in terms of citation relations and dependency contribute to finding relevant papers.

Table 1. Top-10 Results of F1, NDCG and MRR

Approach	F@10	NDCG@10	MRR
cocitation	0.02808	0.08460	0.1155
cocoupling	0.02286	0.07115	0.08039
CCIDF	0.02691	0.08684	0.09822
HITS Vector-based	0.03010	0.1057	0.09297
Katz	0.06240	0.1770	0.1485
GRS	0.1011	0.2574	0.1925

**Fig. 1.** Top- M Results

We further examine the performances of F1-score and NDCG when other results are returned. Figure 1 shows the result of $F@M$ and $NDCG@M$ ($M = 5, 10, 20, 30$) related to all the methods. From Figure 1, we can find that Katz and GRS perform better than cocitation, cocoupling, CCIDF and HITS Vector-based. In addition, comparing GRS with Katz, GRS not only achieves 42.4% of $F@5$, 62.0% of $F@10$, 50.1% of $F@20$ and 36.5% of $F@30$ improvement, but also achieves of 30.7% of $NDCG@5$, 45.4% of $NDCG@10$, 40.7% of $NDCG@20$ and 33.7% of $NDCG@30$ improvement. These results suggest that our method can significantly improve top- M (e.g. from 5 to 30) recommendation performance over the baselines.

Effect of Dependency

As introduced in Section 3.3, our proposed model LRS captures the dependency between a pair of cited and citing papers by considering three components, i.e. (i) the importance of a citation relation, (ii) the surrounding citation environment, and (iii) the temporal distance. In the following experiments, we test the performances of dependency represented by four different cases as follows: (1) LRS: this is our proposed model that all the three components are taken into account; (2) Uniform citation relation (UCR): this is similar to LRS except that the three types of citation relation are forced to carry the same weight of importance in terms of setting $w_C = w_B = w_G = 1/3$; (3) Random citation relation (RCR): this is similar to LRS except that the three types of citation relation are forced to carry a random weight of importance between 0 and 1, and in the meantime, the summation of them keeps to 1; (4) TD: temporal distance is adopted to measure dependency alone. We combine these four dependency representations with our proposed GRS model for testing.

Table 2. Four Cases of Dependency

	F@10	NDCG@10	MRR
UCR	0.09077	0.2302	0.1761
TD	0.06468	0.1807	0.1494
RCR	0.08163	0.2153	0.1731
LRS	0.1011	0.2574	0.1925

From the results in terms of F@10, NDCG@10 and MRR shown in Table 2, it can be found that LRS performs best among the four dependency models. In particular,

- LRS vs. TD: LRS improves the recommendation performance with 56.3% of F@10, 42.4% of NDCG@10 and 28.9% of MRR. These results suggest that considering the temporal distance alone as well as ignoring the underlying citation relation semantics do not success in finding relevant papers;
- LRS vs. UCR, RCR: Compared to UCR, LRS improves the performance with 11.4% of F@10, 11.8% of NDCG@10 and 9, 3% of MRR; compared to RCR, LRS obtains the improvement with 23.9% of F@10, 19.5% of NDCG@10 and 11.2% of MRR. These results suggest that further distinguishing the importance related to the three different citation relations in terms of putting more weight to *Comparable* and *Based-on* relation contributes to finding relevant papers.

4.3 Expert Evaluation

In the following experiment, we ask annotators⁷ to evaluate our proposed method. In particular, we randomly select 20 papers (a subset of test papers used in offline evaluation) whose topic is mainly about classification. For a test paper p , a set of candidate papers are returned by adopting different methods, and then annotators will make a binary decision to judge whether a candidate paper is relevant to p or not. Based on the offline experiment, $w_C = 0.65$, $w_B = 0.2$, $w_G = 0.15$ are set for GRS. The top-5 experiment results in terms of precision and NDCG are presented in Table 3(a). From the results, we can find that GRS outperforms the baselines. Specifically, GRS achieves (i) about 0.657 of precision, which suggests that our proposed method is able to accurately find relevant papers in the top-5 result list; (ii) 0.848 of NDCG, suggesting that our proposed method is capable of efficiently finding relevant papers in the top-5 result list.

Since some “indirect cited” or “indirect citing” papers can be considered as relevant papers, in the following experiments, we examine the capability of our method whether it can recommend these relevant papers. Particularly, given a test paper p , we first extract its neighbors (R_p and Q_p), then we remove those papers in R_p or Q_p from the returned results, and evaluate the recommendation performance from the remaining results. According to the results as shown in Table 3(b), we can find that our proposed GRS is able to better find more relevant papers even though they do not directly cite or are not cited by p .

⁷ Five annotators took part in the expert evaluation, and they are all Ph.D students whose research interests focus on information retrieval and data mining.

Table 3. Top-5 Results

(a) Include Neighbors			(b) Remove Neighbors		
Approach	precision	NDCG	Approach	precision	NDCG
cocitation	0.4571	0.7471	cocitation	0.4000	0.6982
cocoupling	0.4095	0.6481	cocoupling	0.3429	0.5899
CCIDF	0.5429	0.7637	CCIDF	0.4762	0.7355
HITS Vector-based	0.3429	0.6100	HITS Vector-based	0.2857	0.5200
Katz	0.6095	0.8337	Katz	0.6571	0.8415
GRS	0.6571	0.8481	GRS	0.7333	0.8951

Table 4. LRS vs. UCR, TD

(a) Include Neighbors			(b) Remove Neighbors		
	precision	NDCG		precision	NDCG
LRS	0.6571	0.8481	LRS	0.7333	0.8951
UCR	0.6371	0.8296	UCR	0.6571	0.8830
RCR	0.6	0.8269	RCR	0.5810	0.8224
TD	0.6286	0.8240	TD	0.6571	0.8529

In the following experiments, we also compare the performance related to different dependency representations, i.e. LRS, UCR and TD (mentioned in Section 4.2). We also follow two cases to evaluate the results (i.e. whether direct neighbors are included or not). From the results shown in Table 4, we can find that LRS outperforms UCR and TD in both cases, suggesting that (1) incorporating citation relation semantics contributes to improving the performance of paper recommendation; (2) further distinguish the importance of three citation relations (i.e. setting more weight to *Comparable* and *Based-on* relation) can help learners to find relevant papers efficiently.

5 Conclusions

Finding relevant papers is becoming increasingly important especially with the ever-growing number of publications every year. In this paper, we have advocated an approach to use the underlying link semantics to find relevant papers related to a given paper. To solve the problem, first, we have proposed to use three different types of citation relations to quantify the dependency between a pair of cited and citing papers. Second, we have proposed the *Global Relation Strength* model by incorporating the graph distance and the dependency of the citation links to capture the relevance between two papers in a citation graph. Experimental results conducted by both off-line and expert evaluation show that our proposed approach is more effective than the state-of-the-art methods for finding relevant papers.

Acknowledgements. The work described in this paper has been supported by a grant from the City University of Hong Kong (No. 7008043), and the Open Research Fund Program of State Key Laboratory for Novel Software Technology (KFKT2011B24).

References

1. Ekstrand, M., Kannan, P., Stemper, J., Butler, J., Konstan, J., Riedl, J.: Automatically Building Research Reading Lists. In: Proceedings of the Fourth ACM Conference on Recommender Systems, pp. 159–166. ACM, New York (2010)
2. Huang, Z., Qiu, Y.: A Multiple-perspective Approach to Constructing and Aggregating Citation Semantic Link Network. *Future Generation Computer Systems* 26(3), 400–407 (2010)
3. Kessler, M.: Bibliographic Coupling between Scientific Papers. *American Documentation* 14(1), 10–25 (1963)
4. Lawrence, S., Lee Giles, C., Bollacker, K.: Digital Libraries and Autonomous Citation Indexing. *Computer* 32(6), 67–71 (1999)
5. Liang, Y., Li, Q., Qian, T.: Finding Relevant Papers Based on Citation Relations (2011), [http://www.cs.cityu.edu.hk/~sim\\$yicoliang/paper/PaperRec.pdf](http://www.cs.cityu.edu.hk/~sim$yicoliang/paper/PaperRec.pdf)
6. Liben-Nowell, D., Kleinberg, J.: The Link-prediction Problem for Social Networks. *Journal of the American Society for Information Science and Technology* 58(7), 1019–1031 (2007)
7. Lu, W., Janssen, J., Milios, E., Japkowicz, N., Zhang, Y.: Node Similarity in the Citation Graph. *Knowledge and Information Systems* 11(1), 105–129 (2007)
8. McNee, S., Albert, I., Cosley, D., Gopalkrishnan, P., Lam, S., Rashid, A., Konstan, J., Riedl, J.: On the Recommending of Citations for Research Papers. In: Proceedings of the 2002 ACM Conference on Computer Supported Cooperative Work, pp. 116–125. ACM, New York (2002)
9. Nanba, H., Okumura, M.: Towards Multi-paper Summarization Using Reference Information. In: International Joint Conference on Artificial Intelligence, vol. 16, pp. 926–931 (1999)
10. Ratprasartporn, N., Ozsoyoglu, G.: Finding Related Papers in Literature Digital Libraries. In: Research and Advanced Technology for Digital Libraries, pp. 271–284 (2007)
11. Small, H.: Co-citation in the Scientific Literature: A New Measure of the Relationship Between Two Documents. *Journal of the American Society for Information Science* 24(4), 265–269 (1973)
12. Strohman, T., Croft, W., Jensen, D.: Recommending Citations for Academic Papers. In: Proceedings of the 30th Annual international ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 706–707 (2007)
13. Sugiyama, K., Kan, M.: Scholarly Paper Recommendation via User’s Recent Research Interests. In: Proceedings of the 10th annual Joint Conference on Digital Libraries, pp. 29–38. ACM, New York (2010)
14. Tang, J., Zhang, J., Yu, J., Yang, Z., Cai, K., Ma, R., Zhang, L., Su, Z.: Topic Distributions over Links on Web. In: Proceedings of the 2009 Ninth IEEE International Conference on Data Mining, pp. 1010–1015 (2009)
15. Teufel, S., Siddharthan, A., Tidhar, D.: Automatic Classification of Citation Function. In: Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, pp. 103–110 (2006)

ASAP: Towards Accurate, Stable and Accelerative Penetrating-Rank Estimation on Large Graphs

Xuefei Li¹, Weiren Yu^{2,3}, Bo Yang³, and Jiajin Le³

¹ Fudan University, China

xuefei.li89@gmail.com

² University of New South Wales & NICTA, Australia

weirenyu@cse.unsw.edu.au

³ Donghua University, China

{yangbo, lejiajin}@dhu.edu.cn

Abstract. Pervasive web applications increasingly require a measure of similarity among objects. Penetrating-Rank (P-Rank) has been one of the promising link-based similarity metrics as it provides a comprehensive way of jointly encoding both incoming and outgoing links into computation for emerging applications. In this paper, we investigate P-Rank efficiency problem that encompasses its accuracy, stability and computational time. (1) We provide an accuracy estimate for iteratively computing P-Rank. A symmetric problem is to find the iteration number K needed for achieving a given accuracy ϵ . (2) We also analyze the stability of P-Rank, by showing that small choices of the damping factors would make P-Rank more stable and well-conditioned. (3) For undirected graphs, we also explicitly characterize the P-Rank solution in terms of matrices. This results in a novel non-iterative algorithm, termed ASAP, for efficiently computing P-Rank, which improves the CPU time from $O(n^4)$ to $O(n^3)$. Using real and synthetic data, we empirically verify the effectiveness and efficiency of our approaches.

1 Introduction

The study of quantifying structural similarity between vertices in ubiquitous networks has attracted considerable attention over the past decade. Typical structural similarity metrics include Google PageRank, Hyperlink-Induced Topic Search (HITS), Co-citation, Bibliographic Coupling, SimRank and SimFusion (e.g., [12,34,5,6]).

P-Rank (Penetrating-Rank) is a new similarity measure of this kind, which was initially proposed by Zhao *et al.* [7]. The similarity scores flowing from in-link neighbors of entities are penetrated through their out-link neighbors. Emerging real-world applications of P-Rank include biological networks, collaborative filtering, graph classification, web document ranking, and outlier detection (e.g., [8,9,4,10,5,6]).

In contrast to other similarity measures, P-Rank has become one of the important metrics owing to the following two reasons. (i) P-Rank provides a comprehensive way to jointly explore both in- and out-link relationships with semantic completeness. In comparison, other similarity measures, say SimRank, have the “limited information problem” [5], in which only in-link relationships can be partially exploited. (ii) P-Rank transcends other similarity measures in its most general form. Other measures such as SimRank [5], Amsler [1] can be regarded as just special cases of P-Rank [7].

Unfortunately, previous work on P-Rank suffers from the following limitations. (i) P-Rank solution is known to converge [7], but a precise accuracy estimation of P-Rank is not given. (ii) No prior work has well studied *the P-Rank condition number*, which indeed has played a paramount role in measuring how much the web graph can change in proportion to small changes in the P-Rank scoring results. (iii) The P-Rank time complexity in [7] retains *quartic* in the worst case for both digraphs and undirected graphs. To the best of our knowledge, there is no efficient P-Rank algorithm specially designed for *undirected graphs*. These practical needs call for an in-depth investigation of P-Rank and its efficiency.

Contributions. This paper studies the P-Rank problems regarding its *accuracy*, *stability* and *computational efficiency*. Our main results are summarized below.

1. We provide an accuracy estimation for P-Rank iteration (Section 3). We find that the number of iterations $K = \lceil \log \epsilon / \log (\lambda \cdot C_{\text{in}} + (1 - \lambda) \cdot C_{\text{out}}) \rceil$ suffices to acquire a desired accuracy of ϵ , where λ is a weighting factor, and C_{in} and C_{out} are in- and out-link damping factors, respectively.
2. We introduce the notion of P-Rank *condition number* κ_{∞} to investigate the stability issue of P-Rank (Section 4). We show that P-Rank is *well-conditioned* for small choices of the damping factors, by providing a tight *stability bound* for κ_{∞} .
3. We propose a novel *non-iterative* $O(n^3)$ -time algorithm (**ASAP**) for efficiently computing similarities over undirected graphs (Section 5). We explicitly couch the P-Rank solution *w.r.t.* matrix products in connection with its eigen-problem.
4. We experimentally verify the efficiency of our methods on real and synthetic data (Section 6). We find that P-Rank exponentially converges *w.r.t.* the iteration number. The stability of P-Rank is determined by the damping factors and weighting factors. The **ASAP** algorithm outperforms baseline algorithms on undirected graphs.

2 Preliminaries

In this section, we first give a brief overview of the P-Rank model. We then present notations and formulations of the P-Rank similarity.

2.1 An Overview of P-Rank

The basic essence of the P-Rank similarity model is distilled from the standard concept of SimRank metric [5]. Concretely, the theme of P-Rank involves three facets below:

1. Two distinct objects are similar if they are referenced by similar objects. (in-link recursion)
2. Two distinct objects are similar if they reference similar objects. (out-link recursion)
3. Every object is maximally similar to itself. (a base case)

Based on these facets, P-Rank scores, as described in its name, flowing from the incoming neighbors of objects are able to penetrate the outgoing neighbors. The concise elegance of the P-Rank intuition makes it one of the useful and cutting-edge structural metrics in the link-based analysis of information networks.

2.2 Notations

symbol	designation	symbol	designation
\mathcal{G}	information network	$s(a, b)$	P-Rank score between vertices a and b
$\mathcal{I}(a)$	in-neighbors of vertex a	$C_{\text{in}} / C_{\text{out}}$	in-link / out-link damping factor
$\mathcal{O}(a)$	out-neighbors of vertex a	λ	weighting factor
n	number of vertices in \mathcal{G}	\mathbf{A}	adjacency matrix of \mathcal{G}
m	number of edges in \mathcal{G}	\mathbf{S}	P-Rank similarity matrix of \mathcal{G}
K	number of iterations	\mathbf{I}	identity matrix

2.3 Formulation of P-Rank Model

We first define the network graph, and then introduce the P-Rank formulation.

Network Graph. A *network* is a labeled graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}; l)$, in which (i) \mathcal{V} is a finite set of vertices. A partition of \mathcal{V} is formed by dividing \mathcal{V} into N disjointed domain-specific parts \mathcal{V}_i ($i = 1, \dots, N$) s.t. $\mathcal{V} = \bigcup_{i=1}^N \mathcal{V}_i$ and $\mathcal{V}_i \cap \mathcal{V}_j = \emptyset$ if $i \neq j$; (ii) $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is an edge set, and each vertex-pair $(u, v) \in \mathcal{E}$ denotes an edge from vertex u to v ; (iii) \mathcal{A} is a finite alphabet set, with a labeling function $l : \mathcal{V} \rightarrow \mathcal{A}$.

Based on the network graph, P-Rank model can be formulated as follows:

P-Rank Similarity. Given a network \mathcal{G} , for any two distinct vertices $u, v \in \mathcal{V}$, we define a scoring function $s(u, v) \in [0, 1]$ as follows:

$$s(u, u) = 1; \quad (1)$$

$$s(u, v) = \underbrace{\frac{\lambda \cdot C_{\text{in}}}{|\mathcal{I}(u)| |\mathcal{I}(v)|} \sum_{i=1}^{|\mathcal{I}(u)|} \sum_{j=1}^{|\mathcal{I}(v)|} s(\mathcal{I}_i(u), \mathcal{I}_j(v))}_{\text{in-link part}} + \underbrace{\frac{(1-\lambda) \cdot C_{\text{out}}}{|\mathcal{O}(u)| |\mathcal{O}(v)|} \sum_{i=1}^{|\mathcal{O}(u)|} \sum_{j=1}^{|\mathcal{O}(v)|} s(\mathcal{O}_i(u), \mathcal{O}_j(v))}_{\text{out-link part}} \quad (2)$$

where (i) $\lambda \in [0, 1]$ balances the contributions of in- and out-links, (ii) C_{in} and $C_{\text{out}} \in (0, 1)$ are the damping factors for in- and out-link directions, respectively, (iii) $\mathcal{I}_i(u)$ and $\mathcal{O}_i(u)$ are the individual members of $\mathcal{I}(u)$ and $\mathcal{O}(u)$, respectively, and (iv) $|\mathcal{I}(u)|$ and $|\mathcal{O}(u)|$ are the cardinalities of $\mathcal{I}(u)$ and $\mathcal{O}(u)$, respectively.

It is worth noting that to avoid $s(u, v) = \infty$ when $|\mathcal{I}(\cdot)|$ or $|\mathcal{O}(\cdot)| = 0$, we define: (a) the in-link part of Eq.(2) = 0 if $\mathcal{I}(u) \cup \mathcal{I}(v) = \emptyset$; (b) the out-link part of Eq.(2) = 0 if $\mathcal{O}(u) \cup \mathcal{O}(v) = \emptyset$; (c) $s(u, v) = 0$, if $(\mathcal{I}(u) \cup \mathcal{I}(v)) \cap (\mathcal{O}(u) \cup \mathcal{O}(v)) = \emptyset$.

P-Rank Iterative Paradigm. To compute the P-Rank similarity $s(u, v)$ of Eq.(2), the conventional approach is to construct an iterative paradigm as follows:

$$s^{(0)}(u, v) = \begin{cases} 0, & \text{if } u \neq v; \\ 1, & \text{if } u = v. \end{cases} \quad (3)$$

For each $k = 0, 1, 2, \dots$, the $(k + 1)$ -th iterate $s^{(k+1)}(\cdot, \cdot)$ can be computed as

$$\begin{aligned}
 s^{(k+1)}(u, u) &= 1. \\
 s^{(k+1)}(u, v) &= \frac{\lambda \cdot C_{in}}{|\mathcal{I}(u)||\mathcal{I}(v)|} \sum_{i=1}^{|\mathcal{I}(u)|} \sum_{j=1}^{|\mathcal{I}(v)|} s^{(k)}(\mathcal{I}_i(u), \mathcal{I}_j(v)) \\
 &\quad + \frac{(1-\lambda) \cdot C_{out}}{|\mathcal{O}(u)||\mathcal{O}(v)|} \sum_{i=1}^{|\mathcal{O}(u)|} \sum_{j=1}^{|\mathcal{O}(v)|} s^{(k)}(\mathcal{O}_i(u), \mathcal{O}_j(v)). \tag{4} \\
 s^{(k+1)}(u, v) &= \text{the in-link part of Eq. (4), if } \mathcal{O}(u) \cup \mathcal{O}(v) = \emptyset, \\
 s^{(k+1)}(u, v) &= \text{the out-link part of Eq. (4), if } \mathcal{I}(u) \cup \mathcal{I}(v) = \emptyset.
 \end{aligned}$$

where $s^{(k)}(u, v)$ is the k -th iterative P-Rank score between vertices u and v . It has been proved in [7] that the exact similarity $s(u, v)$ is the supremum of $\{s^{(k)}(u, v)\}_{k=0}^{\infty}$, denoted by $\sup_k \{s^{(k)}(u, v)\}$, i.e.,

$$\lim_{k \rightarrow \infty} s^{(k)}(u, v) = \sup_{k \geq 0} \{s^{(k)}(u, v)\} = s(u, v) \quad (\forall u, v \in \mathcal{V}). \tag{5}$$

3 Accuracy Estimate on P-Rank Iteration

Based on Eq. (4), the iterative P-Rank similarity sequence $\{s^{(k)}(\cdot, \cdot)\}_{k=0}^{\infty}$ was known to converge monotonically [7]. However, the gap between the k -th iterative similarity $s^{(k)}(\cdot, \cdot)$ and the exact similarity $s(\cdot, \cdot)$ still remains unknown. Practically, quantifying this gap is very important in estimating the accuracy of iteratively computing P-Rank. For instance, given a tolerated error $\epsilon > 0$, one may naturally ask ‘‘What is the number of P-Rank iterations needed for achieving such an accuracy?’’.

This motivates us to study the *P-Rank accuracy estimate problem*. Given a network \mathcal{G} , for any iteration number $k = 1, 2, \dots$, it is to find an upper bound ϵ_k of the gap between the k -th iterative similarity $s^{(k)}(\cdot, \cdot)$ and the exact $s(\cdot, \cdot)$, s.t. $|s^{(k)}(u, v) - s(u, v)| \leq \epsilon_k$ for any vertices u and v in \mathcal{G} .

The main result of this section is the following.

Theorem 1. *The P-Rank accuracy estimate problem has a tight upper bound*

$$\epsilon_k = (\lambda C_{in} + (1 - \lambda) C_{out})^{k+1}$$

such that

$$|s^{(k)}(u, v) - s(u, v)| \leq \epsilon_k. \quad (\forall k = 0, 1, \dots, \forall u, v \in \mathcal{V}) \tag{6}$$

A detailed proof of Theorem 1 can be found in [11] due to space limitations.

Theorem 1 provides an a-priori estimate for the gap between iterative and exact P-Rank similarity scores, which solely hinges on the weighing factor λ , the in- and out-link damping factors C_{in} and C_{out} , and the number k of iterations. To ensure more accurate P-Rank estimation results, it can be discerned from $\epsilon_k = (\lambda C_{in} + (1 - \lambda) C_{out})^{k+1} = (\lambda(C_{in} - C_{out}) + C_{out})^{k+1}$ that the smaller choices of C_{in} and C_{out} (i) with a smaller λ if $C_{in} > C_{out}$, or (ii) with a larger λ if $C_{in} < C_{out}$, are more preferable.

Example 1. Setting $C_{in} = 0.6$, $C_{out} = 0.4$, $\lambda = 0.3$, $k = 5$ will produce the following high accuracy :

$$\epsilon_k = (0.3 \times 0.6 + (1 - 0.3) \times 0.4)^{5+1} = 0.0095. \quad \square$$

Notice that the upper bound in Eq.(6) is attainable. Consider the network \mathcal{G}_0 depicted in Figure 1. It is apparent that $s^{(0)}(u, v) = 0$. For $k = 1, 2, \dots$, it can be easily obtained that $s^{(k)}(u, v) = \lambda C_{in} + (1 - \lambda)C_{out}$, which implies that $s(u, v) = \lambda C_{in} + (1 - \lambda)C_{out}$. Hence, in the case of $k = 0$, $|s(u, v) - s^{(k)}(u, v)| = (\lambda C_{in} + (1 - \lambda)C_{out})^{0+1}$ gives the precise upper bound in Eq. (6).

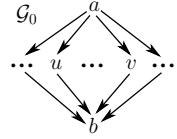


Fig. 1. “=” of Eq. (6) is attainable

Moreover, in the special case of $\lambda = 1$, Eq. (6) can be simplified to the SimRank accuracy estimate problem given in [8]. From this perspective, P-Rank accuracy estimate problem is the extension of Proposition 1 in [8] by jointly considering both in- and out-links of similarity computation.

Conversely, the exponential P-Rank convergence rate in Theorem 1 also implies that the number k of P-Rank iterations needed for attaining a desired accuracy ϵ amounts to

$$k = \lceil \log \epsilon / \log (\lambda \cdot C_{in} + (1 - \lambda) \cdot C_{out}) \rceil .$$

4 Stability Analysis of P-Rank Model

In this section, we investigate the important issues of P-Rank sensitivity and stability. We first reformulate the P-Rank formulae in matrix notations (Subsect. 4.1). Using this representation, we then give a rigorous bound of the P-Rank conditional number for its stability analysis (Subsect. 4.2).

4.1 P-Rank Matrix Representation

We start by introducing the following symbols used throughout this section.

For a network \mathcal{G} with n vertices, we denote by (i) $\mathbf{A} = (a_{i,j}) \in \mathbb{R}^{n \times n}$ the adjacency matrix of \mathcal{G} whose entry $a_{i,j}$ equals 1, if there exists an edge from vertex i to j ; or 0, otherwise, (ii) $\mathbf{S} = (s_{i,j}) \in \mathbb{R}^{n \times n}$ the P-Rank similarity matrix whose entry $s_{i,j}$ is equal to P-Rank score $s(i, j)$, and (iii) $\mathbf{Q} = (q_{i,j}) \in \mathbb{R}^{n \times n}$ and $\mathbf{P} = (p_{i,j}) \in \mathbb{R}^{n \times n}$ the one-step backward and forward transition probability matrix of \mathcal{G} , respectively, whose entries defined as follows:

$$q_{i,j} \triangleq \begin{cases} a_{j,i} / \sum_{j=1}^n a_{j,i}, & \text{if } \mathcal{I}(i) \neq \emptyset; \\ 0, & \text{if } \mathcal{I}(i) = \emptyset. \end{cases} \quad p_{i,j} \triangleq \begin{cases} a_{i,j} / \sum_{j=1}^n a_{i,j}, & \text{if } \mathcal{O}(i) \neq \emptyset; \\ 0, & \text{if } \mathcal{O}(i) = \emptyset. \end{cases} \tag{7}$$

With the above notations, the P-Rank formulae (1) and (2) can be rewritten as¹

$$\mathbf{S} = \lambda C_{in} \cdot \mathbf{Q} \cdot \mathbf{S} \cdot \mathbf{Q}^T + (1 - \lambda) C_{out} \cdot \mathbf{P} \cdot \mathbf{S} \cdot \mathbf{P}^T + (1 - \lambda C_{in} - (1 - \lambda) C_{out}) \cdot \mathbf{I}_n, \tag{8}$$

Also, dividing both sides of Eq. (8) by $(1 - \lambda C_{in} - (1 - \lambda) C_{out})$ results in

$$\mathbf{S}' = \lambda C_{in} \cdot \mathbf{Q} \cdot \mathbf{S}' \cdot \mathbf{Q}^T + (1 - \lambda) C_{out} \cdot \mathbf{P} \cdot \mathbf{S}' \cdot \mathbf{P}^T + \mathbf{I}_n, \text{ and} \tag{9}$$

¹ Although in this case the diagonal entries of \mathbf{S} are not equal to 1, \mathbf{S} still remains diagonally dominant, which ensures that “every vertex is maximally similar to itself”. Li *et al.* [12] has showed that this revision for SimRank matrix representation is reasonable, which can be applied in the similar way to P-Rank.

$$\mathbf{S} = (1 - \lambda C_{in} - (1 - \lambda)C_{out}) \cdot \mathbf{S}'.$$

Comparing Eq. (8) with Eq. (9), we see that the coefficient $(1 - \lambda C_{in} - (1 - \lambda)C_{out})$ of \mathbf{I}_n in Eq. (8) merely contributes an overall multiplicative factor to P-Rank similarity. Hence, setting the coefficient to 1 in Eq. (8) still preserves the *relative* magnitude of the P-Rank score though the diagonal entries of \mathbf{S} in this scenario might not equal 1.

4.2 Conditional Number of P-Rank

Using the P-Rank matrix model of Eq. (9), we next theoretically analyze the stability of P-Rank, by determining its *conditional number* $\kappa_\infty(\cdot)$ in the infinity norm sense, which has important implications for the sensitivity of P-Rank computation. One complicated factor in P-Rank stability analyses is to bound the conditional number $\kappa_\infty(\cdot)$.

Before giving a formal definition of $\kappa_\infty(\cdot)$, we first introduce the following notations:

(i) We denote by $\text{vec}(\mathbf{X}) \in \mathbb{R}^{n^2}$ the *vectorization* of the matrix $\mathbf{X} \in \mathbb{R}^{n \times n}$ formed by stacking the columns of \mathbf{X} into a single column vector, and (ii) the operator \otimes the *Kronecker product* of two matrices.

Taking $\text{vec}(\cdot)$ on both sides of Eq. (9) and applying the Kronecker property of $(\mathbf{B}^T \otimes \mathbf{A})\text{vec}(\mathbf{X}) = \text{vec}(\mathbf{AXB})$ [13], we have

$$\text{vec}(\mathbf{S}) = \lambda C_{in}(\mathbf{Q} \otimes \mathbf{Q})\text{vec}(\mathbf{S}) + (1 - \lambda)C_{out}(\mathbf{P} \otimes \mathbf{P})\text{vec}(\mathbf{S}) + \text{vec}(\mathbf{I}_n).$$

Rearranging terms in the above equation produces

$$\underbrace{(\mathbf{I}_{n^2} - \lambda C_{in}(\mathbf{Q} \otimes \mathbf{Q}) - (1 - \lambda)C_{out}(\mathbf{P} \otimes \mathbf{P}))}_{\triangleq \mathbf{M}} \cdot \underbrace{\text{vec}(\mathbf{S})}_{\triangleq \mathbf{s}} = \underbrace{\text{vec}(\mathbf{I}_n)}_{\triangleq \mathbf{b}}. \tag{10}$$

Notice that Eq. (10) is a *linear* system of n^2 equations in terms of $s_{i,j}$ in nature as it can be reduced in the form of $\mathbf{M} \cdot \mathbf{s} = \mathbf{b}$, as illustrated in Eq. (10).

Based on Eq. (10), we now give a formal definition of *P-Rank conditional number*.

Definition 1 (P-Rank conditional number). For a network \mathcal{G} , let \mathbf{Q} and \mathbf{P} be the one-step backward and forward transition probability matrix of \mathcal{G} , respectively, defined by Eq. (7), and let

$$\mathbf{M} \triangleq \mathbf{I}_{n^2} - \lambda C_{in}(\mathbf{Q} \otimes \mathbf{Q}) - (1 - \lambda)C_{out}(\mathbf{P} \otimes \mathbf{P}). \tag{11}$$

The P-Rank conditional number of \mathcal{G} , denoted by $\kappa_\infty(\mathcal{G})$, is defined as

$$\kappa_\infty(\mathcal{G}) \triangleq \|\mathbf{M}\|_\infty \cdot \|\mathbf{M}^{-1}\|_\infty, \tag{12}$$

where $\|\cdot\|_\infty$ is the ∞ -norm that returns the maximum absolute row sum of the matrix.

The conditional number is introduced for being applied to P-Rank stability analysis.

Theorem 2. Given the network \mathcal{G} , for any weighting factor $\lambda \in [0, 1]$ and in- and out-link damping factors $C_{in}, C_{out} \in (0, 1)$, the P-Rank problem has the following tight bound of conditional number

$$\kappa_\infty(\mathcal{G}) \leq \frac{1 + \lambda \cdot C_{in} + (1 - \lambda) \cdot C_{out}}{1 - \lambda \cdot C_{in} - (1 - \lambda) \cdot C_{out}}. \tag{13}$$

² In what follows, we shall base our techniques on the P-Rank matrix form of Eq. (9).

For the interest of space, please refer to [11] for a detailed proof of Theorem 2.

Theorem 2 provides a tight upper bound of the P-Rank conditional number $\kappa_\infty(\mathcal{G})$. Intuitively, the value of $\kappa_\infty(\mathcal{G})$ measures how stable the P-Rank similarity score is to the changes either in the link structure of the network \mathcal{G} (by inserting or deleting vertices or edges), or in the damping and weighting factors of C_{in} , C_{out} and λ .

To get a feel for how $\kappa_\infty(\mathcal{G})$ affects P-Rank sensitivity, we denote by $\Delta\mathbf{A}$ the updates in \mathcal{G} to the adjacency matrix \mathbf{A} , $\Delta\mathbf{M}$ (resp. $\Delta\mathbf{s}$) the changes (caused by $\Delta\mathbf{A}$) to \mathbf{M} (resp. \mathbf{s}) defined in Eq. (10). As Eq. (10) is a linear equation, it is known that

$$\frac{\|\Delta\mathbf{s}\|_\infty}{\|\mathbf{s}\|_\infty} \leq \kappa_\infty(\mathcal{G}) \cdot \frac{\|\Delta\mathbf{M}\|_\infty}{\|\mathbf{M}\|_\infty} \leq \frac{1 + \lambda \cdot C_{in} + (1 - \lambda) \cdot C_{out}}{1 - \lambda \cdot C_{in} - (1 - \lambda) \cdot C_{out}} \cdot \frac{\|\Delta\mathbf{M}\|_\infty}{\|\mathbf{M}\|_\infty}. \quad (14)$$

This tells that the small $\kappa_\infty(\mathcal{G})$ (i.e., small choices for C_{in} and C_{out}) would make P-Rank stable, implying that a small change $\Delta\mathbf{M}$ in the link structure to \mathbf{M} may not cause a large change $\Delta\mathbf{s}$ in P-Rank scores. Conversely, the large value of $\kappa_\infty(\mathcal{G})$ would make P-Rank ill-conditioned.

To see how the weighting factor λ affects $\kappa_\infty(\mathcal{G})$, we compute the partial derivatives of the bound for $\kappa_\infty(\mathcal{G})$ w.r.t. λ to get

$$\frac{\partial}{\partial \lambda} \left(\frac{1 + \lambda \cdot C_{in} + (1 - \lambda) \cdot C_{out}}{1 - \lambda \cdot C_{in} - (1 - \lambda) \cdot C_{out}} \right) = \frac{2(C_{in} - C_{out})}{(1 - \lambda \cdot C_{in} - (1 - \lambda) \cdot C_{out})^2}, \quad (15)$$

which implies that when $C_{in} > C_{out}$ (resp. $C_{in} < C_{out}$), for the increasing λ , a small change in \mathcal{G} may result in a large (resp. small) change in P-Rank, which makes P-Rank an ill-conditioned (resp. a well-conditioned) problem; when $C_{in} = C_{out}$, the value of $\kappa_\infty(\mathcal{G})$ is independent of λ .

5 An Efficient Algorithm for P-Rank Estimating on Undirected Graphs

In this section, we study the P-Rank optimization problem over undirected networks.

The key idea in our optimization is to maximally use the adjacency matrix \mathbf{A} to characterize the P-Rank similarity \mathbf{S} as a power series in a function form like $\mathbf{S} = \sum_{k=0}^{+\infty} f(\mathbf{A}^k)$. The rationale behind this is that $\mathbf{A} = \mathbf{A}^T$ for undirected networks, implying that there exists a diagonal matrix \mathbf{D} such that $\mathbf{Q} = \mathbf{P} = \mathbf{D} \cdot \mathbf{A}$ in Eq. (8). Hence, computing \mathbf{S} boils down to solving $\sum_{k=0}^{+\infty} f(\mathbf{A}^k)$. For computing $f(\mathbf{A}^k)$, it is far less costly to first diagonalize \mathbf{A} into $\mathbf{\Lambda}$ via eigen-decomposition than to compute \mathbf{A}^k in a brute-force way. Then calculating $f(\mathbf{A}^k)$ reduces to computing the function on each eigenvalue for \mathbf{A} .

More specifically, the main result of this section is the following.

Theorem 3. *For undirected networks, the P-Rank similarity problem of Eq. (8) can be solvable in $O(n^3)$ worst-case time.*

It is worth noting that SimRank over undirected graphs is a special case of the more general P-Rank when $\lambda = 1$ and $C_{in} = C_{out}$. In contrast to the $O(n^3 + Kn^2)$ -time of

Algorithm 1. ASAP ($\mathcal{G}, \lambda, C_{in}, C_{out}$)

-
- Input** : a labeled undirected network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}; l)$, the weighting factor λ , and in- and out-link damping factors C_{in} and C_{out} .
- Output**: similarity matrix $\mathbf{S} = (s_{i,j})_{n \times n}$ with $s_{i,j}$ denoting P-Rank score between vertices i and j .
- 1 initialize the adjacency matrix \mathbf{A} of \mathcal{G} ;
 - 2 compute the diagonal matrix $\mathbf{D} = \text{diag}(d_{1,1}, d_{2,2}, \dots, d_{n,n})$ with its entry $d_{i,i} = (\sum_{j=1}^n a_{i,j})^{-1}$, if $\sum_{j=1}^n a_{i,j} \neq 0$; and $d_{i,i} = 0$, otherwise;
 - 3 compute the auxiliary matrix $\mathbf{T} = \mathbf{D}^{1/2} \cdot \mathbf{A} \cdot \mathbf{D}^{1/2}$
 - 4 decompose \mathbf{T} into the diagonal matrix $\mathbf{\Lambda} = \text{diag}(\Lambda_{1,1}, \Lambda_{2,2}, \dots, \Lambda_{n,n})$ and the orthogonal \mathbf{U} via QR factorization *s.t.* $\mathbf{T} = \mathbf{U} \cdot \mathbf{\Lambda} \cdot \mathbf{U}^T$;
 - 5 compute the auxiliary matrix $\mathbf{\Gamma} = (\Gamma_{i,j})_{n \times n} = \mathbf{U}^T \cdot \mathbf{D}^{-1} \cdot \mathbf{U}$ and $\mathbf{V} = \mathbf{D}^{1/2} \cdot \mathbf{U}$ and the constant $C = \lambda C_{in} + (1 - \lambda) C_{out}$;
 - 6 compute the matrix $\mathbf{\Psi} = (\psi_{i,j})_{n \times n}$ whose entry $\psi_{i,j} = \Gamma_{i,j} / (1 - C \cdot \Lambda_{i,i} \cdot \Lambda_{j,j})$;
 - 7 compute the P-Rank similarity matrix $\mathbf{S} = (1 - C) \cdot \mathbf{V} \cdot \mathbf{\Psi} \cdot \mathbf{V}^T$;
 - 8 **return** \mathbf{S} ;
-

SimRank optimization over undirected graphs in our early work [14], this work further optimizes P-Rank time in $O(n^3)$ with no need for iteration.

To show Theorem 3, we first characterize P-Rank elegantly on undirected graphs.

Theorem 4. For the undirected network \mathcal{G} with n vertices, let (i) $\mathbf{A} = (a_{i,j}) \in \mathbb{R}^{n \times n}$ be the adjacency matrix of \mathcal{G} , (ii) $\mathbf{D} \in \mathbb{R}^{n \times n}$ is a diagonal matrix [3].

$$\mathbf{D} = \text{diag}\left(\left(\sum_{j=1}^n a_{1,j}\right)^{-1}, \dots, \left(\sum_{j=1}^n a_{n,j}\right)^{-1}\right), \quad (16)$$

and (iii) \mathbf{U} and $\mathbf{\Lambda}$ the eigen-decomposition results of the matrix $\mathbf{D}^{1/2} \mathbf{A} \mathbf{D}^{1/2}$, in which $\mathbf{U} \in \mathbb{R}^{n \times n}$ is an orthogonal matrix whose columns are all eigenvectors of $\mathbf{D}^{1/2} \mathbf{A} \mathbf{D}^{1/2}$, and $\mathbf{\Lambda} = (\Lambda_{i,j}) \in \mathbb{R}^{n \times n}$ is an diagonal matrix with its diagonal entries being all eigenvalues of $\mathbf{D}^{1/2} \mathbf{A} \mathbf{D}^{1/2}$.

Then the P-Rank similarity matrix \mathbf{S}' of Eq. (9) can be explicitly computed as [4].

$$\mathbf{S}' = \mathbf{D}^{1/2} \mathbf{U} \cdot \mathbf{\Psi} \cdot \mathbf{U}^T \mathbf{D}^{1/2}, \quad (17)$$

where

$$\mathbf{\Psi} = (\Psi_{i,j})_{n \times n} = \left(\frac{[\mathbf{U}^T \mathbf{D}^{-1} \mathbf{U}]_{i,j}}{1 - (\lambda \cdot C_{in} + (1 - \lambda) \cdot C_{out}) \Lambda_{i,i} \Lambda_{j,j}} \right)_{n \times n}, \quad \text{and} \quad (18)$$

$[\mathbf{U}^T \mathbf{D}^{-1} \mathbf{U}]_{i,j}$ denotes the (i, j) -entry of the matrix $\mathbf{U}^T \mathbf{D}^{-1} \mathbf{U}$.

Due to space limitations, a detailed proof of this theorem can be found in [11].

We next prove Theorem 3 by providing an algorithm for P-Rank computation over undirected networks.

³ We define $\frac{1}{0} \triangleq 0$, thereby avoiding division by zero when the column/row sum of \mathbf{A} equals 0.

⁴ $\mathbf{D}^{1/2}$ is a diagonal matrix whose diagonal entries are the principal square root of those of \mathbf{D} .

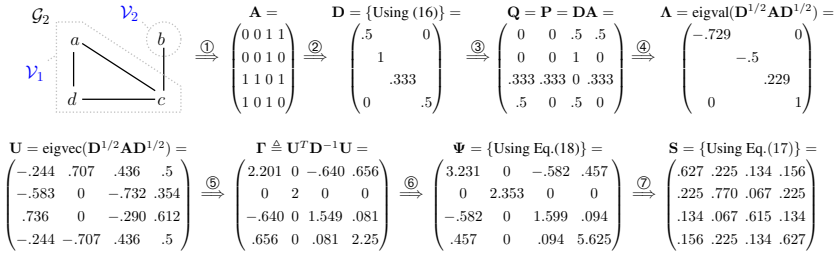


Fig. 2. An example of computing P-Rank over undirected network \mathcal{G}_2 via non-iterative method

Algorithm. The algorithm, referred to as **ASAP**, is shown in Algorithm 1. It takes as input a labeled undirected network \mathcal{G} , a weighting factor λ , and in- and out-link damping factors C_{in} and C_{out} ; and it returns the P-Rank similarity matrix $\mathbf{S} = (s_{i,j})_{n \times n}$ of all vertex-pairs in \mathcal{G} .

The algorithm **ASAP** works as follows. It first initializes the adjacency matrix \mathbf{A} of the network \mathcal{G} (line 1). Using \mathbf{A} , it then compute the auxiliary diagonal matrix \mathbf{D} whose (i, i) -entry equals the reciprocal of the i -th column sum of \mathbf{A} , if this reciprocal exists; or 0, otherwise (line 2). **ASAP** then uses QR eigen-decomposition [13] to factorize $\mathbf{D}^{1/2}\mathbf{A}\mathbf{D}^{1/2}$ as $\mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, in which all columns of \mathbf{U} are the eigenvectors of $\mathbf{D}^{1/2}\mathbf{A}\mathbf{D}^{1/2}$, and all diagonal entries of $\mathbf{\Lambda}$ are the corresponding eigenvalues of $\mathbf{D}^{1/2}\mathbf{A}\mathbf{D}^{1/2}$ (lines 3-4). Utilizing \mathbf{U} and $\mathbf{\Lambda}$, it calculates $\mathbf{\Psi}$ (lines 5-6) to obtain the similarity matrix \mathbf{S} (lines 7-8), which can be justified by Eqs. (17) and (18).

We now give a running example to show how **ASAP** computes the P-Rank similarity matrix \mathbf{S} for a given network.

Example 2. Consider a labeled undirected network \mathcal{G}_2 with 4 vertices $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 = \{a, c, d\} \cup \{b\}$ and 4 edges $\mathcal{E} = \{(a, c), (a, d), (c, d), (b, c)\}$. Figure 2 depicts the detailed process of computing \mathbf{S} step by step with no need for any iteration. **ASAP** returns \mathbf{S} as the P-Rank matrix, which is exactly the solution to the P-Rank formula of Eq. (8).

To complete the proof of Theorem 3, we next show that algorithm **ASAP** has cubic-time complexity bound in the number of vertices.

Complexity. (i) In lines 1-2, computing the diagonal \mathbf{D} and $\mathbf{T} = \mathbf{D}^{1/2}\mathbf{A}\mathbf{D}^{1/2}$ takes $O(m)$ and $O(n^2)$ time, respectively, with n and m being the number of vertices and edges in \mathcal{G} respectively. (ii) In line 3, QR factorization of \mathbf{T} into the orthogonal \mathbf{U} and the diagonal $\mathbf{\Lambda}$ requires $O(n^3)$ worst-case time. (iii) In lines 4-8, computing the auxiliary matrices $\mathbf{\Gamma}$, \mathbf{V} , $\mathbf{\Psi}$ and the similarity matrix \mathbf{S} yields $O(n^3)$, $O(n^2)$, $O(n^2)$ and $O(n^3)$, respectively, which can be bounded by $O(n^3)$. Combining (i), (ii) and (iii), the total time of **ASAP** is in $O(n^3)$.

6 Experimental Evaluation

We next present an experimental study of our P-Rank estimating methods. Using real-life and synthetic data, we conducted two sets of experiments to evaluate the accuracy, stability, computational efficiency of our approaches for similarity estimation v.s. (a)

the conventional pruning P-Rank iterative paradigm [7] and (b) the memoization-based P-Rank algorithm [8,7].

Experimental setting. We used real-life data and synthetic data.

(1) *Real-life data.* The real-life data was taken from DBLP [5]. We extracted the 10-year (from 1998 to 2007) author-paper information from the entire DBLP dataset. We picked up papers published on 6 major conferences (“ICDE”, “VLDB”, “SIGMOD”, “WWW”, “SIGIR” and “KDD”). Every two years made a time step. For each time step, we built a co-authorship network incrementally from the one of previous time step. We chose the relationship that there is an edge between authors if one author wrote a paper with another. The sizes of these DBLP networks are as follows:

DBLP Data	1998-1999	1998-2001	1998-2003	1998-2005	1998-2007
n	1,525	3,208	5,307	7,984	10,682
m	5,929	13,441	24,762	39,399	54,844

(2) *Synthetic data.* We also used a C++ boost generator to produce graphs, controlled by two parameters: the number n of vertices, and the number m of edges. We then produced a set of 5 networks (undirected RAND data) by increasing the vertex size n from 100K to 1M with edges randomly chosen.

(3) *Algorithms.* We have implemented the following algorithms in C++: (a) our algorithm ASAP ; (b) the conventional P-Rank iterative algorithm Iter [7] with the radius-based pruning technique; (c) the memoization-based algorithm Memo [8] applied on P-Rank; (d) SimRank optimized algorithm AUG [14] over undirected graphs.

The experiments were run on a machine with a Pentium(R) Dual-Core (2.0GHz) CPU and 4GB RAM, using Windows Vista. Each experiment was repeated 5 times and the average is reported here.

For ease and fairness of comparison, the following parameters were used as default values (unless otherwise specified). We set the in-link damping factor $C_{in} = 0.8$, the out-link damping factor $C_{out} = 0.6$, the weighting factor $\lambda = 0.5$, the total iteration number $k = 10$, the desired accuracy $\epsilon = 0.001$.

Experimental Results. We now present our findings.

Exp-1: Accuracy. In the first set of experiments, we evaluated the accuracy of P-Rank iteration in Eq. (6) using synthetic and real data. We also investigated the impact of in- and out-link damping factors on P-Rank accuracy, using synthetic data.

We considered various weighting factors λ from 0 to 1 for 5 RAND networks. For each RAND data with vertex size ranged from 0.2M to 1M, fixing the damping factors $C_{in} = 0.8$ and $C_{out} = 0.6$, we varied the number k of P-Rank iterations in Eq. (4) ranged from 2 to 20 for each vertex-pair. Due to space limitations, Figure 3(a) only reports the results over the RAND network (with 1M vertices), which visualizes the P-Rank accuracy *w.r.t.* the number of iterations performed. Here, the accuracy is measured

⁵ <http://www.informatik.uni-trier.de/~ley/db/>

by the absolute difference between the iterative P-Rank and the exact solution⁶. Note that the logarithmic scale is chosen across the y -axis in Figure 3(a) to provide a more illustrative look for the asymptotic rate of P-Rank convergence. For each fixed λ , the downward lines for P-Rank iterations reveal an exponential accuracy as k increases, as expected in Theorem 1. We also observe that the larger λ may dramatically increase the slope of a line, which tells that increasing the weighting factor decreases the rate of convergence for P-Rank iteration.

Using the same RAND, we fixed the desired accuracy $\epsilon = 0.001$ and varied damping factors by increasing C_{in} and C_{out} from 0.1 to 0.9. Fixing $C_{out} = 0.6$, the result of varying C_{in} is reported in Figure 3(b) (for space constraints, a similar result of varying C_{out} is omitted), in which the x -axis represents the value of in-link damping factor, and y -axis gives the number of iterations needed for attaining the given accuracy ϵ . When $\lambda = 0$, the curve in Figure (b) visually approaches a horizontal line. This is because in this case, P-Rank boils down to an iterative form of the reversed SimRank with no in-link similarity considered, which makes C_{in} insensitive to the final P-Rank score. When $0 < \lambda \leq 1$, k shows a general increased tendency as C_{in} is growing. This tells us that small choices of damping factors may reduce the number of iterations required for a fixed accuracy, and hence, improves the efficiency of P-Rank, as expected.

To evaluate the impact of both C_{in} and C_{out} w.r.t. the accuracy, we used the real DBLP data. We only report the result on DBLP 1998-2007 data in Figure 3(c), which shows a 3D shaded surface from the average of accuracy value for all vertex-pairs on z -axis when we fixed $k = 10$ and $\lambda = 0.5$, and varied C_{in} and C_{out} in x -axis and y -axis, respectively. It can be seen that the residual becomes huge only when C_{in} and C_{out} are both increasing to 1; and the iterative P-Rank is accurate when C_{in} and $C_{out} < 0.6$. This explains why small choices of damping factors are suggested in P-Rank iteration.

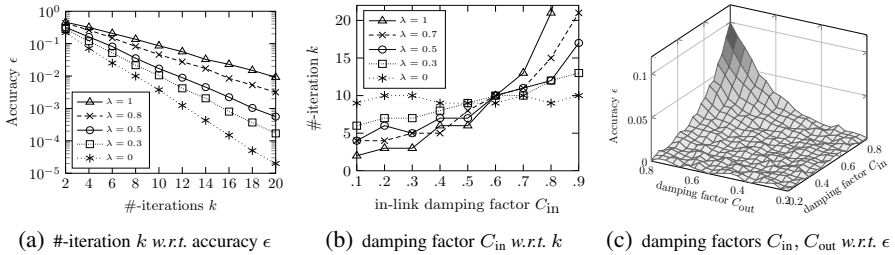


Fig. 3. Experimental Results on P-Rank Accuracy

Exp-2: Stability. We evaluated P-Rank stability using synthetic data and DBLP.

Fixing the value of out-link damping factor $C_{out} = 0.6$, we varied the values of C_{in} from 0.2 to 0.8. The result over RAND 1M data is reported in Figure 4(a), in which x -axis indicates various weighting factors λ with their sizes ranged from 0 to 1. Accordingly, varying λ from 0 and 1, Figure 4(b) visualizes the impact of in-link damping factor C_{in} on P-Rank stability when $C_{out} = 0.6$ is fixed.

⁶ To select the P-Rank “exact” solution $s(\cdot, \cdot)$, we used the Cauchy’s criterion for convergence and regarded the 100th iterative $s^{(100)}(\cdot, \cdot)$ score as the “exact” one s.t. $|s^{(100)}(\cdot, \cdot) - s^{(101)}(\cdot, \cdot)| \ll 1 \times 10^{-10}$.

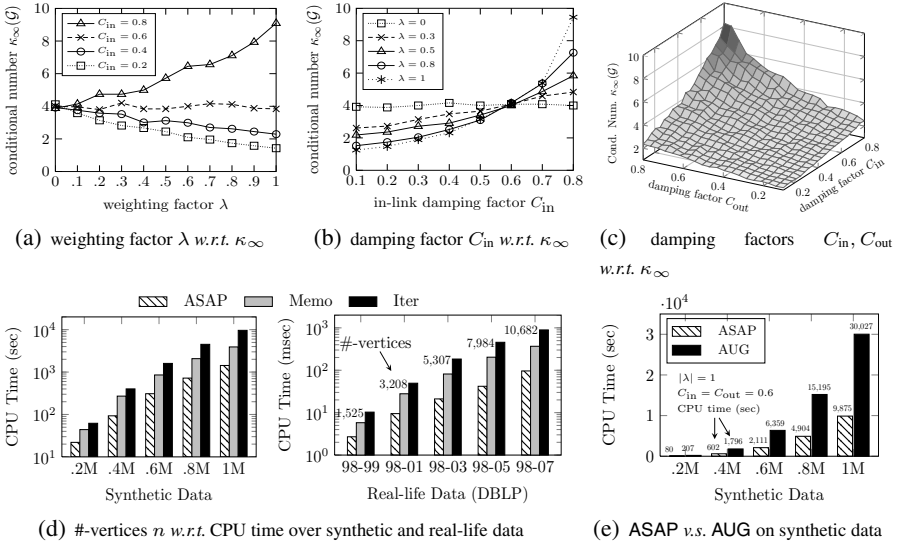


Fig. 4. Experimental Results on P-Rank Stability and Time Efficiency

The results in Figures 4(a) and 4(b) both show that increasing weighting factor λ induces a large P-Rank conditional number when $C_{in} > 0.6$. Notice that for different C_{in} , there is one common point $(\lambda, \kappa_\infty) = (0, 4)$ of intersection of all curves in Figure 4(a); correspondingly, in the extreme case of $\lambda = 0$, the curve in Figure 4(b) approaches to a horizontal line. These indicate that varying C_{in} when $\lambda = 0$ has no effect on the stability κ_∞ of P-Rank, for in this case only the contribution of out-links is considered for computing P-Rank similarity. When $C_{in} < 0.6$, however, $\kappa_\infty(\mathcal{G})$ is decreased as λ grows. This tells that small quantities of weighting factor and damping factors yield small P-Rank conditional numbers, and thus make the P-Rank well-conditioned, as expected in Theorem 2.

For real-life datasets, Figure 4(c) comprehensively depicts in 3D view the impacts of both C_{in} and C_{out} on P-Rank stability over DBLP 1M data, in which x - and y -axis represent in- and out-link damping factors respectively, and z -axis stands for the P-Rank conditional number. The result demonstrates that P-Rank is comparatively stable when both C_{in} and C_{out} are small (less than 0.6). However, when C_{in} and C_{out} are approaching to 1, P-Rank is ill-conditioned and not very useful since small perturbations in similarity computation may cause P-Rank scores drastically altered, which carries the risk of producing nonsensical similarity results. In light of this, small choices of damping factors are preferable.

Exp-3: Time Efficiency over Undirected Networks. In the third set of experiments, we used synthetic RAND and the real DBLP data to evaluate the benefits of the time-efficient algorithm ASAP. Figure 4(d) compares the performance of ASAP with those of Memo and Iter on both datasets. We use the logarithmic scale on the CPU time (y -axis). The iteration number for Iter and Memo is set to 10. Note that different time unit is chosen across the vertical axis in the two plots of Fig 4(d) to provide a clear

look for each bar shape. (i) Varying the number of vertices from 200K to 1M, the result on RAND indicates that **ASAP** outperformed **Memo** and **Iter**; the computational time of **ASAP** has almost one order of magnitude faster than **Iter**, *i.e.*, computing P-Rank similarity from the explicit characterization of its solution is efficient. In most cases, there are a considerable amount of repeated iterations for **Iter** and **Memo** to reach a fixed-point of P-Rank scores, and these impede their time efficiency in P-Rank computation. (ii) The result on DBLP also shows the running time with respect to the number of nodes for P-Rank estimation when the sizes of DBLP data increased from 1.5K to 10K. In all cases, **ASAP** performed the best, by taking advantage of its non-iterative paradigm.

To compare the performances of **ASAP** and **AUG**, we applied them to compute Sim-Rank similarities over synthetic RAND data, by setting $\lambda = 1$ for **ASAP** (a special case of P-Rank without out-links consideration). Figure 4(e) reports the result over synthetic RAND data. It can be seen that **ASAP** runs approx. 3 times faster than **AUG** though the CPU time of the **ASAP** and **AUG** are of the same order of magnitude. The reason is that after eigen-decomposition, **AUG** still requires extra iterations to be performed in the small eigen-subspace, which takes a significant amount of time, whereas **ASAP** can straightforwardly compute similarities in terms of eigenvectors with no need for iterations.

7 Related Work

There has been a surge of studies (*e.g.*, [2][15][3][9][4][5][16][6]) on link-based analysis over information networks in recent years. PageRank became popular since the famous result of Page *et al.* [6] was used by the Google search engine for ranking web sites based on link structures. Since then, a host of new ranking algorithms for web pages have been developed. The famous results include the HITS algorithm [2] proposed by Jon *et al.* (now used `www.ask.com`), SimRank [17][10][5], SimFusion [15][4] and P-Rank algorithm [7].

SimRank [5] is a recursive structural similarity measure based on the intuition that “two objects are similar if they are related to similar objects”, which extends the Bibliometric Coupling and Co-citation [3][1] beyond the local neighborhood so that the information of the entire network can be exploited globally. A naive iterative approach was initially proposed in [5] to compute the SimRank score with a pruning mechanism, which is in quartic worst-case time. Several optimization problems were investigated for SimRank estimation, including the pair-wise iterative methods [5][8], matrix-based approaches [8][12][17][14] and probabilistic algorithms [18].

More recently, Zhao *et al.* [7] presented a new P-Rank model when noticing the limited information problem of SimRank —the similarity scores are only determined by their in-link relationships. P-Rank measure refines SimRank by jointly considering both in- and out-links of entity pairs. The conventional algorithm for computing P-Rank is based on iterative techniques, which still requires quartic time complexity. To optimize its computational time, a similar memoization approach in [8] can be applied to P-Rank, which reduces its time to be cubic in the worst case. In comparison, our work focuses on the problems of P-Rank accuracy, stability and computational time over undirected graphs.

Closer to this work are [14,8]. A time-efficient algorithm AUG for SimRank computation was proposed on undirected graphs in [14], which is in $O(n^3 + kn^2)$ time. In contrast, we further improve [14] (i) by providing a non-iterative $O(n^3)$ -time algorithm that explicitly characterizes the similarity solution, and (ii) by extending SimRank to the general P-Rank measure. An accuracy estimation for SimRank was addressed in [8], it differs from this work in that our focus is on P-Rank estimation, in which the accuracy depends on both in- and out-link damping factors rather than the in-link damping factor alone.

8 Conclusions

In this study, several P-Rank problems were investigated. First, we have proposed an accuracy estimate for the P-Rank iteration, by finding out the exact number of iterations needed to attain a given accuracy. Second, we have introduced the notion of P-Rank conditional number based on the matrix representation of P-Rank. A tight bound of P-Rank conditional number has been provided to show how the weighting factor and the damping factors affect the stability of P-Rank. Finally, we have also devised an $O(n^3)$ -time algorithm to deal with the P-Rank optimization problem over undirected networks. Our empirical results have verified the accuracy, stability and computational efficiency of our approaches.

References

1. Amsler, R.: Application of citation-based automatic classification. Technical Report, The University of Texas at Austin, Linguistics Research Center (December 1972)
2. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. *J. ACM* 46(5), 604–632 (1999)
3. Small, H.: Co-citation in the scientific literature: A new measure of the relationship between two documents. *J. Am. Soc. Inf. Sci.* 24(4), 265–269 (1973)
4. Xi, W., Fox, E.A., Fan, W., Zhang, B., Chen, Z., Yan, J., Zhuang, D.: Simfusion: measuring similarity using unified relationship matrix. In: *SIGIR* (2005)
5. Jeh, G., Widom, J.: Simrank: a measure of structural-context similarity. In: *KDD*, pp. 538–543 (2002)
6. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab (November 1999)
7. Zhao, P., Han, J., Sun, Y.: P-rank: a comprehensive structural similarity measure over information networks. In: *CIKM 2009: Proceeding of the 18th ACM Conference on Information and Knowledge Management* (2009)
8. Lizorkin, D., Velikhov, P., Grinev, M.N., Turdakov, D.: Accuracy estimate and optimization techniques for simrank computation. *VLDB J.* 19(1) (2010)
9. Zhou, Y., Cheng, H., Yu, J.X.: Graph clustering based on structural/attribute similarities. *PVLDB* 2(1) (2009)
10. Antonellis, I., Garcia-Molina, H., Chang, C.C.: Simrank++: query rewriting through link analysis of the click graph. *PVLDB* 1(1) (2008)
11. Yu, W.: Efficiently computing p-rank similarity on large networks. Technical report, The University of New South Wales (2011), <http://www.cse.unsw.edu.au/~weiren/you-tr-waim2011.pdf>

12. Li, C., Han, J., He, G., Jin, X., Sun, Y., Yu, Y., Wu, T.: Fast computation of simrank for static and dynamic information networks. In: EDBT (2010)
13. Golub, G.H., Loan, C.F.V.: Matrix computations, 3rd edn. John Hopkins University Press, Baltimore (1996)
14. Yu, W., Lin, X., Le, J.: Taming computational complexity: Efficient and parallel simrank optimizations on undirected graphs. In: Chen, L., Tang, C., Yang, J., Gao, Y. (eds.) WAIM 2010. LNCS, vol. 6184, pp. 280–296. Springer, Heidelberg (2010)
15. Cai, Y., Zhang, M., Ding, C.H.Q., Chakravarthy, S.: Closed form solution of similarity algorithms. In: SIGIR, pp. 709–710 (2010)
16. Yu, W., Lin, X., Le, J.: A space and time efficient algorithm for simrank computation. In: APWeb, pp. 164–170 (2010)
17. He, G., Feng, H., Li, C., Chen, H.: Parallel simrank computation on large graphs with iterative aggregation. In: KDD (2010)
18. Fogaras, D., Rácz, B.: Scaling link-based similarity search. In: WWW (2005)

Efficient Name Disambiguation in Digital Libraries

Jia Zhu¹, Gabriel Fung², and Liwei Wang³

¹ School of ITEE, The University of Queensland, Australia
jiazhu@itee.uq.edu.au

² iConcept Press
gabriel.fung@iconceptpress.com

³ Wuhan University, China
liwei.wang@whu.edu.cn

Abstract. In digital libraries, ambiguous author names occur due to the existence of multiple authors with the same name or different name variations for the same person. Most of the previous works to solve this issue also known as name disambiguation often employ hierarchal clustering approaches based on information inside the citation records, e.g. co-authors and publication titles. In this paper, we propose an approach that can effectively identify and retrieve information from web pages and use the information to disambiguate authors. Initially, we implement a web pages identification model by using a neural network classifier and traffic rank. Considering those records can not be found directly in personal pages, we then enhance the model to handle such case during the clustering process with performance improvement. We examine our approach on a subset of digital library records and the result is reasonable effective.

1 Introduction

Name disambiguation in digital libraries refers to the task of attributing the citation records to the proper authors [13]. It is very common that several authors share the same name in a typical digital library. In general, there are two kinds of methods to resolve this issue, supervised learning [12, 3, 11] and unsupervised learning [2, 10, 12, 15] also known as clustering. Supervised learning methods require heavy human labeling and expensive training time, which is unfeasible for large-scale digital libraries. Unsupervised learning methods also called clustering and do not need any data for training. The quality of clustering strongly depends on the quality of features and similarity measurements while features should be chosen wisely so that they are symptomatic for the different clusters.

Existing approaches based on these two methods have not gained sound results due to only limited information is used, such as author names and paper titles. In order to enrich the information, an alternative way is to find personal information from the web, personal homepages for instance. The information in these resource is extremely useful to disambiguate authors. One hypothesis of this work is that

we can find such pages by submitting composite queries to a web search engine. The challenge here is to identify which web pages address a single author because some pages return from the search engine might be public pages, e.g. DBLP. Additionally, we also need to consider those records can not be found in any web pages.

In this paper, we propose an approach that uses the web pages identified by a neural network classifier as a source of additional information. This model uses the information by leveraging a traditional hierarchical clustering method with re-clustering mechanism that deeply processes those citation records can not be found in the web pages to disambiguate authors and groups those citations records refer to one person into a group. In addition, we enhance our approach by involving traffic rank analysis in order to reduce the web information extraction cost and also minimize the risk of records being merged incorrectly. This approach is not only suitable for name disambiguation in digital libraries but also can be applied to other name ambiguity related issues.

The rest of this paper is organized as follows. In Section 2, we discuss related work in name disambiguation, especially the use of the web for similar name disambiguation tasks. In Section 3, we describe details of our approach including the web pages identification model and enhancement. In Section 4, we describe our experiments, evaluation metrics, and results. We also conclude this study in Section 5.

2 Related Work

Han et al. [4] proposed an unsupervised learning approach using K-way spectral clustering to solve the name disambiguation problem. Although this method is fast but it depends heavily on the initial partition and the order of processing each data point. Furthermore, this clustering method may not work well when there are many ambiguous authors in the dataset. They also proposed a supervised learning framework (by using SVM and Naive Bayes) to solve the name disambiguation problem [3]. Although the accuracy of this method is high but it relies heavily on the quality of the training data, which is difficult to obtain. In addition, this method will assemble multiple attributes into a training network without any logical order, which may deteriorate the overall performance of the framework.

Yin et al. [18] proposed a semi-supervised learning method which uses SVM to train different linkage weights in order to distinguish objects with identical names. In general, this approach may obtain sound results. Yet, it is quite difficult to generate appropriate linkage weights if the linkages are shared by many different authors. Song et al. [15] used a topic-based model to solve the name disambiguation problem. Two hierarchical Bayesian text models, Probabilistic Latent Semantic Analysis (PLSA) and Latent Dirichlet Allocation (LDA), are used. In general, it achieves better experimental results over the other approaches. Unfortunately, this approach is difficult to implement because it requires too much labor intensive preprocessing and relies on some special assumptions. Their work have been extended further with web information in [17].

In addition, an approach had been proposed in [6] that analyzes not only features but also inter relationships among each object in a graph format to improve the disambiguation quality. Based on their ideas, authors in [21] proposed a taxonomy based clustering model that can enrich the information among citation records. Besides basic metadata, some approaches used additional information obtained from the web in order to improve the accuracy of clustering results.

Kang et al. [7] explored the net effects of coauthorship on author name disambiguation and used a web-assisted technique of acquiring implicit coauthors of the target author to be disambiguated. Pairs of names are submitted as queries to search engines to retrieve documents that contain both author names, and these documents are scanned to extract new author names as coauthors of the original pair. However, their work can not identify which pages are personal pages. Tan et al. [16] presented an approach that analyzes the results of automatically-crafted web searches and proposed a model called Inverse Host Frequency (IHF). They used the URLs returned from search engine as feature to disambiguate author names. They did not exploit the content of the documents returned by the queries, what may mislead the results since many returned URLs do not refer to pages containing publications or the pages are not of a single author. In order to solve these issues, Pereira et al [14] proposed a model to exploit the content of the web to identify if the web page is a personal page. Their approach only use the text in the <title> tag or the first 20 lines of the pages to identify pages which are not sufficient because some authors might put their emails on the bottom of the pages.

3 Our Approach

Given a list of citations C , where each citation $c_i \in C$ is a bibliographic reference containing at least a list of author names $A_i = a_1, a_2, \dots, a_n$, a work title t_i , and publication year y_i . The goal of our approach is to find out the set $U = u_1, u_2, \dots, u_k$ of unique authors and attribute each citation record c_i to the corresponding author $u_j \in U$. Initially, the dataset is a set of citation records that have the same author name, each citation record will be put into a single cluster. The steps to achieve the goal are summarized as below:

- 1) Firstly, we perform pre-clustering for these records by using co-authorship with hierarchal agglomerative clustering method because co-authorship is a strong evidence to disambiguate records as used by many previous approaches. Our clustering criterion is any citation records in the same group must have at least two co-authors are the same as at least one of other records in the group. For other records that can not be grouped, we leave each of them into a single cluster.
- 2) We query records to the search engine ordered by the publication year from the dataset generated in step 1) because the later publications the higher possibility listed by author' home page or captured by search engine.
- 3) For the list of pages returned from search engine, we implement a model to identify if the page is a personal page for each of them. If a record can be

found in a personal page, we should also be able to find some other records in the same page and they should belong to one person.

- 4) After step 3), it might have some records do not belong to any groups because these records can not be found from any web pages. We implement a re-clustering model to handle these records and group them each other or with other groups if possible. In the output dataset, each cluster is expected to represent a distinct author.

From the above steps, we know the key components of our approach are the web pages identification in step 3) and the re-clustering model in step 4). The details about these two components are discussed in the following sections.

3.1 Web Pages Identification

According to our observation, there are two types of web pages returned from search engine if we use the author name plus the publication title as keyword, personal home pages and public pages like DBLP. Obviously, we are only interested in the former type of web pages because it is more sufficient to identify authors if multiple entries are found in a personal page.

Therefore, the first challenge is to identify which pages are personal pages. We implement a web pages identification model by using Neural Network(NN) based on the idea in [8]. Neural Network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use [5].

Since we only have two target categories, personal pages and non-personal pages, we decide to develop a single classifier with multiple target categories rather than a separate classifier for each target category because it requires less training. We use an open source package Neuroph¹ to implement the Neural Network and decide the network is multi-layer perceptron as our problem is not linearly separable. We use the back propagation to figure out how to adjust the weights of edges coming from the input layer. Due to the size limitation and key contribution of this paper, we are not going to discuss the details here. The technical details include feature selection and parameter optimization can be found in [8,9].

The NN model we propose above requires to read the page contents and then determined by the classifier. It will produce very high cost when there are millions of entries need to be identified and each page associated with the entry might contain megabyte size information. Therefore, we propose a traffic rank (TR) model to perform web pages identification without going inside each page in our earlier work [20].

Assume there are a list of citation records C that have the same author name, and given a list of web pages W associated to each record $c, c \in C$. For each page $w, w \in W$, we query them in Alexa². Alexa is a web information company which provides web site information like traffic rank and keywords for web pages.

¹ <http://neuroph.sourceforge.net/>

² <http://www.alexa.com/>

Because its web site information pages are very well organized, we can get the traffic rank and keywords for each page easily. We also set up a traffic rank range threshold β , which is generated according to the traffic rank of top 10 digital libraries, e.g. DBLP. If the page we query has the same or higher level of traffic rank to β , then our model takes this page as a public page, otherwise, the page will be treated a personal page.

However, some personal pages under the host of famous affiliations, e.g. Stanford University, have even higher traffic than public pages because Alexa calculates the traffic against the host. In order to solve this issue, we construct a taxonomy based on an existing education taxonomy³ which contains terms like "University" by using the method in [21] for those high traffic ranking pages. We propose a decision function $f(w) = \frac{N_t}{K}$ according to the taxonomy, where w is a web page, N_t is the number of keywords associated with w and can be found in the taxonomy, and K is the total number of the keywords of w . If $f(w) > \theta$ where θ is a threshold value, then this page is a personal page.

3.2 Mixed Model

As described in Section 3.1, there are some records can not be found in any web pages but in fact they might be able to group with other records. For example, some authors might only list those publications which published in famous conferences or journals in their homepages or some authors do not even have a homepage.

To solve this issue, we implement a mixed model that combined NN and TR models with topic information into a graph. The TR model will be used to quickly identify web pages in the first step so that smaller portion of records will be passed to NN model. The dataset has two parts after being processed and clustered according to co-authorship and the information in those personal pages identified by TR model and NN model. The first part is those citation records have been clustered, we called them $D1$. The second part is those records can not be found with any citation records in web pages and do not have enough same co-authors with others, we called them $D2$.

The records in $D1$ have at least two co-authors are the same as one of other records in the group or they are found in the same page which means they are being successfully grouped and each group represents a distinct author, we can use it to provide reference and group those citation records in $D2$ if possible .

We extend the SA-Cluster model introduced in the work [19] to be suitable for our requirements. There are several cases for the citation records in $D2$: 1) The record does not have co-authors. 2) There is one or more co-authors but does not have the same co-authors with any other records. 3) There is one or more co-authors and has the same co-author with at least one other records. For 1) and 2), since there is no co-authors relationship can be used, we assign a topic T for each citation record according to the taxonomy we built in [21] in order to establish linking among these records. We formalize the model as below:

³ <http://www.taxonomywarehouse.com/>

Assume there is a graph G , each vertex represents a citation record. If there are multiple paths connecting two vertices v_i and v_j , then they are closed which means these two records are likely belong to the same person. On the other hand, if there are very few or no paths between v_i and v_j , we use random walk distances to measure vertex closeness. Let P be the NN transition probability matrix of a graph G . Given l as the length that a random walk can go, $c \in (0, 1)$ as the restart probability, the neighborhood random walk distance $d(v_i, v_j)$ from v_i to v_j is defined as

$$d(v_i, v_j) = \sum_{\tau}^l p(\tau)c(1 - c)^{length(\tau)} \tag{1}$$

where τ is a path from v_i to v_j whose length is $length(\tau)$ with transition probability $p(\tau)$. The matrix form of the neighborhood random walk distance is

$$R^l = \sum_{r=1}^l c(1 - c)^r P^r \tag{2}$$

Here, P is the transition probability matrix for graph G , and R is the neighborhood random walk distance matrix, therefore the closeness between two vertices v_i and v_j is

$$d_S(v_i, v_j) = R^l(i, j) \tag{3}$$

As we mentioned before, each group in $D1$ represents a distinct author so that each citation record in $D2$ must reach a high threshold before it can be merged to any groups in $D1$ to keep the clustering quality. Assume there are N records in a group of $D1$, then we set up the threshold is λN where λ is the control parameter. The equation for the closeness of a record v_i in $D2$ to a group g_j in $D1$ is:

$$d_S(v_i, g_j) = \sum_{v_j \in g_j} d_S(v_i, v_j) \tag{4}$$

Then the total weight between v_i to g_j is:

$$Weight(v_i, g_j) = \omega_c \sum_{v_j \in g_j} d_c(v_i, v_j) + \omega_t \sum_{v_j \in g_j} d_t(v_i, v_j) \tag{5}$$

where ω_c is the weight for each path linked by co-authors and ω_t is the weight for each path linked by topics while $d_c(v_i, v_j)$ and $d_t(v_i, v_j)$ is the co-authors and topics closeness between v_i and v_j . If $Weight(v_i, g_j)$ is greater than λN then v_i can be grouped into g_j , and the $R^l(i, j)$ will be recalculated for the next iteration.

We note that one author might have more than one citation record in one personal page and have other different citation records in another personal page that represent as two distinct authors which can not be fully handled by our approach. However, this situation is very rare and does not exist in our test dataset and we do not consider this scenario in this paper. The implementation of this model are shown in Algorithm 1.

Algorithm 1. Clustering Algorithm

```

input : Citations  $C = c_1, c_2, \dots, c_n$ ;
         ResultSet  $RS = null$ ;
         Dataset  $D = null$ ;
         Dataset  $TempD = null$ ;
1 foreach citation  $c_i$  do
2   | Create new cluster  $cl_i$ ;
3   | Push into Dataset  $D$ ;
4 end
5  $PreClustering(D)$ ;
6 Pointer Start;
7 if  $isNotNull(TempD)$  then
8    $D = TempD$ ;
9   Sort  $D$  in descending order by publication year;
10 foreach cluster  $cl_i$  in  $D$  do
11   | foreach citation  $c_i$  in  $cl_i$  do
12     | Set  $Pages = QuerySearchEngine(c_i)$ ;
13     | foreach page  $p_i$  in  $Pages$  do
14       | if  $isPersonalPage(p_i)$  then
15         |  $DatasetD' = CheckAndMerge(cl_i, p_i, D)$ ;
16         | if  $isNotNull(D')$  then  $TempD = Split(D', D)$ ;
17         |  $RS.Add(D')$ ;
18         | Jump to Pointer Start;
19       | end
20     | end
21 end
22  $ReClustering(D, RS)$ ; output:  $RS$ 

```

Each citation has been put into a single cluster in this algorithm, and push into a dataset D . The $PreClustering$ function clusters the input dataset by using co-authorship since co-authorship is a strong evidence to disambiguate records. The rule is we group those citation records that have at least two co-authors are the same together and sort the dataset in descending order by publication year.

Then for each citation record, we query it from search engine, e.g. Google, using author name plus paper title as search string. If any page returned from the search results is a personal page, then the $CheckAndMerge$ function will check any other citation records from other clusters can be found in this page by using regular expression, and group those records into one cluster and return into dataset D' . If D' is not null, then add D' into resultset and rerun the process for the rest of citation records. Ideally, those records being clustered should be also included in the next iteration. However, according to our observation, one distinct author usually only has one homepage and for the consideration of performance, we do not include them to rerun the process. The process will repeat until no more clusters can be grouped.

The key function of this algorithm is $isPersonalPage$, we use the NN and TR models we discussed above to identify pages. These two models can be mixed

or separately used. Since the function is critical, we very strictly set up the threshold to make sure the identification is correct.

4 Evaluations

To evaluate our approach, we compare it against supervised learning and unsupervised clustering based methods. We perform evaluations on a dataset extracted from DBLP and used by Han et al. [4]. The dataset is composed of 8442 citation records, with 480 distinct authors, divided into 14 ambiguous groups, as shown in Table 1. We submitted queries using the Google Search API⁴ and collected the top 8 documents from each query due to the API restrictions.

Table 1. Evaluation Dataset

Name	Num. Authors	Num. Citations	Name	Num. Authors	Num. Citations
A. Gupta	26	577	J. Smith	31	927
A. Kumar	14	244	K. Tanaka	10	280
C. Chen	61	800	M. Brown	13	153
D. Johnson	15	368	M. Jones	13	259
J. Lee	100	1417	M. Miller	12	412
J. Martin	16	112	S. Lee	86	1458
J. Robinson	12	171	Y. Chen	71	1264

4.1 Evaluation Results

As the results shown at Table 2 for the 14 groups of ambiguous authors, we evaluated three models in our approach, Neural Network(NN) model, Traffic Rank(TR) model, and the mixed model which combines NN and TR models. For all citations in the dataset, we use the query keywords are quoted author name plus paper title.

In NN model, we set up the threshold for the output value is 0.6, which means if the final weight of a page produced by the neural network is greater than 0.6, then it is a personal page. In TR model, we set up the β value is 10 times higher than the average traffic rank of top 10 digital libraries. For those pages' traffic rank are lower than β which might be personal pages, we set the θ value is 0.001 and we double the β and θ value in the mixed model to make sure those personal pages will be picked up by TR model before processed further.

The results show that NN model can achieve better F1 value than TR model because each page will be passed to the classifier we implement so that the information of each page is validated while TR model only uses the traffic ranking and the keywords of the page to determine if it is a personal page. Therefore, some pages that are personal pages might be missed which have impacts on the results.

We also evaluate the accuracy of our strategy for identification of personal pages. We randomly select a sample of 100 pages that contain at least two citations and manually check if they are single author pages. 90 percent of these pages are correctly identified.

⁴ <http://code.google.com/apis/ajaxsearch/>

Table 2. F1 value of Our Models

Name	NN Model	TR Model	Mixed Model
A. Gupta	0.90	0.85	0.90
A. Kumar	0.91	0.91	0.91
C. Chen	0.78	0.58	0.82
D. Johnson	0.74	0.78	0.88
J. Lee	0.79	0.64	0.79
J. Martin	0.93	0.82	0.93
J. Robinson	0.96	0.92	0.96
J. Smith	0.80	0.76	0.85
K. Tanaka	0.94	0.68	0.94
M. Brown	0.82	0.84	0.88
M. Jones	0.90	0.79	0.92
M. Miller	0.94	0.82	0.94
S. Lee	0.93	0.85	0.94
Y. Chen	0.86	0.70	0.87
Mean	0.87	0.78	0.90

4.2 Comparison with Baseline Methods

Our methods was compared with three baseline methods: the WAD method uses rules to identify web pages [14], the k-way spectral clustering based method only uses citation metadata information in DBLP(KWAY) [4], and the web-based method using hierarchical agglomerative clustering (URL) which only uses URL information [16]. We reimplemented them based on the latest information retrieved from search engine.

Table 3 shows the comparison of F1 values between mixed model and the WAD model. In some cases of the dataset, both models have the same F1 value which means there are no difference between our web pages identification model and the rules in WAD model for these authors. However, there are huge improvement in some cases, "C. Chen" for example. The reasons of that is because our method not only analyze the URL information or the first 20 lines of the article but also go through the information in the page so that some pages are missed by the WAD are identified by our model.

We also compare the F1 value with other models proposed in [14]. Table 4 lists the average F1 value for all models. As we can see, KWAY method is the lowest since it only uses basic metadata information which is not sufficient to disambiguate authors. URL method is slightly better because information in the URL is involved and analyzed. WAD method is the best in the existing works and performs better than our TR model.

4.3 Performance Comparison

Since we consider the issue is a real-time application as described earlier, the disambiguation process should be operational in the real environment. As shown in Figure 1, we evaluate the disambiguation speed for all three models for different

number of records. We randomly pick up citation records from the test dataset. The TR model is the fastest because it only takes 12 seconds to disambiguate 100 records and less than 10 minutes for 8000 records. The Mixed model is the ideal one, it takes reasonable time to perform the process but the results is much better than TR model according to the results we evaluated before. The NN model is the slowest which takes nearly 30 minutes to run through 8000 records.

Table 3. Comparison of F1 values with WAD

Name	Mixed Model	WAD
A. Gupta	0.90	0.90
A. Kumar	0.91	0.91
C. Chen	0.82	0.64
D. Johnson	0.88	0.83
J. Lee	0.79	0.59
J. Martin	0.93	0.93
J. Robinson	0.96	0.96
J. Smith	0.85	0.78
K. Tanaka	0.94	0.94
M. Brown	0.88	0.88
M. Jones	0.92	0.92
M. Miller	0.94	0.90
S. Lee	0.94	0.85
Y. Chen	0.87	0.75
Mean	0.90	0.84

Table 4. Comparison of F1 values

Model Name	F1 Value
NN	0.90
TR	0.78
Mixed	0.90
KWAY	0.41
URL	0.60
WAD	0.84

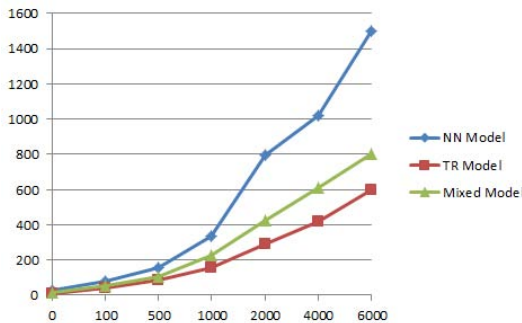


Fig. 1. Comparison of Disambiguation Speed (Seconds)

The testing is performed by a laptop which CPU speed is 1.83 GHz with 2GB memory, and the internet connection is Mobile 3G with theoretical speed is 54Mbps. Though the experiment shows the disambiguation speed is not really fast but considering most of authors have less than 500 records with more powerful machine and faster speed in reality, our solution can be implemented as a real world application with pages indexing if possible.

5 Conclusions

In this paper, we propose an approach for name disambiguation in digital libraries. We proposed a sequence of steps to submit queries to a search engine, identify pages and extract information from pages in the answer sets, create clusters of authors based on the extracted information and handle those records can not be found in any web pages. Our results indicate large gains in the quality of disambiguation when compared to other existing works. In addition, since the cost of our method is reasonably low, our approach can be possibly implemented as a real application in digital libraries as well as other related areas with name ambiguity issue.

References

1. Bilenko, M., Mooney, R.J., Cohen, W.W., Ravikumar, P., Fienberg, S.E.: Adaptive name matching in information integration. *IEEE. Inte. Sys.* 18, 16–23 (2003)
2. Dongwen, L., Byung-Won, O., Jaewoo, K., Sanghyun, P.: Effective and scalable solutions for mixed and split citation problems in digital libraries. In: *Proc of the 2nd Int. Workshop on Information Quality in Info. Sys.*, pp. 69–76 (2005)
3. Han, H., Giles, C.L., Hong, Y.Z.: Two supervised learning approaches for name disambiguation in author citations. In: *4th ACM/IEEE Joint Conf. on Digital Libraries*, pp. 296–305 (2004)
4. Han, H., Zhang, H., Giles, C.L.: Name disambiguation in author citations using a k-way spectral clustering method. In: *5th ACM/IEEE Joint Conf. on Digital Libraries*, pp. 334–343 (2005)
5. Haykin, S.: *Neural networks: A comprehensive foundation* (1999)
6. Kalashnikov, D.V., Mehrotra, S.: Domain-independent data cleaning via analysis of entity relationship graph. *ACM Trans. Database System* (2006)
7. Kang, I.S., Na, S.H., Lee, S., Jung, H., Kim, P., Sung, W.K., Lee, J.H.: On co-authorship for author disambiguation. In: *Information Processing and Management*, pp. 84–97 (2009)
8. Kennedy, A., Shepherd, M.: Automatic identification of home pages on the web. In: *Proc. of the 38th Annual Hawaii Int. Conf. on System Sciences*, pp. 99–108 (2005)
9. Koehler, H., Zhou, X., Sadiq, S., Shu, Y., Taylor, K.: Sampling dirty data for matching attributes. In: *SIGMOD*, pp. 63–74 (2010)
10. Lee, D., Kang, J., Mitra, P., Giles, C.L.: Are your citations clean? new scenarios and challenges in maintaining digital libraries. *Communication of the ACM*, 33–38 (2007)

11. Li, G., Zhou, X., Feng, J., Wang, J.: Progressive keyword search in relational databases. In: ICDE, pp. 1183–1186 (2009)
12. McCallum, A., Nigam, K., Ungar, L.H.: Efficient clustering of high-dimensional data sets with application to reference matching. *Knowledge Discovery and Data Mining*, 169–178 (2000)
13. Pasula, H., Marthi, B., Milch, B., Russell, S.J., Shpitser, I.: Identity uncertainty and citation matching. *Neur. Info. Proc. Sys.* 9, 1401–1408 (2002)
14. Pereira, D.A., Ribeiro, B.N., Ziviani, N., Alberto, H.F., Goncalves, A.M., Ferreira, A.A.: Using web information for author name disambiguation. In: 9th ACM/IEEE Joint Conf. on Digital Libraries, pp. 49–58 (2009)
15. Song, Y., Huang, J., Councill, I.G., Li, J., Giles, C.L.: Efficient topic-based unsupervised name disambiguation. In: In 7th ACM/IEEE Joint Conf. on Digital Libraries, pp. 342–352 (2007)
16. Tan, Y.F., Kan, M.Y., Lee, D.W.: Search engine driven author disambiguation. In: 6th ACM/IEEE Joint Conf. on Digital Libraries, pp. 314–315 (2006)
17. Yang, K.-H., Peng, H.-T., Jiang, J.-Y., Lee, H.-M., Ho, J.-M.: Author name disambiguation for citations using topic and web correlation. In: Christensen-Dalsgaard, B., Castelli, D., Ammitzbøll Jurik, B., Lippincott, J. (eds.) *ECDL 2008. LNCS*, vol. 5173, pp. 185–196. Springer, Heidelberg (2008)
18. Yin, X.X., Han, J.W.: Object distinction: Distinguishing objects with identical names. In: *IEEE 23rd Int. Conf. on Data Engineering*, pp. 1242–1246 (2007)
19. Zhou, Y., Cheng, H., Yu, J.X.: Graph clustering based on structural/attribute similarities. *Proc. VLDB Endow.*, 718–729 (2009)
20. Zhu, J., Fung, G., Zhou, X.: Efficient web pages identification for entity resolution. In: 19th International WWW, pp. 1223–1224 (2010)
21. Zhu, J., Fung, G.P.C., Zhou, X.F.: A term-based driven clustering approach for name disambiguation. In: Li, Q., Feng, L., Pei, J., Wang, S.X., Zhou, X., Zhu, Q.-M. (eds.) *APWeb/WAIM 2009. LNCS*, vol. 5446, pp. 320–331. Springer, Heidelberg (2009)

A Classification Framework for Disambiguating Web People Search Result Using Feedback

Ou Jin¹, Shenghua Bao², Zhong Su², and Yong Yu¹

¹ Shanghai Jiao Tong University, Shanghai, China, 200240
{kingohm,yyu}@apex.sjtu.edu.cn

² IBM China Research Lab, Beijing, China, 100094
{baoshhua,suzhong}@cn.ibm.com

Abstract. This paper is concerned with the problem of disambiguating Web people search result. Finding the information about people is one of the most common activities on the Web. However, the result of searching person names suffers a lot from the problem of ambiguity. In this paper, we propose a classification framework to solve this problem using an additional feedback page. Compared with the traditional solution which clusters the search result, our framework has lower computational complexity and better effect. We also developed two new features under the framework, which utilized the information beyond tokens. Experiments show that the performance can be improved greatly using the two features. Different classification methods are also compared for their effectiveness for the task.

1 Introduction

Finding the information about entities such as people is one of the most common activities of Web surfing. The queries containing person names have already taken up around 30% of search engine queries [1]. However, person names are highly ambiguous. 90,000 different names are shared by 100 million people according to the U.S. Census Bureau [2], and much more share the same abbreviation. Therefore, the search result of a person name is usually related to several different individuals. This is a great obstacle for browsing.

A traditional approach for disambiguating web people search result is clustering. Much work has been done along this way. But clustering has many limits, which make it inappropriate for practical use. First of all, its computational complexity is too high. Applications can only deal with a small part of the entire search result. Secondly, it is usually not easy to identify the number of clusters and the performance of clustering is still not satisfying. This deficiency in performance has a severe influence on browsing. If one page is clustered to a wrong cluster, finding it could cost a considerable amount of time. Thirdly, it does not naturally lead to a convenient browsing. Even after the search results are clustered by different individuals, the user still has to browse all of them to decide who each person is [2].

In this paper we introduce a different way to solve the problem. Particularly, we first introduce a classification framework for disambiguating Web people search result using an additional feedback page, and then comprehensively study different features and learning models within the framework. Our motivation is based on two observations. First, although people search results typically involve a great many ambiguous entities, a user usually cares about only one of them. Second, it is usually easy for a user to find one related page of the targeted entity. Figure 1 illustrates two typical cases of finding the relevant page of Alvin Cooper: a) user finds a relevant page by browsing the search results of “Alvin Cooper”. b) user feels interested in Alvin when the user is browsing a Web page of him. This feedback page can provide additional information about the person to be disambiguated. It is referred as the **given page** in the paper.



(a) Find the related page by browsing the search result



(b) Find the related page by surfing the Web

Fig. 1. Illustration of two typical cases of finding the relevant page

Given these two observations, it is natural to model the problem as a classification problem. Particularly, we use the given page as a positive instance, then classify all the other result pages into relevant and non-relevant classes. Although useful as it is, dynamically training a classifier for each query is not realistic since there is only one positive instance supplied and time is unaffordable. In our framework, we reformulate the problem as deciding whether a pair of pages are of the same person. Pairs of pages (a given page and a result page) are used as instances, and features are extracted by comparing the two pages using different similarity metrics. Therefore, adequate training instances can be obtained through annotation, and the classifier can be trained offline. It is also guaranteed that test instances are in the same feature space as training instances. Given any query and feedback page, we can classify all the result pages using the classifier. Compared with the traditional clustering method, the time cost is dramatically reduced, since the complexity of extracting features is only linear to the number of pages.

Previous features developed for disambiguating Web people search result in clustering method such as token based features and NLP based features can still be used in the classification framework. But these features are all obtained by only shallow processing of texts. In this paper we are to explore more useful features obtained through deep analysis. Particularly, we propose two new features: *key token* and *topic*. Key tokens are defined as the most discriminative tokens of an individual. They tend to co-occur a lot with the individual they relate to. Finding these key tokens can help disambiguate Web pages, since irrelevant contents can thus be ignored. Topics are the subjects the Web pages

talk about. Experiments show that using the two features can greatly improve the performance of disambiguation.

Another challenge in the framework is that, it can not be guaranteed that the test set obeys the same distribution as the training set. Since the number of related pages in Web people search result is usually small compared to the sum, it leads to imbalance problem. When confronted with imbalance training data, traditional classifiers tend to be overwhelmed by large classes and ignore small classes. Due to these problems, it is very hard to obtain a robust classifier. In this paper we compared some existing methods for imbalance corpus on this problem.

The rest of the paper is structured as follows. Section 2 surveys the related work. In Section 3 we formulate our problem and introduce the classification framework. In Section 4 we study different features, and propose two new features for the classification framework. In Section 5 we study different classifiers and methods for overcoming imbalance data problem. Section 6 presents the experimental results. Finally, we make some concluding remarks and discuss the future work in Section 7.

2 Related Work

While much work has been done on both name disambiguation and relevance feedback, to the best of our knowledge, it is the first try on disambiguating the Web people search result using relevance feedback within a classification framework. In this section, we introduce some most related work on name disambiguation, relevance back.

2.1 Name Disambiguation

There are mainly two kinds of previous work on person name disambiguation. Some deal with the ambiguities of the citations [7, 10]. [10] used a two-step framework to disambiguate names in citations on large scale. They first reduced the number of ambiguous candidates via blocking, then measured the distance of two names via coauthor information. [7] used the K-way spectral clustering method to cluster person names. They made use of the relations between people as additional information to perform disambiguation. Others try to disambiguate Web people search result [6, 9]. [9] automatically extracted patterns of biographic facts such as birth day via a bootstrapping process, and clustered the documents with the extracted information. [6] also applied the unsupervised learning method, and enriched the features by exploring syntactic and semantic features. Recently, disambiguating Web people search result is also studied intensively at Web People Search task (WePS) [1] and Spock Entity Resolution Challenge [4].

Person name disambiguation is similar to our task. But previous work is all based on unsupervised learning. In this paper we focus on disambiguating Web people search result, and model it in a classification framework. To the best of our knowledge, this is a novel study.

¹ <http://challenge.spock.com/>

2.2 Relevance Feedback

Relevance feedback has been studied for a long time. The main idea is to utilize the user's feedback pages which may be considered as relevant to the user's information need to improve the search result.

Rocchio's method [12] is a classic algorithm for using relevance feedback. It proposed a model to incorporate relevance feedback into vector space model. [4] applied modified Rocchio formula to incorporate relevance feedback on TREC. Later Buckley proposed a dynamic feedback optimization method for optimizing query weights in [3]. Recently, the research of relevance feedback grew many branches. [11] used click through data as implicit feedback to learn ranking functions for Web search result. [13] deal with the negative relevance feedback problem, in which no positive feedback was provided.

Our work is quite different from these work. First, the feedback page is an indispensable part of our framework. It provides exclusive information for the entity to be disambiguated, since the query (person name) provides no information for disambiguation. Second, Our problem is a classification problem, instead of a ranking problem. We need to identify those pages describing the same entity as the given page describes.

3 The Classification Framework

Web people search result is usually highly ambiguous. Pages in the search result could relate to different individuals. To satisfy the user's information need, it is necessary to find out pages related to the person the user really cares about. We summarize the problem as follows.

Formally, given a person name as a query, a feedback page the user selected as the given page, we wish to automatically identify the pages describing the same individual as described in the given page. To approach this problem, we model it in a classification framework. The framework is shown in Figure 2.

The framework works as follows. For training, given a pair of a person name and a given page, each page in the search result of the person name is annotated as related or not related to the given page. Related means they are describing the same person. If it is related, then it is a positive instance, and vice versa. After adequate training instances are gathered, we use them to train a classifier. In the process of training, features are formed by measuring the similarities between the instance page and its given page. A comprehensive study of types of similarities used for features in the classification framework will be given in the next section. For testing, each page in the search result is firstly represented using features extracted by comparing with the given page, then classified using the trained classifier. The classification framework is compatible to traditional methods in two ways: 1) The features developed for clustering in previous work for disambiguation can still be used here. 2) Any classification method can be used in the framework.

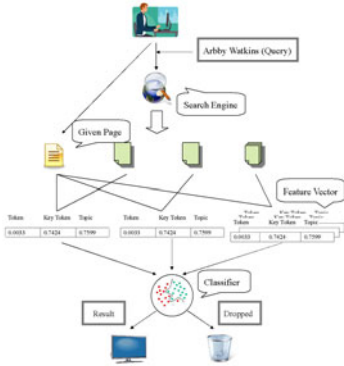


Fig. 2. Our classification framework for disambiguating Web people search result



Fig. 3. A screenshot of our prototype system

4 Features

Many features have been studied in the setting of disambiguation. As we mentioned, they can still be used in our classification framework. They mainly use four types of information: basic tokens, links between pages, noun phrases and name entities, sentences where the target person name appears. These features have been discussed a lot in previous work. But most of them are only obtained by shallow processing of texts. In this section, we propose two novel features, which are obtained through deep analysis of page contents.

Pages describing the same entity are usually about different events. Thus most of their contents could be different. It should be noticed that the key tokens are not necessarily the tokens with highest TF, IDF or TF*IDF. Meanwhile, sometimes, different pages describing the same person use very different wording. If we only use simple features such as tokens or named entities, they would have very small similarity. To conquer this problem, we make use of the topic information of Web pages. Particularly, we use the ODP to automatically decide the topic of Web pages.

4.1 Key Tokens

Key tokens are defined as the most discriminative words for individuals. They usually accompany a person’s appearances in Web pages. Thus they are naturally good clues for disambiguation. However, finding these key tokens is not a easy work since web pages contain much noise. Traditional methods use TF*IDF to weight terms to reduce the influence of the noise. Unfortunately, key tokens are not necessarily the terms with highest weights. Let us take a look at an example. Consider three kinds of pages involving a psychologist: his homepage declaring himself as an expert on psychology on the internet; pages of on-line book stores with his book name “Infidelity on the Internet: Virtual Relationships and Real Betrayal”; pages quoting a sentence from his work: “The Internet

is attracting people with a variety of fetishes to chat rooms. And, those chat rooms are attracting sexual predators”. The only word the three have in common beside his name is “internet”, which has a low TF weight in any of these pages. Considering another situation that all the pages belonging to a person relate to computer science. Thus all the pages contain the word “computer”, which appears so frequently that sometimes it is even considered as a stopword. The IDF weighting does not work well, either.

In practice we utilize the co-occurrence information of words to determine whether they are key tokens. Intuitively, words co-occurring in many pages tend to be descriptions for the same person. While words occurring singly are usually noises for disambiguation, which should be ignored. Below we explain in detail how the feature works.

Let us consider a matrix M . One dimension of M is the Web pages in the search result: P_1, P_2, \dots, P_n , the other is the tokens in these pages: T_1, T_2, \dots, T_m . For traditional token similarity, the matrix is filled by TF*IDF and cosine similarity is calculated between pages. To extract key tokens, we first use the matrix M to cluster the tokens. By doing so, we can discover groups of tokens that appear in patterns. Then by removing the single tokens that can not be clustered, we could filter out those noisy information. What are left are the key tokens. When using key tokens, we treat a cluster of tokens as a unit. That is, a new matrix M' is formed, with one dimension still being the pages in the search result P_1, P_2, \dots, P_n and the other the token clusters C_1, C_2, \dots, C_s . Similarity is then calculated on the new matrix.

In practice, to reduce noise and computation cost, we do not use all tokens in the search result. Instead, we use only tokens in local sentences[6], which are the nearest sentences to the appearances of person names. Beside TF*IDF, we also use different representation functions F_{token} to fill out the matrix. Candidate functions are listed below.

$$\text{Boolean}(P_i, T_j) = \begin{cases} 1, & \text{if } P_i \text{ contains } T_j \\ 0, & \text{otherwise} \end{cases}$$

$$\text{TF}(P_i, T_j) = \text{the appearance times of } T_j \text{ in } P_i$$

$$\text{TF-IDF}(P_i, T_j) = \text{TF} * \log \frac{n}{DF}$$

Then the similarity between two tokens can be calculated by cosine similarity function.

$$\text{Sim}(T_i, T_j) = \frac{\sum_k M[i][k] \cdot M[j][k]}{|M[i]| \cdot |M[j]|}$$

Given a threshold θ_{token} , we apply agglomerative clustering[8] to cluster the tokens. Then we choose another function F_{page} to fill out the new matrix M' . Finally, features between two pages are extracted by calculating the similarity on M' . The influence of threshold θ_{token} and different representation functions will be demonstrated in the experiment section.

4.2 Topics

By topics we refer to the subject of Web pages. Sometimes documents telling the same story use very different wording. Finding the topics of them could be very helpful to measure the similarity between the result pages and the given page. In this paper we use the categories defined by experts in ODP to help us to decide topics of pages. ODP (Open Directory Project) is aimed to build the largest human edited directory of the Web. It manually organize a huge amount of Web pages in a hierarchical category tree. We use the categories in ODP to train a classifier, and classify all result pages as well as the given page according to these predefined categories.

Classification using ODP categories has been studied a lot previously. Since in our work efficiency is a primary concern, we only make use a simple classification method to accelerate the processing speed. Particularly, we first filter out two miscellaneous top categories: World and Adult. Then we select a depth D . Categories deeper than D will be folded into their parent categories. Then we combine all the titles and descriptions of documents in a category into a single token vector. By calculating the cosine similarities between them and the vectors of Web pages, we can get similarities between Web pages and categories in ODP.

After getting the similarities, there are many ways of utilizing them. Below are two extremes:

Vector. Consider all the similarities as a vector. Compute the topic similarity between Web pages by calculating the cosine similarities between the vectors.

Best. Assign the category of highest similarity to the page. Compute the topic similarity by simply comparing the category they belong to.

The first method usually brings much noise since many irrelevant topics are involved. While the second may lose a lot of important information since a page can be related to many categories. Therefore, we propose the third way to conquer the problem.

Top. Filter out the categories whose similarity is less than a confidence threshold θ_c . If one page has more than θ_n topic, the pages with small similarity values will also be filtered out. Then the similarity between two pages is calculated by the cosine similarity between the two filtered topic vectors.

5 Learning Methods

As we mentioned, any classification method could be used in our classification framework. The traditional classifier always assume that the distributions of training set and testing set are the same. But in our environment, the situation is much different. These could affect the disambiguation quality as well as its robustness. In this section, we compare different learning methods within the classification framework.

We compared the Naive Bayes(NB) and Support Vector Machine(SVM). The experiment shows that NB is quite robust over all three data sets. But it does not achieve a satisfactory performance, since it is based on strong independence assumption. Although SVM has better performance on some data sets, it is not as robust as NB. This is mainly because of the imbalance data problem. To conquer the problem we tried Over sampling and Under sampling. One randomly add instances from minor class to make it as large as the major class, and the other randomly eliminate instances from major class to make it as small as minor class.

6 Experiments

In this section, we empirically evaluate on our proposed methods. We will first introduce our experimental setup, and then discuss the experiment results.

6.1 Experiment Setup

Based on the framework, we implemented a prototype system. It can disambiguate the search result given a feedback click. Figure 3 shows the result of our system when the button of “find more” in Figure 1 is clicked.

We use the data from WePS 2007 [1] for our experiments. The data set consists of a training set and a test set. The training set contains 49 queries and the test set contains 30 queries. The queries are generated from three different ways: 1) a list of ambiguous person names in English Wikipedia; 2) program committee list of ECDL 2006 and ACL 2006; 3) random combinations of the most frequent first and last names in the US Census.

The statistics of the corpus are shown in Table 1. We denote the number of names, the number of pages and the number of entities as N , E and P , respectively. It shows that the test set has more ambiguity, since more entities belong to one name(E/N). There are much difference between training and test set, although the queries are generated in the same way. We refer to these data sets as data 1, 2 and 3 according the way they generated.

Then our data set is generated by selecting each page as a given page, and separate other pages as positive or negative according to the original annotation. The ratio of negative instances to positive instances is shown in Table 2. We can

Table 1. Analysis of data set

	Data	N	P/N	E/N	P/E
Train Set	1.Wikipedia	7	90	21.1	4.3
	2.ECDL	10	68	14.3	4.8
	3.US Census	32	27	6.6	4.2
Test Set	1.Wikipedia	10	100	43.9	2.3
	2.ACL	10	85	28.9	3.0
	3.US Census	10	84	44.8	1.9

Table 2. The ratio of negative to positive in train and test set

Data	Training Set	Test Set
1	1.66	7.89
2	0.82	1.81
3	0.40	9.70

Table 3. Comparison on the performances of different features

	Precision	Recall	F-Measure
Token	0.796	0.268	0.401
+LocalToken	0.861	0.263	0.403
+NameEntity	0.763	0.295	0.426
+Key Token	0.786	0.332	0.467
+Topic	0.856	0.608	0.711

see except in data 2, the ratio is quite different in training set and test set. Since the traditional classifier assumes that the distribution in training set and the test is same, they can not perform well on data 1 and 3. Therefore, to evaluate different features, we only experiment on data 2 with NB method. Data 1 and 3 are experimented on for the robustness problem in the subsection which evaluates different learning methods.

The evaluation metrics we use are precision(P), recall(R) and F-measure ($F_{0.5}$, $F_{0.5} = \frac{2PR}{P+R}$).

6.2 Experiment on Features

As mentioned, we use NB classifier and data 2 to test the effectiveness of different features. Our baseline is to use only the token feature for classification. For comparison, we separately add in other features and test the improvement they brought to the performance of classification.

We first compare the performance of using different features based on the baseline system. Beside the features proposed in this paper, we also include features proved to be effective in previous work: local token and local named entity. They are defined as the tokens and named entities that appear in the sentences which have the target person name, respectively. The comparison is shown in Table 3. According to our experiment, although traditional features (local token and named entity) bring improvement to the baseline, the improvement is not as big as the two proposed features bring. Below we discuss the performance of using key token and topic in detail.

Key Token. Recall the process for producing key tokens in Section 4. To reduce the noise and the computation cost, we only consider the tokens in local sentence. We first cluster the tokens according to their appearances in pages, then use the token clusters to measure the key token similarity between pages. There are two kinds of functions and a parameter involved. F_{token} is used to fill out the initial token-page matrix M to cluster tokens. θ_{token} is used to control the clustering process. F_{page} is used to fill out the new token cluster-page matrix M' for measuring the key token similarity between pages.

The comparison of different choices of F_{token} and F_{page} is shown in Figure 4. The influence of θ_{token} is also demonstrated. As we can see, by choosing the appropriate functions, the key token feature can greatly improve the classification performance. Also we can see that Boolean Model is the best for F_{token} and for

F_{page} . TF function does not work here since key tokens themselves are already highly discriminative which do not necessarily appear many times in Web pages. IDF is not a good function either since key tokens are usually the ones that appear regularly and thus probably do not have a high IDF value. It is also showed that the key token feature reaches its best performance when threshold θ_{token} is around 0.9. This is because we focus on finding the key tokens which tend to appear in a clear pattern. Small threshold will bring too many unrelated token clusters. Too large threshold can not work well, either, since very few tokens can be clustered.

Topic. The ODP data we used in the experiment is downloaded from the Web site of DMOZ². It includes the hierarchy of the categories and the information of the pages belonging to the categories. Instead of having the whole pages, it only includes the titles and abstractions of them. Here we combine the titles and abstractions and extract token vectors from them. Threshold D is used to control the depth of categories involved. The categories deeper than D will be merged into its father category. After we get the similarities between pages and topics, we experiment on the three methods of topic selection described in the Section 4.2

The comparison of the three methods using different D is shown in the Figure 5. We can see that *Top method* performs much better than the others. The reason is that the *Vector method* is usually noisy while the *Best method* loses too much information. Most of the methods perform badly when number of topics grows large, since the topics are too detailed and therefore too difficult to match.

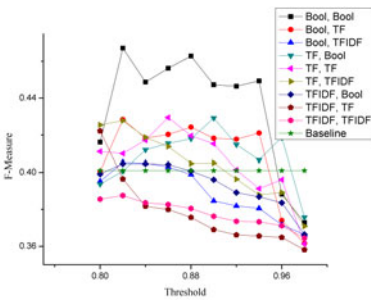


Fig. 4. Comparison on the performances of different combinations of F_{token} and F_{page}

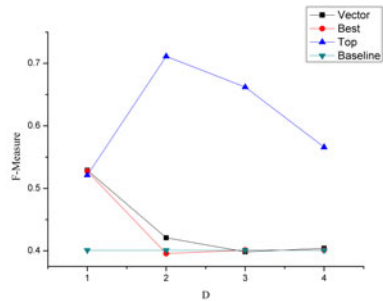


Fig. 5. Comparison on the performances of different methods of using topic feature

6.3 Experiment on Learning Methods

In this subsection, we test on different classification methods to reach a robust solution. To maximize the performance, we make use of all the proposed features, namely token, key token and topic.

² <http://rdf.dmoz.org/rdf/content.rdf.u8.gz>

Table 4. Comparison on the performances of NB and SVM on three data sets

	Data	Precision	Recall	F-Measure
NB	1	0.466	0.498	0.482
	2	0.782	0.348	0.482
	3	0.419	0.544	0.474
SVM	1	0.226	0.557	0.321
	2	0.784	0.668	0.721
	3	0.093	1	0.171

Table 5. Performance of SVM with sampling method

	Data	Precision	Recall	F-Measure
Over Sampling	1	0.235	0.595	0.337
	2	0.786	0.667	0.722
	3	0.210	0.607	0.312
Under Sampling	1	0.235	0.595	0.337
	2	0.786	0.667	0.722
	3	0.210	0.607	0.312

Firstly, we compare the performance of NB and SVM on three data set. The result is shown in Table 4. We can see that, although SVM performs much better on data 2, it does not do well in data 1 and 3 since the distributions of positives and negatives are different in training and test set. Especially it fails completely on data 3 since it is biased and tend to classify all instances as positive. Although NB appears much more robust, it does not achieve a satisfying result.

To solve the imbalance data problem exhibited in data 1 and 3, we employed two sampling methods to help SVM: under sampling and over sampling. They are both widely used for imbalance learning problems [5]. Their objectives are to make the ratios of negatives to positives equal in training and test sets. But since we do not know any information about the test data, we can only sample the ratio as 1:1.

The result of using the two sampling methods to help SVM is shown in Table 5. Although they can help make SVM more robust, the improvement on performance is limited. This is probably because that, even though we adjusted the ratio of negatives to positives in the training set, it is still not guaranteed to be the same as that in the test set.

7 Conclusion and Future Work

In this paper, we proposed a classification framework for disambiguating Web people search result with feedback. Compared with traditional clustering methods, the classification framework is more scalable and practical. The framework is also compatible to previous work since all the features developed for clustering methods can still be used.

Under the framework we proposed two new features to assist classification: key token and topic. Experiments demonstrated that they can increase the performance of disambiguation greatly. We also studied different classification methods under the framework. NB is much robust than SVM, but SVM performs much better on one data set. Sampling methods can help a little, but is limited.

As for future work, we will discover more sophisticated features for disambiguating Web people search result. Besides, we are also to apply the proposed framework for disambiguating more entities, such as companies, locations, and products.

References

1. Artiles, J., Gonzalo, J., Verdejo, F.: A testbed for people searching strategies in the www. In: In Proceedings of the 28th Annual International ACM SIGIR Conference, SIGIR 2005 (2005)
2. Artiles, J., Sekine, S., Gonzalo, J.: Web people search: results of the first evaluation and the plan for the second. In: WWW 2008: Proceeding of the 17th International Conference on World Wide Web, pp. 1071–1072. ACM, New York (2008)
3. Buckley, C., Salton, G.: Optimization of relevance feedback weights. In: Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 351–357. ACM Press, New York (1995)
4. Buckley, C., Salton, G., Allan, J.: The effect of adding relevance information in a relevance feedback environment. In: Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, SIGIR 1994 (1994)
5. Chawla, N.V., Japkowicz, N., Kolcz, A.: Editorial: Special issue on learning from imbalanced data. SIGKDD Explorations 6, 1–6 (2004)
6. Chen, Y., Martin, J.: Towards robust unsupervised personal name disambiguation. In: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (CoNLL 2007), pp. 190–198 (2007)
7. Han, H., Zha, H., Giles, C.L.: Name disambiguation in author citations using a k-way spectral clustering method. In: Proceedings of Joint Conference on Digital Libraries (JCDL 2005) (2005)
8. Kurita, T.: An efficient agglomerative clustering algorithm using a heap. Pattern Recognition 24(3), 205–209 (1991)
9. Mann, G., Yarowsky, D.: Unsupervised personal name disambiguation. In: Proceeding of the 2003 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, CoNLL 2003 (2003)
10. On, B.W., Lee, D., Kang, J., Mitra, P.: Comparative study of name disambiguation problem using a scalable blocking-based framework. In: Joint Conference on Digital Libraries, JCDL 2005 (2005)
11. Radlinski, F., Joachims, T.: Query chains: Learning to rank from implicit feedback. In: The Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2005 (2005)
12. Rocchio, J.J.: Relevance feedback in information retrieval. In: The SMART Retrieval System: Experiments in Automatic Document Processing, pp. 313–323 (1971)
13. Wang, X., Fang, H., Zhai, C.: A study of methods for negative relevance feedback. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 219–226. ACM, Singapore (2008)

Incorporating User Feedback into Name Disambiguation of Scientific Cooperation Network

Yuhua Li*, Aiming Wen, Quan Lin, Ruixuan Li, and Zhengding Lu

Intelligent and Distributed Computing Lab,
School of Computer Science and Technology,
Huazhong University of Science and Technology,
Wuhan 430074, P.R. China
wenaiming@smail.hust.edu.cn, linquan.hust@gmail.com,
{idcliyuhua,rxli,zdlu}@hust.edu.cn

Abstract. In scientific cooperation network, ambiguous author names may occur due to the existence of multiple authors with the same name. Users of these networks usually want to know the exact author of a paper, whereas we do not have any unique identifier to distinguish them. In this paper, we focus ourselves on such problem, we propose a new method that incorporates user feedback into the model for name disambiguation of scientific cooperation network. Perceptron is used as the classifier. Two features and a constraint drawn from user feedback are incorporated into the perceptron to enhance the performance of name disambiguation. Specifically, we construct user feedback as a training stream, and refine the perceptron continuously. Experimental results show that the proposed algorithm can learn continuously and significantly outperforms the previous methods without introducing user interactions.

Keywords: Name Disambiguation, User Feedback, Scientific Cooperation Network, Perceptron, Constraint.

1 Introduction

Name ambiguity is widely existing in academic digital libraries, such as Springer, ACM, DBLP and CiteSeer. For different authors may be thought of as the same author, name ambiguity makes data robust even dirty and lowers the precision of researches based on it.

Name disambiguation is a very critical problem in multiple applications, and it is often desirable to be able to disambiguate author names for many reasons. First, while browsing or searching academic papers, users would be interested to find papers written by a particular author. Second, the resulting disambiguated names can be used to improve other applications such as homepage search and calculating the academic ability of scientists.

* Correspondence author.

Though lots of work has been involved in name disambiguation, the problem has still not been well settled. Furthermore, User feedback which has been widely studied in many other applications and has good performance, is largely ignored in most researches about name disambiguation. To fill this gap, we leverage user feedback to improve the performance of name disambiguation.

This paper focuses on the problem of assigning papers to the right author with the same name. We investigate into name disambiguation of the scientific cooperation network. The problem is formalized as follows: given a list of papers with authors sharing one name but might actually referring to different people, the task then is to assign the papers to different classifications, each of which contains only papers written by the same person. This paper has three main contributions:

- A approach that incorporates user feedback into the perceptron is proposed to handle the name disambiguation of scientific cooperation network. Two new features and a constraint are drawn from user feedback to help achieve a better disambiguation result.
- User feedback is constructed as training stream to train the perceptron, therefore, perceptron can be refined continuously.
- We conducted the experiment using our approach in a real-world dataset, and the experimental results have proved that our proposal is an effective approach to solve the problem of name ambiguity.

The rest of the paper is organized as follows. In Section 2, we review related works. Section 3 gives name disambiguation a formal definition. In Section 4, we introduce the main idea of name disambiguation using constraint-based perceptron and explain how to incorporate user feedback into the model. Section 5 discusses experimental results, while Section 6 gives the conclusion and future work of this paper.

2 Related Works

In this section, we briefly introduce previous works, which fall into two major aspects: name disambiguation and machine learning methods joining user feedback.

Name Disambiguation: A great deal of research has focused on the name disambiguation in different types of data. [1] addressed the problem of named entity disambiguation in Wikipedia. A heuristic approach in [2] is proposed to author name disambiguation in bibliometrics.

[3] proposed a hybrid disambiguation method, which exploits the strength of both unsupervised and supervised methods. [4] described an algorithm for pairwise disambiguation of author names based on a machine learning classification algorithm, random forests.

Two supervised methods in [5] used supervised learning approaches to disambiguate authors in citations. An unsupervised method in [6] is proposed for name disambiguation in bibliographic citations. Besides, [7] proposed an unsupervised learning approach using the K-way spectral clustering method, they calculate a Gram matrix for each person name and apply K way spectral clustering algorithm to the Gram matrix.

[8] proposed two kinds of correlations between citations, namely, topic correlation and web correlation, to exploit relationships between citations. [9] concentrated on investigating the effect of co-authorship information on the resolution of homonymous author names in bibliographic data. [10] explored the semantic association of name entities and cluster name entities according to their associations, the name entities in the same group are considered as the same entity.

[11] presented a constraint-based probabilistic model for semi-supervised name disambiguation, they formalize name disambiguation in a constraint-based probabilistic framework using Hidden Markov Random Fields. [12] proposed a constraint-based topic model that uses predefined constraints to help find a better topic distribution, and in turn, achieve a better result in the task of name disambiguation.

Machine learning methods joining user feedback: Traditionally, machine learning systems have been designed and implemented off-line by experts. Recently however, it has become feasible to allow the systems to continue to adapt to end users by learning from their feedback.

Clicking data is a kind of invisible user feedback information [13]. The machine learning performance can be improved by employing user's clicking data, and it is widely applied to information retrieval and advertising fields.

[14] proposed a machine learning method to understand the keywords submitted by users. [15] analyzed the user feedback reliability in the Internet. [16] proposed a real-time keywords recommendation algorithm. Matthew Richardson [17] put up a method to predict whether a user will click a ads through analyzing user feedback of clicking ads. [18] proposed a solution for users to directly provide feedback and incorporating the feedback into information extraction.

Most of these algorithms access the user feedback information implicitly. User feedback obtained by this way may have lots of noises and the useful information always hide deeply. With the prevailing of Web2.0 applications, interactive design enriches the forms of user feedback. New forms of user feedback contain more abundant information, and also more accurate. To the best of our knowledge, no previous work has directly involved in incorporating user feedback into name disambiguation.

As just mentioned, multiple methods have been raised up to solve name disambiguation, but they all face the problem of low accuracy. Besides, there inevitably exist some mistakes in the result, however, those methods can not correct themselves. In this paper, we employ user feedback to name distinguishing in scientific cooperation network, which can revise the result constantly.

3 Problem Formulation

3.1 Problem Definition

In scientific cooperation network, we here give a formal definition of the name disambiguation problem. As our previous work [19], to formally describe the problem, we define the scientists sharing the same name a as a collection $A = \{a_1, a_2, \dots, a_i\}$, and we can get a publication set sharing the same author name a and denote it as $P = \{p_1, p_2, \dots, p_n\}$. Our task is to find the real author of these academic papers and tell them apart, that is, partition the academic paper collection P into small collections $\tilde{P}_1, \tilde{P}_2, \dots, \tilde{P}_i$, make sure each collection \tilde{P}_i only contains papers written by one scientist, as shown in Figure 1.

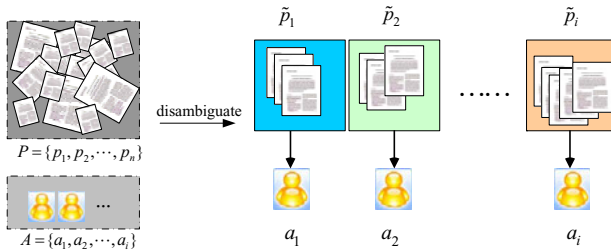


Fig. 1. Name Disambiguation in Scientific Cooperation Network

3.2 User Feedback

The disambiguation result inevitably exists some mistakes which can be easily found out by the users, for example, if one of the target paper’s authors sees the disambiguation result about his name, he can easily find out the mistakes.

The original feedback collected from users inevitably contains some mistakes. If the noises of user feedback are not filtered, the performance of name disambiguation will be affected. We divide user feedback into three kinds according to the users providing the feedback.

1. Fully credible user feedback. The feedback is provided by one of the target paper’s authors or one of the the target paper’s co-authors. As authors or co-authors of the target paper are very familiar with the papers, so their feedback is credible.

2. Credible user feedback. The feedback is provided by the friends of the ambiguous authors. The ambiguous authors’ friends can be considered persons who have co-worked with the ambiguous authors. These users are not directly involved in the creation of the target paper, so they are not very sure about who are the authors, there might exist certain misjudgments.

3. Generally credible user feedback. The feedback is provided by users without explicit relationship with the authors. Their feedback is unsure.

3.3 Feedback Training Stream

The biggest challenge of the machine learning based name disambiguation algorithms is the construction of training set. The training sets of the traditional machine learning algorithms are generally static, that is, learning all the training sets at one time. The user feedback arrives continuously, therefore, the training set constructed by user feedback should be provided to the preceptron as a stream for real-time learning.

The stream of training set constructed by user feedback is showed as (1).

$$feedback_training_stream = [(p_{i1}, p_{j1}, v_n, c_n), (p_{i2}, p_{j2}, v_n, c_n), \dots, (p_{in}, p_{jn}, v_n, c_n)] \quad (1)$$

It is a sequence constructed by user feedback according to the time stamp $\{1 \dots n\}$. Where the value of v_n is 1 or 0, denotes user feedback considers the two papers are written by one author or different authors respectively, and c_n denotes the reliability of user feedback, if it is fully credible, the value is 1, otherwise if it is credible, the value is 0, and the value is -1 when it is generally credible.

3.4 Feature Definition

A variety of information can be utilized for name disambiguation [19]. First, we define a set of features to be exploited for each paper pair (p_i, p_j) as

$$R = \{r_1, r_2, \dots, r_k\} \quad (2)$$

where each r_i denotes one feature capturing one relationship between papers p_i and p_j , as shown in Table 1. All the features are defined over the papers sharing the same primary author. For each feature in Table 1, the feature value is binary, that is, if the description is true, then the value is 1; otherwise 0. If the feature value is 1, then the feature indicates that the two papers are probably written by the same author. More detailed descriptions about the features in Table 1 can refer to [19].

Besides the above seven features, two new features are drawn from user feedback. According to the difference of users' credibility, we extract two features from credible user feedback and non-credible user feedback respectively.

Table 1. Feature definition for paper pair (p_i, p_j)

R Feature	Description
r_1 Co-Author	exist $u, v > 0, a_i^{(u)} = a_j^{(v)}$
r_2 Co-Org	$a_i.organization = a_j.organization$
r_3 Citation	p_i and p_j has citation links
r_4 Title-Similarity	$a_i.title$ and $a_j.title$ are similar
r_5 Homepage	p_i and p_j appear in someone's homepage
r_6 Digital-Lib	p_i and p_j appear in the same Springer/CiteSeer page
r_7 PDF File	$a_i.organization$ appears on the PDF format file for p_j and vice-versa

Suppose credible users have returned some user feedback about the paper pair $\{p_i, p_j\}$, of which m users consider the two papers are written by the same target author, while n users consider them belonged to different authors, then feature extracted from such feedback is donated as credible user feedback feature, as shown in the formula (3).

$$r_8 = \begin{cases} \frac{m}{m+n} & \text{if } m + n \neq 0 \\ 0.5 & \text{else} \end{cases} \quad (3)$$

When no one has submitted user feedback, the default value is 0.5.

For the non-credible users, though there contain lots of mistakes, however, when the quantity is larger enough, the correct feedback will dominate, that is, the main opinion of the non-credible users is right. Correspondingly, we can define a non-credible user feedback feature r_9 , the formula is the same as (3), but m donates there are m non-credible users considering p_i and p_j written by the same author, and n donates n non-credible users consider the two papers written by different authors. The default value is 0.5 when no one has submitted user feedback.

4 Incorporating User Feedback into Constraint-Based Perceptron

By analyzing the seven features showed in Table 1, we figure out homepage is different from the other features that if its value is 1, we can confirm the two papers are written by the same author, while if the feature value of homepage is -1, we can conclude the two papers are not written by one author. However, we can not have this conclusion with other features. Therefore, we designate homepage as a constraint for the perceptron.

Feedback provided by fully credible users has very high accuracy. Therefore, it can be used as a constraint as the homepage feature, to revise the output of perceptron. The constraint extracted from fully credible user feedback is donated as user feedback constraint, the constraint formula is as (4).

$$Constraint_{feedback}(p_i, p_j) = \begin{cases} 1 & p_i \text{ and } p_j \text{ belong to one author} \\ -1 & p_i \text{ and } p_j \text{ belong to different authors} \end{cases} \quad (4)$$

The user feedback constraint may be conflicted with the homepage constraint, though the probability is very low. When it happens, we give the priority to the user feedback constraint.

Perceptron with constraint aims to restrict the output. The final output is calculated using the formula (5).

$$output(p_i, p_j) = \begin{cases} 1 & c(y) = 1 \text{ or } c(y) = 0 \text{ and } Sgn(y) = 1 \\ 0 & c(y) = -1 \text{ or } c(y) = 0 \text{ and } Sgn(y) = 0 \end{cases} \quad (5)$$

Where $c(y)$ denotes the constraint, $c(y) = R_0(p_i, p_j)$, $Sgn(y)$ is the output of perceptron and it is defined as formula (6).

$$Sgn(y) = \begin{cases} 1 & \text{if } \omega * y + b > 0 \\ 0 & \text{else} \end{cases} \quad (6)$$

where y consists of the features showed in Table 1 except homepage and two features drawn from user feedback.

The model of the constraint-based perceptron after importing user feedback is shown in Figure 2. As seen from the graph, two new features r_8 and r_9 are added to the perceptron which are drawn from user feedback. The two features are different from the former several features that their value is not limited to only 1 or 0, it will change according to the user feedback. The appropriate feedback will dominate when there are a great number of user feedback, so the two features actually reflect the major users' view about the name disambiguation result. and the new features will get more accurate as the user feedback multiplied. $Constraint_{feedback}(p_i, p_j)$ drawn from user feedback is utilized to restrict the output of perceptron, so as to make the algorithm more accurate.

The procedures of our proposed method is shown in Figure 3. Firstly, we use the static training set containing only six features to train the perceptron, the output will contain some mistakes, then users find out mistakes and feedback, the feedback will form streams of training set, and also form two new features which will be added to the input of the perceptron. The perceptron revises itself according to feedback training set, updates the weight of each feature and then outputs again, the users will continue to feedback if mistakes are found. This feedback and revise process will continue until no more mistakes are found.

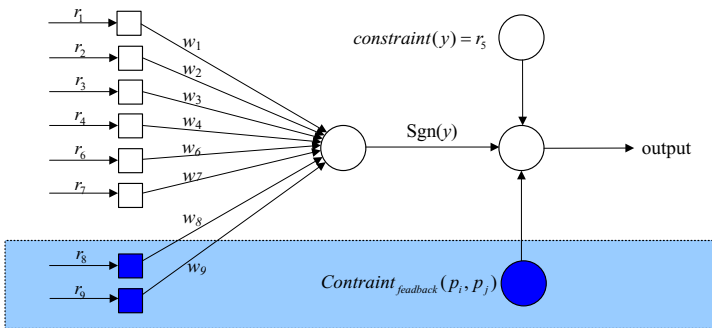


Fig. 2. Incorporating User Feedback into constraint-based Perceptron

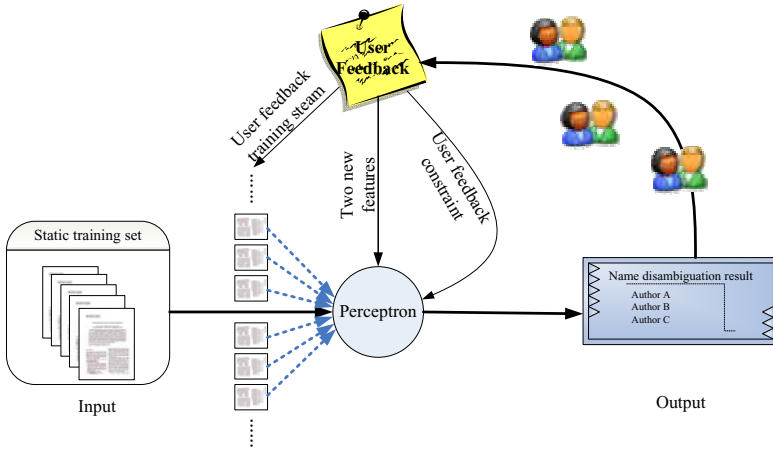


Fig. 3. Procedures of our proposed method

Our algorithm of revising the perceptron with user feedback training stream is as follows.

Algorithm 1. Revising the perceptron with user feedback training stream

Input: User feedback training stream

$[(p_{i1}, p_{j1}, v_1, c_1), (p_{i2}, p_{j2}, v_2, c_2), \dots, (p_{in}, p_{jn}, v_n, c_n)]$

Output: The final output of the perceptron y_t when no more user feedback returned

- 1 Train the perceptron with static training set, and get the initial weights ω_t ;
 - 2 **repeat**
 - 3 Pick a piece of user feedback sample $(p_{in}, p_{jn}, v_n, c_n)$ from the user feedback training stream;
 - 4 Calculate the real value of the user feedback sample $\tilde{y}_t = v_n$;
 - 5 Calculate the output of the perceptron $y_t = \omega_t * \mathbf{x}_t$;
 - 6 Update weights $\omega_{t+1} = \omega_t + \alpha_f * (y_t - \tilde{y}_t) * \mathbf{x}_t$;
 - 7 **until** No more user feedback returned ;
-

In step 5, the two new features drawn from user feedback are added to the \mathbf{x}_t . In step 6, the α_f is the learning rate when using user feedback as training set, the value of α_f is related to c_n in user feedback training sample $(p_{in}, p_{jn}, v_n, c_n)$, when the value of c_n is 1, it means that the training sample is fully credible, and in this time, the value of α_f should be greater than the value of α_f sample when the value of c_n in credible sample is 0.

As seen from Figure 3, our approach is differ from previous methods without introducing user interactions that it can correct itself, since it has continuous learning ability.

5 Experiment

In this section, we report our test on the effectiveness of the proposed approach.

5.1 Dataset

To evaluate our algorithm, we create a dataset from four different online digital library data sets: the DBLP, IEEE, ACM, and Springer. This dataset includes 20 real person names with their 1534 papers. For these names, some only have a few persons. For example, "Juan Carlos Lopez" only represents one person, and "Koichi Furukawa", "David E. Goldberg" and "Thomas D. Taylor" represent three. However, there are 25 different persons named "Bing Liu". To obtain the user feedback and their types, we construct a simple system, one has to register and login to submit his user feedback about the disambiguating results. By this way, we can record his relationship with the author then assign different credibility to the user feedback. Table 2 will give some detail information about the dataset.

Table 2. Number of publications and persons in real name dataset

Name	Pub	Person	Name	Pub	Person
Satoshi Kobayashi	38	6	Bing Liu	215	25
Lei Jin	20	8	R. Ramesh	46	9
David Jensen	53	4	David E. Goldberg	231	3
Thomas Wolf	36	9	Rakesh Kumar	96	12
Koichi Furukawa	77	3	Thomas D. Taylor	4	3
Thomas Tran	16	2	Richard Taylor	35	16
Thomas Hermann	47	9	Jim Gray	200	9
Yun Wang	57	22	Juan Carlos Lopez	36	1
Cheng Chang	27	5	Sanjay Jain	217	5
Gang Luo	47	9	Ajay Gupta	36	9

5.2 Evaluation Measures

We use pairwise measures, namely, Pairwise_Precision, Pairwise_Recall, and F-measure to evaluate the name disambiguation results and for comparison with baseline method. The disambiguation result of paper pairs has two kinds that are written by the same author and by different authors, combined with two kinds of real states. The four states are (1) true positive (tp): paper pairs are written by the same author and the disambiguation result is right; (2) false positive (fp): paper pairs are written by different people while disambiguation think they are written by the same author; (3) true negative (tn): paper pairs are written by different people and disambiguation result also think so; (4) false negative (fn): paper pairs are written by the same author while disambiguation think they are written by different author. The definitions are

$$\text{Pairwise_Precision} = \frac{\text{tp}}{\text{tp} + \text{fp}} \quad (7)$$

$$\text{Pairwise_Recall} = \frac{\text{tp}}{\text{tp} + \text{fn}} \quad (8)$$

$$\text{F - measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (9)$$

Baseline methods. The baselines use the bias classifier learned from each single feature and SA-Cluster which is a graph clustering [20] with the coauthor relationship used as the edge and all the other relationships used as the attribute features, and we use the Pairwise-Classification [19] which combines seven features to the constraint-based perceptron as the no-feedback method for comparison with our approach.

5.3 Experiment Results

The performance of the current algorithm for each author is listed in Table 3 with the performance for baseline method and our algorithm. The results show that our method significantly outperforms the baseline method.

Table 3. Results for 20 real names

Name	Proposed Method			Baseline Method		
	Pre.	Rec.	F	Pre.	Rec.	F
Satoshi Kobayashi	92.05	73.41	81.68	90.12	73.01	80.67
Lei Jin	100	100	100	99.4	99.86	99.63
David Jensen	95.56	87.43	91.31	94.25	87.04	90.5
Thomas Wolf	89.36	33.33	48.55	89.02	31.56	46.6
Koichi Furukawa	96.93	72.72	83.1	96.48	71.59	82.19
Thomas Tran	100	56.04	71.83	99.89	55.87	71.66
Thomas Hermann	100	71.3	83.25	98.89	70.16	82.08
Yun Wang	100	67.65	80.7	99.94	65.1	78.84
Cheng Chang	100	83.95	91.28	98.2	80.96	88.75
Gang Luo	98.41	100	99.2	98.03	97.16	97.59
Bing Liu	88.99	93.88	91.36	88.2	92.46	90.3
R. Ramesh	100	68.12	81.04	99.53	67.46	80.41
David E. Goldberg	99.12	98.26	98.69	99.08	98.08	98.6
Rakesh Kumar	100	97.49	98.73	100	96.43	98.18
Thomas D. Taylor	100	100	100	99.72	100	99.86
Richard Taylor	100	67.82	80.82	99.91	66.76	80.04
Jim Gray	93.76	85.72	89.24	91.5	84.03	87.61
Juan Carlos Lopez	100	89.05	94.21	99.91	87.6	93.35
Sanjay Jain	100	97.74	98.86	99.4	94.16	96.71
Ajay Gupta	100	63.03	77.32	99.75	60.72	75.49

Figure 4 shows the contributions of each feature. For example, Co-Author has very high F-measure because the author names in each paper are complete, compared to the other features. Citation is very useful information because people tend to cite their own papers if they have published related papers. Since

the citation information is crawled from the internet, it is not complete, so the recall is low. The precision of homepage is 100%, since the homepage normally contains the owner's papers only. However owners usually only put their best papers on the homepages and not all authors' homepages are available, so the recall is very low. Title Similarity gives very good performance. Because the title information is complete and an author usually publishes a series of papers in one direction. These authors tend to name their papers in similar ways. More detailed discussions about the contribution of each single feature can refer to our previous work [19].

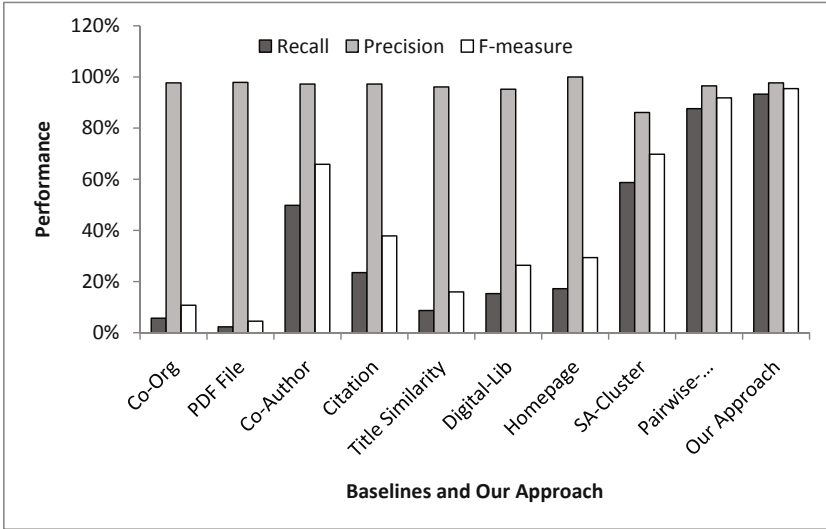


Fig. 4. Comparison of all the baselines and our approach

Specifically, as seen from Figure 4, our proposal significantly outperforms the SA-Cluster method, and despite the inadequate and inaccurate information of user feedback, incorporating user feedback into name disambiguation gives better result in Pairwise_Precision, Pairwise_Recall, and F-measure compared with the no-feedback method Pairwise-Classification.

6 Conclusion

This paper focuses on the problem of name disambiguation in scientific cooperation network. We have proposed a constraint-based perceptron to handle the problem. The method can incorporate features extracted from scientific cooperation network and user feedback to the model. Despite the noises of user feedback, bringing in user feedback gives the best result.

In the future work, we will consider introducing the notion of "credibility" of users, to learn users' credibility dynamically and assign different weights to the feedback according to different credibilities of users, and explore the effects of different kinds of user feedback. Furthermore, we will design more efficient and

convenient user feedback forms to attract more users to submit their feedback. And we will consider the case that there is mis-ordering or missing authors in our methods. In addition to the homepage, reading-list will be investigated to further improve the performance of name disambiguation. This model can also be used in other applications for mining the advisor-advisee relationship, searching scientists, etc.

Acknowledgement. This work is supported by National Natural Science Foundation of China under Grant 70771043, 60873225, 60773191.

References

1. Nguyen, H.T., Cao, T.H.: Exploring wikipedia and text features for named entity disambiguation. In: Nguyen, N.T., Le, M.T., Świątek, J. (eds.) ACIIDS (2). LNCS, vol. 5991, pp. 11–20. Springer, Heidelberg (2010)
2. D'Angelo, C.A., Giuffrida, C., Abramo, G.: A heuristic approach to author name disambiguation in bibliometrics databases for large-scale research assessments. *JASIST* 62(2), 257–269 (2011)
3. Ferreira, A.A., Veloso, A., Gonçalves, M.A., Laender, A.H.F.: Effective self-training author name disambiguation in scholarly digital libraries. In: *JCDL*, pp. 39–48 (2010)
4. Treeratpituk, P., Giles, C.L.: Disambiguating authors in academic publications using random forests. In: *JCDL*, pp. 39–48 (2009)
5. Han, H., Giles, C.L., Zha, H., Li, C., Tsioutsoulouklis, K.: Two supervised learning approaches for name disambiguation in author citations. In: *JCDL*, pp. 296–305 (2004)
6. Cota, R.G., Ferreira, A.A., Nascimento, C., Gonçalves, M.A., Laender, A.H.F.: An unsupervised heuristic-based hierarchical method for name disambiguation in bibliographic citations. *JASIST* 61(9), 1853–1870 (2010)
7. Han, H., Zha, H., Giles, C.L.: Name disambiguation in author citations using a k-way spectral clustering method. In: *JCDL*, pp. 334–343 (2005)
8. Yang, K.H., Peng, H.T., Jiang, J.Y., Lee, H.M., Ho, J.M.: Author name disambiguation for citations using topic and web correlation. In: Christensen-Dalsgaard, B., Castelli, D., Ammitzbøll Jurik, B., Lippincott, J. (eds.) *ECDL 2008*. LNCS, vol. 5173, pp. 185–196. Springer, Heidelberg (2008)
9. Kang, I.S., Na, S.H., Lee, S., Jung, H., Kim, P., Sung, W.K., Lee, J.H.: On co-authorship for author disambiguation. *Inf. Process. Manage.* 45(1), 84–97 (2009)
10. Jin, H., Huang, L., Yuan, P.: Name disambiguation using semantic association clustering. In: *ICEBE*, pp. 42–48 (2009)
11. Zhang, D., Tang, J., Li, J.Z., Wang, K.: A constraint-based probabilistic framework for name disambiguation. In: *CIKM*, pp. 1019–1022 (2007)
12. Wang, F., Tang, J., Li, J., Wang, K.: A constraint-based topic modeling approach for name disambiguation. *Frontiers of Computer Science in China* 4(1), 100–111 (2010)
13. Wang, C., Han, J., Jia, Y., Tang, J., Zhang, D., Yu, Y., Guo, J.: Mining advisor-advisee relationships from research publication networks. In: *KDD*, pp. 203–212 (2010)

14. Liu, Y., Zhang, M., Ru, L., Ma, S.: Automatic query type identification based on click through information. In: Ng, H.T., Leong, M.-K., Kan, M.-Y., Ji, D. (eds.) AIRS 2006. LNCS, vol. 4182, pp. 593–600. Springer, Heidelberg (2006)
15. Joachims, T., Granka, L.A., Pan, B., Hembrooke, H., Gay, G.: Accurately interpreting clickthrough data as implicit feedback. In: SIGIR, pp. 154–161 (2005)
16. Cao, H., Jiang, D., Pei, J., He, Q., Liao, Z., Chen, E., Li, H.: Context-aware query suggestion by mining click-through and session data. In: KDD, pp. 875–883 (2008)
17. Richardson, M., Dominowska, E., Ragno, R.: Predicting clicks: estimating the click-through rate for new ads. In: WWW, pp. 521–530 (2007)
18. Chai, X., Vuong, B.Q., Doan, A., Naughton, J.F.: Efficiently incorporating user feedback into information extraction and integration programs. In: SIGMOD Conference, pp. 87–100 (2009)
19. Lin, Q., Wang, B., Du, Y., Wang, X., Li, Y.: Disambiguating authors by pairwise classification. *Tsinghua Science and Technology* 15(6), 668–677 (2010)
20. Zhou, Y., Cheng, H., Yu, J.X.: Graph clustering based on structural/attribute similarities. *PVLDB* 2(1), 718–729 (2009)

Multi-core vs. I/O Wall: The Approaches to Conquer and Cooperate

Yansong Zhang^{1,2}, Min Jiao^{1,3}, Zhanwei Wang^{1,3}, Shan Wang^{1,3}, and Xuan Zhou^{1,3}

¹ DEKE Lab, Renmin University of China, Beijing 100872, China

² National Survey Research Center at Renmin University of China, Beijing 100872, China

³ School of Information, Renmin University of China, Beijing 100872, China
{zhangys_ruc, swang}@ruc.edu.cn

Abstract. Multi-core comes to be the mainstream of processor techniques. The data-intensive OLAP relies on inexpensive disks as massive data storage device, so the enhanced processing power oppose to I/O bottleneck in big data OLAP applications becomes more critical because the latency gap between I/O and multi-core gets even larger. In this paper, we focus on the disk resident OLAP with large dataset, exploiting the power of multi-core processing under I/O bottleneck. We propose optimizations for schema-aware storage layout, parallel accessing and I/O latency aware concurrent processing. On the one hand I/O bottleneck should be conquered to reduce latency for multi-core processing, on the other hand we can make good use of I/O latency for heavy concurrent query workload with multi-core power. We design experiments to exploit parallel and concurrent processing power for multi-core with DDTA-OLAP engine which minimizes the star-join cost by directly dimension tuple accessing technique. The experimental results show that we can achieve maximal speedup ratio of 103 for multi-core concurrent query processing in DRDB scenario.

Keywords: I/O wall, OLAP, multi-core OLAP, processing slots, DDTA-JOIN.

1 Introduction

I/O performance is always the bottleneck for data-intensive OLAP(On-line Analytical Processing) applications. The hard-drive seek latency and rotational latency are decided by physical structure and the gap between hard disk and SSD, RAM, CPU grows larger. In data warehouse, big data have to depend on inexpensive and large capacity hard disks. At the same time, multi-core with several-fold data-consuming capacity makes the unbalanced architecture even skewed, and the powerful multi-core is blocked by the wall of I/O.

OLAP is typical data-intensive application with producer-consumer model, the storage engine is producer with low throughput and the processor is consumer with high performance, the memory is buffer between producer and consumer. For single producer-consumer pair, the key to improve system performance is to improve producer performance. While for single producer and multiple consumer scenario, the more important thing is to promote consumer efficiency upon the slow producer. So we should consider the multi-core OLAP as a whole and perform deep research on when, where and how the multi-core optimizations are implemented.

OLAP workload is characterized as long-run queries with complex processing on big dataset. Many researches focused on how to improve performance for single query processing with storage optimization and parallel processing. As OLAP concurrent workload grows rapidly, more and more researchers pay attention to how to perform efficient concurrent processing in DRDB(Disk Resident Database) scenario. We classify the roadmaps into three types: (1) Improving hard disk accessing efficiency. (2) Perform parallel accessing with RAID or multiple hard disks. (3) Sharing the low performance table scan on hard disk. The first two roadmaps try to reduce I/O latency and the third roadmap tries to take use of I/O latency to serve for concurrent queries.

In this paper, we introduce our researches on DDTA-OLAP(Directly Dimensional Tuple Accessing-OLAP) model with schema-aware storage mechanism, parallel hard disk accessing and concurrent processing. The background is state-of-the-art server with multi-core processors, large main memory, and multiple disks. The application scenario is typical star schema with high concurrent query workload. We implement an unbalanced storage model for the skewed star schema in which the small size but frequently accessed dimensions are memory resident with column-oriented model during query processing and the large fact table employs traditional row-oriented storage engine for compatibility. We adopt vertical partition technique as tradeoff policy between I/O efficiency and I/O sharing. DDTA-OLAP inherits key-to-address mapping feature from multi-dimension OLAP model(MOLAP), so the complex star-joins from fact table to multiple dimension tables are replaced by direct data accessing with mapped address. The key-to-address mechanism enables complex query plan to be simplified as atomic plan with logical on-the-fly pre-join table without private dataset such as hash table which is adapted to perform in parallel with horizontal partition policy. Furthermore, the key-to-address mapping keeps dimensions as public address based lookup tables, the concurrent queries eliminate space contention for private hash tables within small cache during multi-core parallel processing. These optimizations enable data producer by I/O supporting more consumers by multi-core processors.

Our contributions are as follows:

1. I/O bottleneck is a traditional concept for DRDB and is emphasized in multi-core era due to even unbalanced producer-consumer model. We propose a comprehensive storage model for diverse scenarios, exploiting the pros and cons the I/O cost produced for multi-core processing.
2. We exploit the optimization policy on multiple hard disk system and RAID.
3. We propose a cost model for multi-core concurrent query processing to exploit the maximal parallelism for multi-core under I/O bottleneck.

The related work is presented in Sec.2. Multi-core DDTA-OLAP model is discussed in Sec.3. Sec.4 shows the results of experiments. Finally, Sec.5 summarizes the paper and discusses future work.

2 Related Work

In this section, we analyze the I/O optimization roadmaps following the classifications mentioned in previous section.

(1) Improving hard disk accessing efficiency

If sequential scan and random scan are concurrently performed on same disk, the whole performance will be degraded much by head contention. [1] proposed a workload-aware storage layout for analytic database, in which dimension table, temporary tables and indexes are stored together while large fact table is stored separately to eliminate access contention between sequential scan and random scan. Another way to promote I/O efficiency is to reduce necessary I/O blocks by column-store and compression. When only very small portion of columns are accessed in a wide table, column-store can improve I/O efficiency by only loading necessary columns[2,3,4,5]. Column-oriented model is suitable for wide while sparse accessed benchmarks such as TPC-H, SSB[9] but not suitable for narrow fact table such as Foodmart demo database in many OLAP systems. Column-store can also improve cache performance. MonetDB[6] employs BAT algebra instead of relational algebra for cache optimization. PAX[7] reorganizes tuples as column-store inside blocks, so update on multiple attributes only occurs one I/O as row-store and blocks in memory acts as column-store to improve cache performance. PAX optimizes not I/O performance but memory bandwidth. [8] attempts to achieve similar performance as column-store by row-store model. Table based compression techniques are proposed to improve storage efficiency for less I/O cost. Column-oriented model benefits from I/O efficiency but suffers from column combination cost, the traditional pipeline optimization is hard to be employed due to materialization mechanism like MonetDB. Some column-oriented database, such as C-store, Greenplum, and Infobright adopt on-the-fly layout conversion[3] during scan operation to keep compatible with row-oriented query engine. [10] proposed an DSM(column-oriented store)/NSM(row-oriented store) layout conversion to exploit SIMD performance. As conversion, tuples are processed through memory to processor for candidate layout format, the processing cycles and synchronous cost are considering issues. For I/O to memory DSM/NSM layout conversion, the individual stored column files are to be loaded into memory synchronously, for example if 4 columns are involved in query, each i -th block should be loaded into memory synchronously to prepare for on-the-fly DSM/NSM layout conversion. Reading i -th blocks from each columns means random access on disk which will occur disk head contention.

(2) Perform parallel accessing with RAID or multiple hard disk architecture

DSM/NSM layout conversion and horizontal partition based parallel processing on hard disk suffer from disk head contention. Data layout re-organization[1] on multiple disk can be adopted for DSM or vertical partition re-organization. If 4 columns are stored on 4 disks then reading i -th blocks from 4 columns can be performed in parallel. The stripe RAID is hardware level solution for improving I/O latency, nowadays moderate server usually configures with RAID adaptor to control multiple hard disks.

(3) Sharing the high latency table scan on hard disk

For single long-run query, I/O latency is a wall to be break. For concurrent queries, I/O latency can serve for processing slots to share the buffered blocks. Redbrick

DW[11] shared disk I/O among multiple scan operations for concurrent queries on multiple processors. Cooperative scans [12] improved data sharing across concurrent scans by dynamically scheduling queries and their data requests. DataPath[13] proposed a data-centric model for continuously pushing data to processors, queries are attached to data stream for computing. CJOIN[14] is another approach to normalize star schema queries(SSB benchmark) as a shared continuous table scan and filter pipeline processing for concurrent queries. From the roadmap of shared scan we can summarize that the ideal model is to share unique sequential scan on disk for large concurrent and low latency in-memory processing i.e. single producer for maximal I/O efficiency and as many as possible on-the-fly consumers. The existed approaches rely on data stream(JDBC stream in CJOIN), the cost model is absence without detailed information of I/O and processing latency.

I/O wall remains with hierarchical storage architecture, the lower and inexpensive large storage device continues to be bottleneck no matter whether SSD or other advanced storage device taking the place of hard disks. Whether the I/O wall should be conquered or cooperated depends on diverse application requirements.

3 Disk Resident DDTA-OLAP for Multi-core

[15] summarized the killer requirements of low latency, heavy concurrent workload and complex processing for nowadays OLAP applications. The storage model for OLAP under state-of-the-art hardware is not merely the decision for column- or row-oriented model should be employed, but a comprehensive consideration for what kind of physical storage device the system relies on, the schema feature and workload character, the workload of concurrent queries, etc.

3.1 DDTA-OLAP Model

We have proposed a DDTA-JOIN[16] based OLAP system. Figure 1 illustrates the rational of DDTA-JOIN. The preliminaries are as follows:

- ◆ Dimension tables are small in size than fact table and can be full or partial resident in main memory;
- ◆ Primary key in dimension table is normalized as order reserved consecutive sequence such as 1, 2, ...;
- ◆ Dimension columns are stored as arrays in memory and the array index can be directly mapped from dimension key;
- ◆ Dimension tables can be updated with appending, update and delete operations in OLTP system, the primary keys are reserved for updated(may be new tuple with old primary key) and deleted tuples(empty tuple or marked tuple). After updating, dimension columns are re-loaded into memory;
- ◆ Join from fact tuple to dimension tuple is normalized as directly accessing dimension array with array index mapped by dimension key in fact tuple.

DDTA-OLAP model normalizes complex OLAP queries into the same table scan style processing, all the queries share the public dimension columns. The example query and processing is described as follows:

```

SELECT c_nation, sum(lo_ordertotalprice) as profit
FROM customer, supplier, part, lineorder
WHERE lo_custkey = c_custkey
      AND lo_suppkey = s_suppkey
      AND lo_partkey = p_partkey
      AND c_region = 'AMERICA'
      AND s_region = 'AMERICA'
      AND (p_category = 'MFGR#41' OR p_category = 'MFGR#42')
GROUP BY c_nation ORDER BY c_nation;

```

Query processing begins with fact table scan, as each fact tuple is sequentially accessed, three dimension key with related predicate expressions (for example, $lo_suppkey:4$, $lo_partkey:2$ and $lo_custkey:5$) are mapping to dimension array units of $s_region[3]$, $p_category[1]$ and $c_region[4]$, then the predicate expression:

```

c_region[4] = 'AMERICA' AND s_region[3] = 'AMERICA'
AND (p_category[1] = 'MFGR#41' OR p_category[1] = 'MFGR#42')

```

is processed, when we get a **TRUE** result, dimension attribute of $c_nation[4]$ and $lo_ordertotalprice$ in fact tuple are extracted to form result tuple and pipelined to hash group-by operator for aggregating, when all fact tuples are processed, the aggregate results are sorted by ORDER-BY clause.

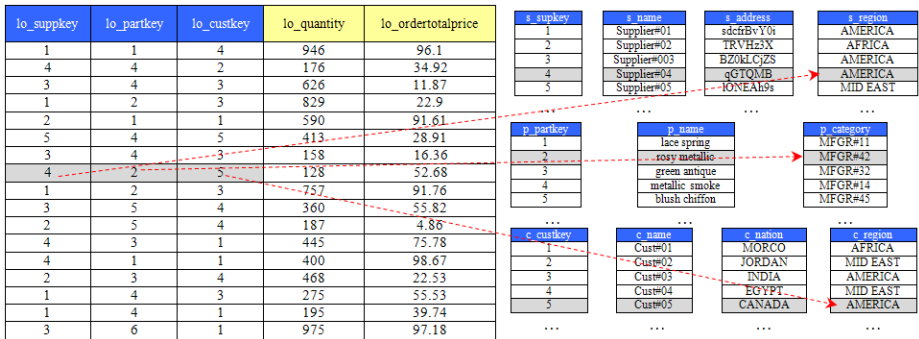


Fig. 1. DDTA-OLAP model

3.2 Storage Model for Multi-core Processing

DDTA-OLAP is table scan based processing. The small but frequently accessed dimension tables are column-oriented and memory resident to minimize accessing latency and improve cache performance.

The large fact table can be row- or column-oriented model. The column-oriented model should support on-the-fly DSM/NSM conversion to keep compatible with once a tuple style processing. Figure 2 shows the statistical information of field access frequency for SSB benchmark. For sequentially executed workload, column-oriented fact table can minimize I/O cost by only loading columns as needed. For concurrent workload, it is important to organize multiple scans as single shared scan to eliminate disk head contention. For example, we can divide fact table in SSB into two vertical

latter is a $1+(n-1)$ -thread mode. The n -thread mode provides maximal parallelism for multi-core processing, if the aggregate hash table is large, more cache capacity conflicts may degrade processing performance. The $1+(n-1)$ -thread mode sacrifices one processing thread, if aggregate hash table is large, this policy can reduce overall aggregating cost like StageDB[17]. The SMP(inside are multi-core processors) architecture makes data migrate cross cores by high latency RAM, if the aggregate result set is much smaller than L2 cache, the n -thread mode is adaptive to be implemented and efficient. In SSB, the aggregate result set size varies from 1 to maximal 800 tuples, we employ n -thread mode in experimental part.

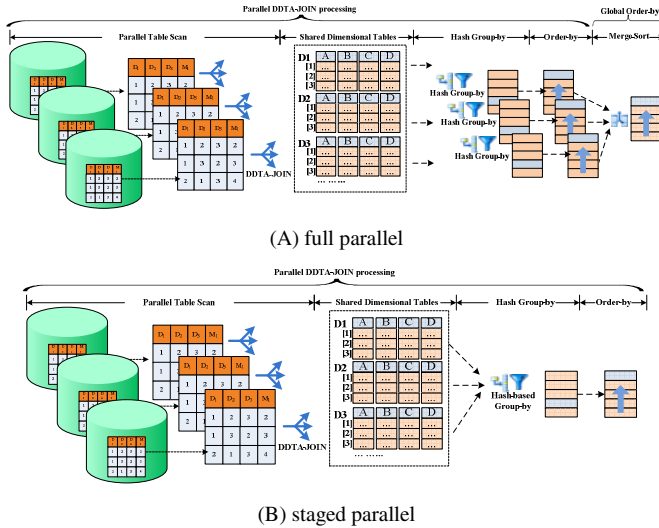


Fig. 3. Intra-parallel DDTA-OLAP for DRDB scenario

RAID 0 stripes data across disk array to achieve parallel I/O. Assuming that there are 4 horizontal partitions(A, B, C, D) on a 4-disk array like figure 4(A1, A2,... are striped blocks), the left figure is a conflicting data layout case because parallel accesses on 4 horizontal partitions((A1, B1, C1, D1), (A2, B2, C2, D2), ...) generate conflicts on each disk; the right figure is the ideal data layout case that parallel accesses on 4 horizontal partitions have no conflicts with each other. We can hardly configure data layout on RAID with best strategy, so we can get parallel benefits by physically partitioning fact table into n sub tables to perform intra-parallel DDTA-OLAP for DRDB, the theoretic speedup ratio is no more than parallel disks.

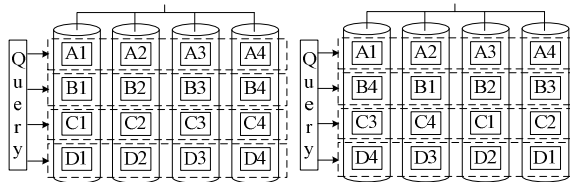


Fig. 4. Parallelism analysis of RAID

B. Inter-parallel DDTA- OLAP for DRDB

More and more users and applications rely on the analytical information, the workload of concurrent query increased rapidly in recent years discovered by many survey reports.

To share the time-consuming table scan from hard disk in DRDB is an interesting issue from Red Brick in the 1990’s to CJOIN[14] and DataPath[13]. The table scan sharing based inter-parallel can be described as how can a slow producer satisfy as many fast consumers as possible. The producer speed is fixed and decided by hardware performance(I/O rate), so the inter-parallel performance is decided by how fast the query processing can achieve. CJOIN optimizes tree-shade physical plan into linear hash filters to accelerate query processing, and Blink[18] uses row-wise compressed de-normalized table and bitwise operation based predicate processing to eliminate join cost and normalize query processing as constant time(Blink is memory resident system, the de-normalization technique can be implemented into DRDB scenario). In this paper, we use DDTA-JOIN as a faster “consumer” for inter-parallel, DDTA-JOIN has less processing latency than CJOIN’s pipelined hash filters operation and DDTA-JOIN has high performance in throughput by smaller tuple width than de-normalized Blink.

Figure 5 illustrates inter-parallel strategy for highly concurrent analytical workloads. We perform the continuous scan on fact table just like CJOIN, each concurrent query starts an independent thread to share table scan in parallel. This strategy is flexible for both static and dynamic query task management, a query task can be dynamically attached to table scan loops with start position k , the complete query processing can be finished after reading first $k-1$ tuples in next scan circle.

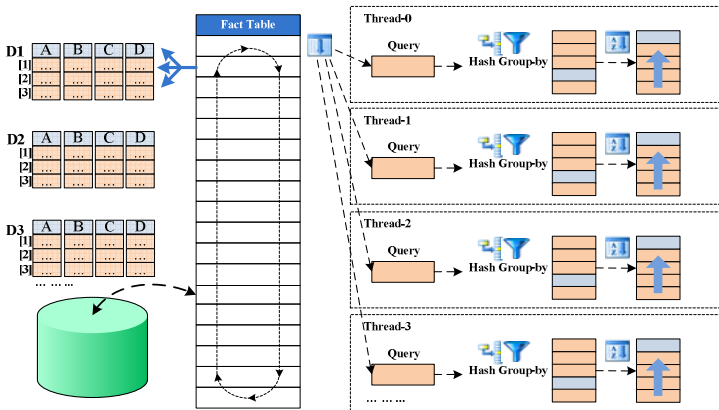


Fig. 5. Inter-parallel DDTA-OLAP for DRDB scenario

3.4 Cost Model for Multi-core DDTA-OLAP

DDTA-JOIN optimizes and normalizes OLAP processing, leaving I/O performance as the last bottleneck. Intra-parallel DDTA-OLAP can exploit parallel processing with support of parallel storage devices. But the parallel I/O still far slower than

in-memory OLAP processing, the power of multi-core is still underutilized. Inter-parallel DDTA-OLAP model sets up a one-to-many(one producer to many consumers) model to bridge high latency serial device to low latency parallel processors. The latency of I/O is a brick wall for sequential OLAP processing to conquer but produces many processing slots to contain as many concurrent query workload as possible. The power of multi-core can be exploited based on the ratio of I/O latency and DDTA-OLAP executing time.

In DRDB, the cost of reading a fact table block from hard disk dominates the total cost of query processing. Let $T_{I/O}$ denote time of fetching a block from hard disk, T_{avgQP} denotes the average time of DDTA-OLAP query processing. In practice, $T_{avgQP} = T_{parsing} + T_{processing}$, $T_{parsing}$ denotes the time of parsing a page-slot tuple into an in-memory structure to reduce latency during concurrent accessing. The ideal scenario is that when continuous I/Os are performed, there are maximal $(T_{I/O} - T_{parsing}) / T_{processing}$ parallel DDTA-OLAP processing threads share the buffered block, parallel queries are finished on buffered block as soon as new block is coming, no waiting time for parallel queries or I/O fetcher. The cost model for concurrent query processing is described as follows:

$$conR = \frac{T_{I/O} - T_{parsing}}{T_{processing}} \times n_{cores} \times \gamma, \text{ where:}$$

$conR$ denotes parallel speedup ratio of concurrent DDTA-OLAP;

$T_{I/O}$ denotes time of fetching block from storage device;

$T_{parsing}$ denotes time of parsing tuples in block as in-memory processing structure;

$T_{processing}$ denotes time of in-memory DDTA-OLAP processing;

n_{cores} denotes available processing cores;

γ denotes parallelism degradation ratio for multi-core processing, including synchronous degradation, memory bandwidth and cache conflicting degradation, etc.

4 Experiments

All our experiments are conducted on a HP-350 server with two Quad-core Intel Xeon E5310 1.6GHz CPUs, 6GB RAM and four 80GB hard disks as RAID running Linux version 2.6.30-0.1. We use dataset with SF=32 for DRDB testing with standard SSB data generator. DDTA-OLAP is developed on interfaces of PostgreSQL to keep compatible with PostgreSQL and provide a near real system performance. We use storage engine of PostgreSQL, design DDTA-JOIN algorithm by rewriting table scan module, the group-by and order-by algorithms are developed with inner interfaces of PostgreSQL, i.e., we plug our DDTA-JOIN algorithm in a standard table scan module.

4.1 Basic DDTA-OLAP Performance for DRDB

We compared DDTA-OLAP against PostgreSQL-8.3.4 in the DRDB scenario. The PostgreSQL was tuned carefully to achieve its optimal performance. In the experiment, we used the dictionary encoding mode[19] for DDTA-OLAP. The results are shown in Figure 6.

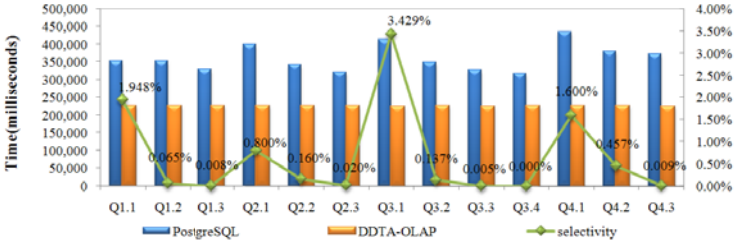


Fig. 6. DDTA-OLAP vs. PostgreSQL

As we can see, the performance of PostgreSQL varies sharply with the selectivity of the query, while the performance of DDTA-OLAP is quite consistent without tuning. This is because DDTA-OLAP utilizes the virtual de-normalized table and asynchronous block accessing, which bound the execution time to that of the optimized sequential scan. By contrast, PostgreSQL has to perform a large number of join operations, whose random I/Os dominate the query processing time. In average, DDTA-OLAP is 1.59 times as fast as PostgreSQL. It can be foreseen that, for more complex star schemas or snowflake schemas, the join-cost would further deteriorate the performance of a traditional DRDB.

4.2 Intra-Parallel DDTA-OLAP for DRDB

For DRDB based DDTA-JOIN, the parallelism is decided by parallel physical channels of hard disks. Firstly, we configure two disks as RAID 1 mode(mirrored disks), so the server has two independent hardware I/O channels. Parallelism with 2 physical partitions achieves the best speedup ratio of 1.55(average speedup ratio). When partition grows, the I/O performance suffers from contention of disk head and proves low parallelism as shown in figure 7. Secondly, we configure 4 hard disks as RAID 0 mode (striped set) to verify parallelism of physical partition based parallel processing. The data distribution is not controlled by user but by array controller, physical partition can provide parallelism with maximal speedup ratio of 1.71, the results are shown in figure 8.

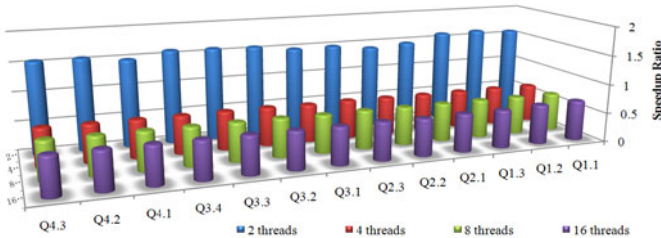


Fig. 7. Parallelism of intra-parallel DDTA-JOIN in DRDB scenario(RAID 1)

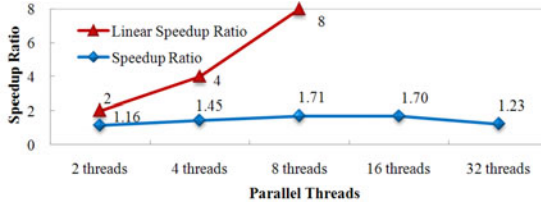


Fig. 8. Parallelism of intra-parallel DDTA-JOIN in DRDB scenario(RAID 0)

4.3 Inter-parallel DDTA-OLAP for DRDB

In our final set of experiments, we evaluate the performance of DDTA-OLAP's inter-parallel mechanism. We use all the 13 SSB testing queries and organize query groups in a round-robin order. In the experiments, we gradually increase the parallelism (the number of concurrent queries), and observe the throughput of the system.

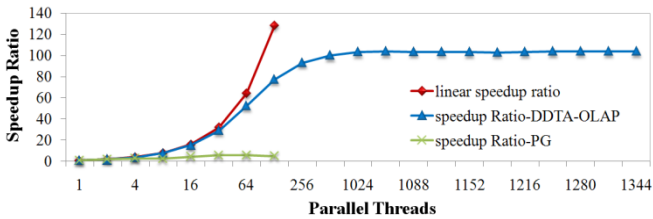


Fig. 9. Speedup Ratio of Inter-Parallel DDTA-OLAP

The results are shown in Figure 9. (Please note that the parallelism varies exponentially from 1 to 1024, but only linearly after 1024, because the speedup ratio converges after 1024.)

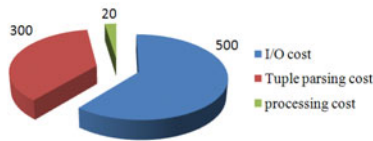


Fig. 10. Processing cost for concurrent speedup ratio analysis

The blue line in Figure 9 shows the speedup ratios achieved by different granularity of parallelism. The red line indicates the linear speedup ratio. The green line is the speedup ratios achieved by concurrent PostgreSQL. In query processing of DRDB, I/O latency dominates the whole performance. The system needs to process as many concurrent queries as possible during each I/O interval.

However, the parallelism cannot grow infinitely. To ensure an uninterrupted sequential scan of the fact table, we need to keep the parallelism below $(T_{I/O} - T_{parsing}) / T_{processing} = (500 - 300) / 20 = 10$. The speedup ratio of 8-core parallel processing is about

5.8, so the theoretical speedup ratio for concurrent processing is $5.8 \times 10 = 58$. This can be seen clearly in Figure 10 –when the parallelism is below the threshold (58 in our experiments), the system achieves a near linear speedup ratio; when it exceeds the threshold, the speedup ratio starts to fall behind the linear ratio and gradually achieves its maximum. This is because the data processing becomes slower than the sequential I/O at the threshold and forces the I/O to be suspended for synchronizing the concurrent queries. As the overhead of data processing increases, I/O latency increases too and the benefit of parallelism weakens. With an 8-core system, our DDTA-JOIN achieved a speedup ratio of 103 in the experiments. In contrast, the maximal speedup ratio achieved by PostgreSQL is only 5.84. (Its performance becomes unacceptable when the parallelism exceeds 128.)

5 Conclusions and Future Work

I/O latency will remain bottleneck of the data-intensive application in the hierarchical storage architecture, as the top level, multi-core processor enlarges the gap between data producer and consumer. The straightforward parallel processing relies on parallelism of different levels, such as main memory and hard disk, the physical character of storage device dominates the overall parallelism. As a result in this paper, concurrent processing achieves the maximal benefits upon I/O wall, the parallelism relies on in-memory processing efficiency and processing slots produced by I/O latency. For heavy concurrent workload, I/O latency enables multi-core concurrently serving for more requirements. The specified support which enables multi-core cooperating with I/O wall is DDTA-OLAP model, DDTA-OLAP model minimizes memory consuming and star-join processing cost which provides more processing slots during I/O latency. Another key technique is to exploit low level interfaces of PostgreSQL to synchronize concurrent queries with I/O accessing. The dynamic concurrent query processing will be our future work for prototype system which is more close to real-world requirements.

Acknowledgements. This work was supported by the National High Technology Research and Development Program of China (2009AA01Z149), the Major National Science and Technology Project of China (2010ZX01042-001-002-002), the projects of Grant No.61070054,10XNI018,10XNB053,10XNH096 and 11XNH120.

References

1. Ozmen, O., Salem, K., Schindler, J., Daniel, S.: Workload-aware storage layout for database systems. In: SIGMOD Conference 2010, pp. 939–950 (2010)
2. MacNicol, R., French, B.: Sybase IQ Multiplex -Designed for analytics. In: Proceedings of VLDB (2004)
3. Stonebraker, M., Abadi, D.J., Batkin, A., Chen, X., Cherniack, M., Ferreira, M., Lau, E., Lin, A., Madden, S., O’Neil, E.J., O’Neil, P.E., Rasin, A., Tran, N., Zdonik, S.B.: C-Store: A Column-oriented DBMS. In: Proceedings of VLDB, Trondheim, Norway, pp. 553–564 (2005)

4. Abadi, D.J., Madden, S.R., Hachem, N.: Column-Stores vs. Row-Stores: How Different Are They Really? In: *Proceeding of SIGMOD*, Vancouver, BC, Canada (2008)
5. Zukowski, M., Nes, N., Boncz, P.A.: DSM vs. NSM: CPU performance tradeoffs in block-oriented query processing. In: *DaMoN 2008*, pp. 47–54 (2008)
6. Boncz, P.A., Mangegold, S., Kersten, M.L.: Database architecture optimized for the new bottleneck: Memory access. In: *VLDB*, pp. 266–277 (1999)
7. Ailamaki, DeWitt, D.J., Hill, M.D.: Data page layouts for relational databases on deep memory hierarchies. *The VLDB Journal* 11(3), 198–215 (2002)
8. Bruno, N.: Teaching an Old Elephant New Tricks. In: *CIDR 2009*, Asilomar, California, USA (2009)
9. O’Neil, P., O’Neil, B., Chen, X.: The Star Schema Benchmark (SSB), <http://www.cs.umb.edu/~poneil/StarSchemaB.PDF>
10. Zukowski, M., Nes, N., Boncz, P.A.: DSM vs. NSM: CPU performance tradeoffs in block-oriented query processing. In: *DaMoN 2008*, pp. 47–54 (2008)
11. Fernandez, P.M.: Red Brick warehouse: A read-mostly RDBMS for open SMP platforms. In: *ACM SIGMOD Intl. Conf. on Management of Data* (1994)
12. Zukowski, M., Héman, S., Nes, N., Boncz, P.: Cooperative scans: dynamic bandwidth sharing in a DBMS. In: *Intl. Conf. on Very Large Data Bases* (2007)
13. Arumugam, S., Dobra, A., Jermaine, C.M., Pansare, N., Perez, L.L.: The DataPath system: a data-centric analytic processing engine for large data warehouses. In: *SIGMOD Conference 2010*, pp. 519–530 (2010)
14. Candea, G., Polyzotis, N., Vingralek, R.: A scalable, Predictable Join Operator for Highly Concurrent Data Warehouse. In: *VLDB 2009*, pp. 277–288 (2009)
15. SAP NetWeaver: A Complete Platform for Large-Scale Business Intelligence. Winter Corporation White Paper (May 2005)
16. Zhang, Y., Hu, W., Wang, S.: MOSS-DB: A Hardware-Aware OLAP Database. In: Chen, L., Tang, C., Yang, J., Gao, Y. (eds.) *WAIM 2010*. LNCS, vol. 6184, pp. 582–594. Springer, Heidelberg (2010)
17. Harizopoulos, S., Ailamaki, A.: StagedDB: Designing Database Servers for Modern Hardware. *IEEE Data Eng. Bull.* 28(2), 11–16 (2005)
18. Johnson, R., Raman, V., Sidle, R., Swart, G.: Row-wise parallel predicate evaluation. In: *Proceedings of the 32nd International Conference on Very Large Data Bases*, Auckland, New Zealand (2008); *VLDB Endowment* 1(1), 622–634
19. Binnig, C., Hildenbrand, S., Färber, F.: Dictionary-based order-preserving string compression for main memory column stores. In: *SIGMOD Conference 2009*, pp. 283–296 (2009)

W-Order Scan: Minimizing Cache Pollution by Application Software Level Cache Management for MMDB

Yansong Zhang^{1,2}, Min Jiao^{1,3}, Zhanwei Wang^{1,3}, Shan Wang^{1,3}, and Xuan Zhou^{1,3}

¹ DEKE Lab, Renmin University of China, Beijing 100872, China

² National Survey Research Center at Renmin University of China, Beijing 100872, China

³ School of Information, Renmin University of China, Beijing 100872, China

{zhangys_ruc, swang}@ruc.edu.cn

Abstract. The utilization of shared LLC (Last Level Cache) is important for efficiency of multi-core processor. Uncontrolled sharing leads to cache pollution i.e. the weak locality data (single-usage data without re-using) continuously evict the strong locality data (frequently re-used data) from LLC in both inner query processing and co-running programs. For analytical MMDB (Main-Memory Database) applications, with skewed star schema of DW, more than 95% memory capacity is occupied by memory-resident fact table with weak locality and there are only small size dimension tables with strong locality. Cache partitioning must manage data with different localities inside query processing to avoid cache pollution by weak locality fact table. The static OS-based cache partitioning suffers from insufficient memory address capacity due to large fact table and the dynamic OS-based cache partitioning also suffers from data movement overhead during cache re-allocation. In order to employ a practical and effective cache partitioning policy, we propose an application software-based *W-order* scan policy for real analytical MMDB application. The consecutive physical address based *W-order* policy is proposed to reduce cache misses with high memory utilization by controlling the physical page accessing order within large and consecutive physical pages. Another approach is page-color index i.e. we extract page-color bits from pages of weak locality data and sort the page address by page-color bits, when we perform a page-color index scan, we can control the physical page accessing order too without supporting from OS for large consecutive physical page allocating. We measure the L2 cache miss rate by simulating a typical hash join operation. The experimental results show that DBMSs can improve cache performance through controlling weak locality data accessing pattern by themselves oppose to depending on supports by hardware or OS.

Keywords: cache pollution, *W-order* scan, cache partitioning, page-color index.

1 Introduction

On-chip shared LLC architecture is the mainstream technique in state-of-the-art multi-core processors. For DBMSs, buffer management between hard-disk and

processor seems similar to cache management between main-memory and multi-core processors. The vital difference between the two techniques is that the former is controlled by DBMS while the latter is controlled by hardware i.e. it is uncontrolled by software.

In MMDB based OLAP scenario, the large fact table and small dimension tables are all memory resident, and the fact table occupies large consecutive addresses. Considering the typical hash-join operation in which the sequential scan on large fact table continuously evicts cached data, the evicted cached hash table data leads to degradation of performance due to cache misses occurred by frequently re-used hash table data. The key issue of improving shared L2 cache utilization is how to limit weak locality data with no re-using like sequentially scanned fact table into small cache regions to reduce conflicts with strong locality data like hash table.

Most of existing researches focused on cache partitioning for threads[2,3] i.e. allocating or re-allocating memory of weak locality thread within less page colors and allocating memory of strong locality thread within more page colors. The ideal scenario is that the weak locality data are allocated within single color. In practice, the large majority weak locality data occupy almost all the colors. If we remove each weak locality page to selected page color before loading into cache, the cost of *memcpy* is unacceptable. In spite of data movement overhead and memory utilization, the thread-level cache partitioning policy is a coarse granularity that it cannot optimize cache pollution inside query processing thread. Figure 1 shows the experimental results of cache miss rate by simulating a typical hash join operation. To analyze the influence of cache pollution, we first run a simplified hash join program with 1GB sequential scan and 1MB, 2MB, 4MB, 8MB hash tables(with 6MB shared L2 cache in a dual core processor), then we run another hash join without fetching data from memory by on-the-fly generating fact tuple with random tuple generator. We can see that cache pollution by conflicts between sequential data and hash tables within the shared L2 cache size is the major factor for cache miss rate. Furthermore, for parallel query processing with homogeneous sub query plans and concurrent query processing with similar query plan, thread-level cache partitioning policy can hardly further optimize cache conflicts.

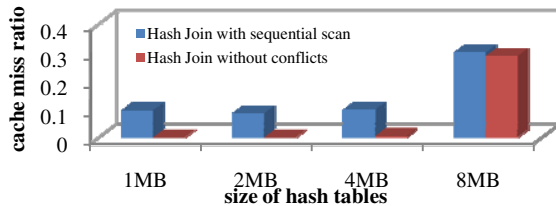


Fig. 1. Effects of cache pollution

The key to reduce cache pollution is to allocate weak locality data within fewer colors in cache. The challenging issue is how to load large data from memory to cache through limited colors of pages in memory. The static page-color policy suffers from insufficient page-color capacity and the dynamic policy as motioned in [2] suffers from data movement overhead during page-recoloring.

In this paper, we propose a *W-order* scan to reduce cache pollution by controlling page accessing order of sequential scan for analytical MMDB. We transform linear data accessing order into *W-order* which enables the memory pages to be accessed by page-color order to reduce cache pollution. During accessing with single page-color, only strong locality data with the same page-color may involve cache conflicting with the weak locality data, the data with other page-colors will not be evicted by the weak locality data. One way is to control a large consecutive physical block in memory and we can directly control the page-color based memory accessing by address computing. In this paper, we employ *kmalloc* to allocate memory for consecutive physical block, and we can allocate maximal 128MB consecutive physical block size for weak locality data to perform an inner *W-order* scan. Another way is to control page accessing order by page-color index. We create page-color index to record the color of each page, so we can organize physical page accessing order according to color order in page-color index. This approach needs no support from OS and the index is small because each page generates only one index item. These two approaches can also be named as physical and logical *W-order* scan.

The *W-order* scan is application software level cache optimization compared with hardware-level or OS-level cache managements. We don't need rely on the new design of future processors or modifying the kernel of OS. The contributions of this paper are three folds.

- 1) We gain insights into thread to discover weak and strong locality accessing patterns in analytical MMDB scenario;
- 2) Our approach provides an application software-based cache management which enables software to manage cache partition without OS supports;
- 3) The *W-order* scan is adaptive to table scan in MMDB and sequential accessing in other applications especially for the application with very large weak locality data that cannot be re-colored into limited page colors.

The rest of the paper is organized as follows. We analyze existing cache management in related work section. The *W-order* scan based cache partitioning mechanism is presented in section 3. Performance evaluation is presented in section 4, and we conclude the paper in section 5.

2 Related Work

Researches[4,5] discovered that single-usage(a cache block is accessed only once before getting evicted) blocks evict much reused data in shared cache. Upon the misses, cache blocks must be retrieved back from lower level which caused severe performance degradation. The wall facing researchers is the hard to be changed hardware-based LRU(*n-way set associative* is a combination of LRU and direct-map) cache algorithm.

The existing studies can be classified into three categories: 1)hardware(simulation)-based cache partitioning. 2)software(OS)-based cache partitioning. 3)hybrid (hardware support to coordinate with OS-based cache management) cache partitioning.

Hardware-based solutions[5,6,7,8,9,10] seem to be thorough and efficient solutions, but the complexity of altering the LRU replacement policy and cache line optimization is hard to be accepted by processor vendors, maybe some techniques will appear in future processors but nowadays applications cannot rely on the techniques evaluated upon simulations.

Page coloring[1,11,12,13] is a low-cost and high-speed address translation mechanism to map virtual/physical address pages to cache address pages. OS-based cache partitioning mechanisms assign cache regions by allocating or re-allocating physical pages based on page-color lists. [14] firstly proposed page-coloring based compiler and hardware approaches to eliminate conflict misses in physically addressed cache, this work is still on a simulated multi-core processor. [15] focused on how to exploit the hardware performance monitoring features of the microprocessor to aid determining the L2 partition size that should be given to each application. Then [2] extended [14] with a purely OS software based re-coloring scheme for both static and dynamic cache partitioning on multi-core processors. As a case study, [16] implemented the OS-based cache partitioning in data-intensive analytical database for query processing threads with different data locality patterns.

The great drawback of page-coloring is overhead of data movement from original colors to given colors. The memory space allocation always conflicts with application memory needs, and page re-coloring cache partitioning policy in a multi-programmed execution environment may incur substantial overhead. So [17] proposed a tradeoff policy of coloring on only a small set of frequently accessed (or hot) pages for each process which avoided conflicts among frequently accessed datasets of different processes. Compared with pre-assigned page-coloring policies, the hot page coloring policy can share more cache space for weak locality data and maintain required cache performance on hot pages. [18] exploited different locality types of data objects inside a thread or process via memory-trace sampling and partitioned the cache regions among data objects with a heuristic algorithm. The cache partition techniques develop from coarse granularity to fined granularity.

The hardware-based solutions have the best performance but must be simple and efficient enough to be acceptable for processor vendors, so complicated algorithms are impossible to be employed. OS-based methods can adapt to various application features but suffers from non-trivial software overhead. So [19] proposed a hybrid approach with a lightweight hardware mechanism for allocating cache blocks as well as providing information of cache usage and OS-based resource allocating policy for dynamic cache allocation. [19] designed a region mapping table to map a physical page to any cache color for minimizing page coloring overhead which thoroughly solve the key issue of OS-based methods. Whether this approach will be accepted by future processors is beyond our knowledge and we have to go on finding our solution which can be employed in practice.

Beyond the three categories, we can consider application software-based cache partitioning solution. Many applications have their native data locality patterns especially for DBMSs. For example, DBMS knows data locality patterns and how to achieve higher performance than OS does but has no capability to control cache partitioning. [16] tried to affect cache features by scheduling co-running different data locality type of queries with application knowledge, but the further application software-based

cache partitioning should go deep into directly managing physical address layout or coloring mechanism by DBMS itself.

3 Application Software-Based Cache Partitioning Mechanism

In this section, we first analyze the background and requirements, then discuss how to optimize in memory analytical processing by reducing cache conflicts.

3.1 Motivation

We limit our research of cache partitioning within analytical MMDB scenario for the reasons that:

- DRDB(Disk Resident Database) employs buffer pool to swap pages from disk to memory, so DBMS can control weak locality data to be assigned with few pages(colors) and strong locality data to be assign more pages(colors). But for MMDB, data are memory resident and directly accessed by CPU without buffer pool mechanism, how to control large weak locality data with full colors to be limited in specified colors is a challenging issue;
- Analytical MMDB commonly employs star-schema or snow-flake schema which can be divided into large single-usage fact table and small frequent-usage dimension tables. The weak locality and strong locality data are pre-defined by the schema;
- The execution plan of analytical queries is modeled by query optimizer, and DBMS knows how to use the weak and strong locality data during processing, so we can perform further optimized execution plan with cache partitioning optimizations;
- The last but the most important reason is that the cache buffer management is absent for MMDB. The sophisticated memory buffer management of DRDB cannot automatically upgrade to cache buffer management of MMDB due to the uncontrolled hardware-level LRU algorithm. MMDB tried to limit frequently accessed dataset within size of cache, but it did not work well for cache pollution.

3.2 Physical Address Layout for Cache and Main Memory

The *n-way set associative* algorithm is commonly employed in many multi-core processors. For example, the Intel Quad-core Xeon processor has two 4MB, 16-way set associative L2 caches for each two cores and the page size is set to 4KB. The well accepted OS technique called *page coloring*[1] describes how an memory page is mapped to cache region. Figure 2 illustrates physical address layout for this processor. The shared L2 cache is broken into 64 colors (cache size(4MB)/page size(4KB)/cache associativity(16)=64) and 16-way. So we can divide the whole L2 cache into 16×64 blocks. Let's look into the physical address of 4MB L2 cache. The 4MB L2 address comprises with 22 bits that the physical memory address is mapped into L2 cache address with the lower 22 bits. The upper 4 bits identify *n-way set associative* information, the next 6 bits represent *color* ID, the lower 12 bits denote 4KB addresses and the lowest 6 bits denote cache line offset. The physical main memory

address includes two parts: physical page number and page offset. The lowest 12 bits denote page offset for 4KB page. The lower 6 bits of physical page number represent *page color*. The adjacent upper 4 bits related to *set-associative* are neglected by hardware LRU policy of L2 cache.

We can consider the memory address and cache address as two matrixes, each matrix comprises with 64 columns. Cache matrix has 16 rows and memory matrix has N (decided by memory capacity) rows. Blocks in same column belong to same page color. We call blocks in same color of L2 cache as cache region. The hardware cache algorithm pulls page from memory to cache region with the same page color.

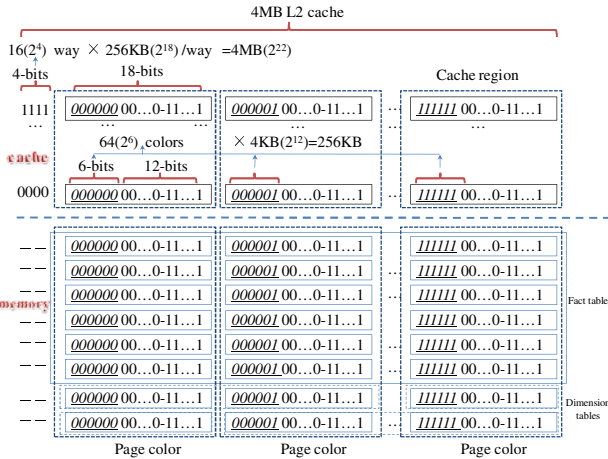


Fig. 2. Physical address layout for cache and main memory

3.3 W-Order Scan for Main Memory Database

For static page-coloring policy, memory space is allocated for strong and weak locality data through color list to pre-assign cache regions for the two datasets. So the cache conflicts between the two types of dataset can be eliminated. Compared with the whole dataset, the shared L2 cache is far smaller than memory. The ideal scenario is that the whole L2 cache is allocated for strong locality data and the weak locality data leave no footprints in L2 cache and can be directly passed to L1 cache. But it is hardly to realize and the practical approach is to assign as less as possible colors for weak locality data and assign as much as possible colors for strong locality data. Each color denotes $1/n$ (n denotes the amount of colors) memory capacity, it is impossible to allocate $1/n$ memory space for more than 80% weak locality data in memory.

Figure 3 illustrates the address layout for memory and cache. Sequential scan on consecutive addresses follows the *Z-order*(the red dashed lines) i.e. accessing page group from page-color 1 to page-color 64 and then accessing next group till last page. The *Z-order* scan forces the cached strong locality data to be evicted sequentially which is called cache pollution. We re-arrange the page scanning order as *W-order*(the blue real lines) in figure 3 which performs scan by page-color order. All the sequentially scanned pages are naturally divided into groups based on page-color.

During scanning in each group, the sequentially scanned data swap from memory to cache region with same color, and only strong locality data with the same color may produce cache conflicts with the sequentially scanned data. So in each *W-order* scan group, cache pollution domain is limited in $1/n$ cache region.

The page coloring partitions the cache by optimizing the layout of in-memory data, it is suitable for small size of weak locality dataset which can be further allocated in specified memory address space(page color). *W-order* scan focuses on large weak locality dataset by optimizing the block accessing order for sequential in-memory data which is suitable for the large fact table of main-memory analytical database.

For weak locality random scan such as index scan, *W-order* scan has no chance to optimize because the accessing order cannot be re-organized. If the memory resident base table is large, the random scan on it also covers large proportion of page colors which is beyond the control of page-coloring. The page-coloring can optimizes index scan for DRDB by mapping buffered page of index scan to specified colors.

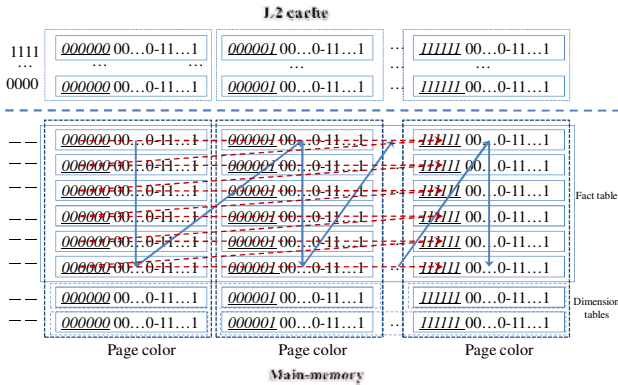


Fig. 3. Z-order and W-order scan

For summary, *W-order* scan is an adaptive light-weight cache partitioning mechanism for sequential scan on large memory resident weak locality dataset.

3.4 The Realization of W-Order Scan

The key of *W-order* scan is to control the accessing order for memory resident pages.

In real system, the consecutive data cannot acquiescently allocate large consecutive physical addresses. Furthermore, in-memory data are addressed by virtual addresses which need virtual-to-physical address translation to support *W-order* scan. So we propose two solutions to support *W-order* scan based cache partitioning.

(1) *kmalloc* based W-order scan

The *kmalloc* function can allocate physical consecutive addresses for application. The default maximal block size is 128KB. We can allocate maximum 128MB consecutive physical addresses with *kmalloc* function in our work by re-compiling Linux kernel

with modified parameter. So we can manage fact table address space with a 128MB block list, and each 128MB block is allocated by *kmalloc* function. The limitation is that the free large physical consecutive address space is limited and we can only allocate total 1GB blocks of our 4GB experimental platform. The memory utilization rate is limited by memory management module of Linux kernel.

In physical consecutive address block, page colors are continuous for each address segment, and we can perform the *W-order* scan by controlling the page accessing order with on-the-fly address calculating.

(2) page-color index based *W-order* scan

Virtual addresses have to be translated into physical addresses to be used for cache swapping. We propose a page-color index to construct mappings from virtual pages to page-colors. We define page-color index as:

PC-index <page entry address, page-color key>, where:

page entry address: virtual address of page.

page-color key: color bits abstracted from virtual-to-physical address translation.

As all the weak locality data pages are accessed to create PC-index, we get a binary table for virtual page address to page-color mapping. After ordering by page-color key, the virtual pages are grouped by page-color. The PC-index is the linear representation for *W-order* scan with virtual addressed pages. The PC-index needs no modification of OS or re-compiling the OS for *kmalloc* function support. The PC-index should be pre-generated after data are re-loaded into main-memory, and the OS must guarantee the virtual address of pages not to be swapped out. We close swap function by “*swapoff*” command and configure the dataset no more than physical memory size to keep memory pages to be constant.

kmalloc based *W-order* scan is a physical page-color order while the PC-index is a logical virtual page-color order. The key point of the two methods is that application software can control cache conflicting by re-organizing the page accessing order.

4 Experiments

4.1 Design of Experiments

L2 cache is shared for data and instructions, the *n-way associative* cache acts as a Least-Recently Used(LRU) stack for cache and the processors which normally uses an approximation mechanism of LRU algorithm to replace cache blocks, so it is hard to accurately predict the cache missing ratio for specified program. We employ PAPI to measure cache performance. PAPI is a software layer(library) designed to provide the tool developer and application engineer for use of the performance counter hardware found in most major micro-processors. We use four generic performance counter events in our experiments: PAPI_TOT_CYC(Total cycles), PAPI_TOT_INS(Total instructions), PAPI_L2_TCA(Level 2 cache accesses), and PAPI_L2_TCM(Level 2 cache misses). The L2 cache miss rate can be calculated as $L2_cache_miss_rate = PAPI_L2_TCM / PAPI_L2_TCA$ and CPI(Cycles Per Instruction) can be calculated

by $CPI = PAPI_TOT_CYC / PAPI_TOT_INS$. We use the metric $PAPI_TOT_CYC$ to measure execution time, $L2_cache_miss_rate$ to measure conflicts in L2 cache, and CPI to measure performance of program.

We simulate an in-memory hash join processing with one fact table and four dimension tables. The query is described as “SELECT count(*) FROM $F, D1, D2, D3, D4$ WHERE $F.d1=D1.pk$ and $F.d2=D2.pk$ and $F.d3=D3.pk$ and $F.d4=D4.pk$ and ...;”. During query processing, fact table is sequentially scanned and performs hash-join with four hash tables generated by four dimension tables. We alter the size of hash tables to measure the effects of hash table size for W -order scan. We simplify the hash join algorithm by directly employing primary key of dimension table as hash key and ignore group-by and order-by operations to measure how the cache partitioning mechanism accurately affects the most costly join procedure.

We measure three algorithms in different scenarios, the algorithms are designed as:

(1) Hash-join with sequential scan

Sequential scan represents the commonly used full table scan which follows the accessing order by virtual-address sequence. Virtual memory is allocated by 4KB page, each page is specified to fixed color according to color bits in physical address. The addresses are consecutive inside the page but the adjacent virtual pages may not have adjacent page color bits.

(2) Hash-join with physical consecutive W -order scan

We organize the fact table in physical consecutive blocks allocated from OS by *kmal-loc* function. We can allocate page segment by on-the-fly address calculating. The physical consecutive blocks are organized as block list. During W -order scan, we first scan 4KB segments with color bits 0 in each block and then scan segments in all blocks with next page color bits.

(3) Hash-join with page-color index W -order scan

Page-color index scan is developed directly on the common virtual-address sequential dataset. The page-color bits for each default page (4KB) are extracted to create index for identifying color of each page. The sorted page-color index on page-color bits shows a nature W -order for scan.

4.2 Platform and Datasets

All our experiments are conducted on an HP workstation with Intel Core2 Extreme X9100 processor, there are two cores of 3.06GHz EX64T. Each core has a 32KB 8-way associative instruction cache with 64-byte cache line and a 32KB 8-way associative data cache with 64-byte cache line. The dual-core processor shares a 6MB 24-way associative L2 cache with 64-byte cache line. The workstation is configured with 4GB DD2 memory and 80GB hard disk. The OS version is Ubuntu Linux SMP 2.6.31.6.

We generate star schema style datasets for performance measuring. The fact tuple is 16-byte wide and hash item width is 64-byte wide. The ID attributes in fact table

are generated with normal pseudo-random numbers related with hash keys in each hash tables. Other attributes are all generated by random numbers.

The fact table increases with step of 2MB till the maximal 1GB size. The four hash tables are designed with different sizes of 1MB, 2MB, 4MB, 8MB.

For physical consecutive block based *W-order* scan algorithm, we adjust system parameter to allocate physical consecutive blocks with block size varies from 1MB, 2MB, 4MB till 128MB(2^n).

4.3 Performance Evaluation

The following figures show the improvements for *W-order* scan and page-color index scan vs. sequential scan. We measure the cache miss rates with 1MB, 2MB, 4MB, 8MB, 16MB, 32MB, 64MB and 128MB physical consecutive blocks, we find that as the block size increases cache miss rate improvement degrades especially when size of dimension hash tables(8MB) exceeds cache size(6MB), larger physical consecutive blocks(larger than 32MB) degrades cache miss rate remarkably.

Larger block means less address calculating with shorter block list. But large block allocated by *kmalloc* produces more micro operations in kernel level. As we have no detailed information about how *kmalloc* function assigns physical blocks in buddy system of Linux kernel, the observation is that *W-order* scan with smaller physical consecutive block outperforms that of larger block. Because the default *kmalloc* block size is only 128KB, we enforce OS to assign large block which may produce more inner address computing during accessing.

Figure 4 and figure 5 illustrate the L2 cache miss rate improvements of *W-order* scan with physical consecutive block and page-color index scan. We can observe that: (1) *W-order* Block scan with small block size(minimum 1MB in our experiments) outperforms those with larger block size; (2) *W-order* Block scan outperforms PCI-scan in average due to higher coding efficiency; (3) the trend of PCI-scan improvements follows the pattern of sequential scan when dimension hash table size increases while *W-order* Block scan proves a larger improvements.

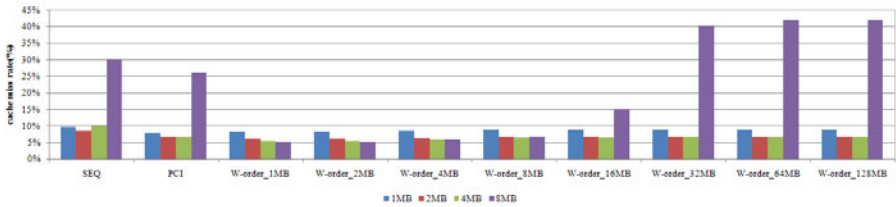


Fig. 4. L2 cache miss rate for *W-order* scan

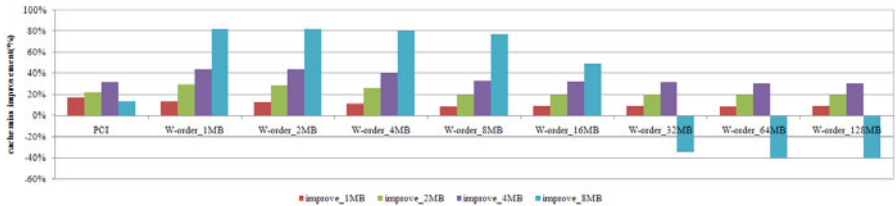


Fig. 5. L2 cache miss rate improvements for *W-order* scan

Figure 6 and figure 7 illustrates the execution cost improvements with metric of execution cycles. Comparing with figure 4 and figure 5, we can see that the overall improvements are below them. *W-order* scan optimizes cache partitioning performance by reducing L2 cache misses, but at the same time more CPU cycles are consumed for the additional instructions for page order control. The improvement trend follows the trend of L2 cache miss rate improvement with small gap for additional instruction cost.

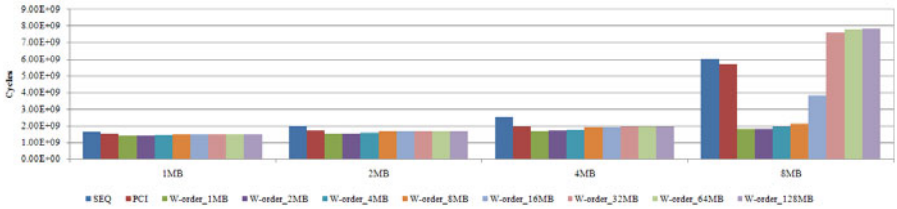


Fig. 6. Execution cycles for *W-order* scan



Fig. 7. Execution cycles improvements for *W-order* scan

For shared last level cache architecture in multi-core platform, the cache size and cache conflicting control are key issues for performance improvements. The hardware leaves no more space to achieve these targets: (1) Reducing strong locality dataset size to increase possibility of resident. (2) Controlling and optimizing cache pollution to reduce cache misses. Our approaches in this paper states *W-order* scan to optimize cache pollution in application software level which breaks the dependence from hardware of OS. Considering the page limitation, we didn't mention our research for multi-core parallel cache optimization, and the further experiments on complete benchmark(e.g. SSB) will appear in our future work. This paper focuses on a model study with single representative DBMS operation case without interferences of non-critical factors.

5 Conclusion

The LLC is critical for computer hierarchy to overlap latency gap between main memory and processors. There are various data patterns in applications, in which the most representational patterns are single-usage pattern and frequent-usage pattern especially for analytical MMDB application, in which the frequent-usage data are

continuously evicted by the new coming single-usage data by LRU algorithm. Our proposal in this paper isolates single-usage data(weak locality data) and frequent-usage data(strong locality data) by page-color based optimizations. The most important contribution of our work lies in that our cache management is application software based without depending on modified processor or OS as most of the existing researches mentioned. Another considerable contribution is that the memory page accessing order control mechanism can be extended as cache buffer mechanism for analytical MMDB which optimizes data accessing pattern with physical address granularity of page-color which can be managed upon the OS.

State-of-the-art hardware techniques support more and more main-memory(updated from GB to TB level) and larger LLC(e.g. Intel® Itanium® processor 9350 provides 4 cores with 24MB L3 cache). The key to break memory wall is how to efficiently manage the LLC to minimize cache misses. Our proposal of application software base cache partitioning fills the blank left by hardware- and OS-based approaches.

Acknowledgements. This work was supported by the National High Technology Research and Development Program of China (2009AA01Z149), the Major National Science and Technology Project of China (2010ZX01042-001-002-002), the projects of Grant No.61070054,10XNI018,10XNB053,10XNH096 and 11XNH120.

References

1. Taylor, G., Davies, P., Farnwald, M.: The TLB slice-a low-cost high-speed address translation mechanism. In: Proceedings of the ISCA 1990, pp. 355–363 (1990)
2. Lin, J., Lu, Q., Ding, X., Zhang, Z., Zhang, X., Sadayappan, P.: Gaining insights into multi-core cache partitioning: Bridging the gap between simulation and real systems. In: Proceedings of the 14th International Symposium on High-Performance Computer Architecture, pp. 367–378 (2008)
3. Piquet, T., Rochecouste, O., Sez nec, A.: Minimizing Single-Usage Cache Pollution for Effective Cache Hierarchy Management. Research report (2006), <http://hal.inria.fr/docs/00/11/66/11/PDF/PI-1826.pdf>
4. Piquet, T., Rochecouste, O., Sez nec, A.: Exploiting Single-Usage for Effective Memory Management. In: Proceedings of Asia-Pacific Computer Systems Architecture Conference, pp. 90–101 (2007)
5. Chang, J., Sohi, G.S.: Cooperative cache partitioning for chip multiprocessors. In: Proceedings of the 21st Annual International Conference on Supercomputing, pp. 242–252 (2007)
6. Kim, S., Chandra, D., Solihin, Y.: Fair cache sharing and partitioning in a chip multiprocessor architecture. In: Proceedings of the 13th International Conference on Parallel Architectures and Compilation Techniques, pp. 111–122 (2004)
7. Qureshi, M.K., Patt, Y.N.: Utility-based cache partitioning: A low-overhead, high-performance, runtime mechanism to partition shared caches. In: Proceedings of the 39th International Symposium on Microarchitecture, pp. 423–432 (2006)
8. Suh, G.E., Rudolph, L., Devadas, S.: Dynamic partitioning of shared cache memory. The Journal of Supercomputing 28(1), 7–26 (2004)

9. Chandra, D., Guo, F., Kim, S., Solihin, Y.: Predicting inter-thread cache contention on a chip multi-processor architecture. In: Proceedings of HPCA (2005)
10. Guo, F., Solihin, Y.: An analytical model for cache replacement policy performance. In: Proceedings of SIGMETRICS (2006)
11. Bray, B.K., Lynch, W.L., Flynn, M.: Page allocation to reduce access time of physical caches. Tech. Rep. CSL-TR-90-454, Computer Systems Laboratory, Stanford University (1990)
12. Taylor, G., Davies, P., Farmwald, M.: The TLB slice – A low-cost high-speed address translation mechanism. In: Proceedings of the 17th Annual International Symposium on Computer Architecture, pp. 355–363 (1990)
13. Kessler, R.E., Hill, M.D.: Page placement algorithms for large real-indexed caches. *ACM Transactions on Computer Systems* 10(11), 338–359 (1992)
14. Sherwood, T., Calder, B., Emer, J.: Reducing cache misses using hardware and software page placement. In: Proceedings of the 13th International Conference on Supercomputing, pp. 155–164 (1999)
15. Tam, D., Azimi, R., Soares, L., Stumm, M.: Managing shared L2 caches on multicore systems in software. In: Proceedings of WIOSCA 2007 (2007)
16. Lee, R., Ding, X., Chen, F., Lu, Q., Zhang, X.: MCC-DB: Minimizing Cache Conflicts in Multi-core Processors for Databases. *PVLDB* 2(1), 373–384 (2009)
17. Zhang, X., Dwarkadas, S., Shen, K.: Towards practical page coloring-based multicore cache management. In: EuroSys 2009, pp. 89–102 (2009)
18. Lu, Q., Lin, J., Ding, X., Zhang, Z., Zhang, X., Sadayappan, P.: Soft-OLP: Improving Hardware Cache Performance through Software-Controlled Object-Level Partitioning. In: Proceedings of PACT 2009, pp. 246–257 (2009)
19. Lin, J., Lu, Q., Ding, X., Zhang, Z., Zhang, X., Sadayappan, P.: Enabling software management for multicore caches with a lightweight hardware support. In: Proceedings of SC 2009 (2009)
20. Boncz, P.A., Mangegold, S., Kersten, M.L.: Database architecture optimized for the new bottleneck: Memory access. In: Proceedings of the VLDB 1999, pp. 266–277 (1999)

Index Structure for Cross-Class Query in Object Deputy Database

Yuwei Peng^{1,*}, Hefei Ge¹, Mao Ding¹, Zeqian Huang², and Chuanjian Wang¹

¹ Computer School of Wuhan University, Wuhan, China

² State Key Lab of Software Engineering, Wuhan University, Wuhan, China
ywpeng@whu.edu.cn

Abstract. This paper proposes an index structure for Object Deputy Database, which is named as 'Object Deputy Path Index (ODPI)'. ODPI can reduce the cost of evaluating path expressions, which is the most important process of Cross-Class Query in Object Deputy Database. Object Deputy Path Index can materialize path instances in the index structure, avoiding redundant object traversal in query processing. This paper also introduces a maintenance method for ODPI. The experiments analyze the influential factors of path expression evaluation, and the experimental results demonstrate that path expression evaluation with ODPI outperforms the other methods in most cases.

Keywords: index, object deputy database, cross-class query, path expression evaluation, optimization.

1 Introduction

Object-oriented (OO) data model has solved the problem of modeling complex data, which is difficult to the traditional relational data model. Although OO data model can model complex data directly and provide rich semantics, it does not have the flexibility like relational data model. In addition, OO data model is lack of ability to model multi-roles and dynamic of real world entities. Object deputy model [1, 2] is a new data model based on traditional OO data model. It uses concept of deputy object to improve flexibility and modeling ability of OO data model. Object deputy model has been widely used in data warehouse [3], workflow view [4], scientific workflow [5], biological data management [6] and GIS [7]. Database techniques and object deputy model result in object deputy database (ODDB). Besides features of OODB, ODDB allows users to create flexible object views and supports dynamic classification of objects. It also provides a special query method named as Cross-Class Query.

* This work is supported by the Major State Basic Research Development Program of China under Grant No. 2007CB310806, the Major State Research Program of National Natural Science Foundation of China under Grant No. 90718027, the Major Program of Natural Science Foundation of Hubei Province under Grant No. 2008CDA007, and the Fundamental Research Funds for the Central Universities under Grant No. 6082011.

Practice has proved that OODB is more efficient than OODB in complex data management applications such as spatial databases and multimedia databases.

Object deputy query language (ODQL), a declarative query language, is proposed for OODB in [8]. This language has some new features such as deputy object query and Cross-Class Query. As two main queries of OODB, their performance is worthy of attention. Cross-Class Query of OODB is realized by path expression. Starting from objects of one class (deputy class), Cross-Class Query will allow users to get objects of class which has direct or indirect deputy relationship with the head class along bilateral points between these objects. Obviously, optimization of path expression evaluation is very important to OODB's performance.

This paper proposes a new index structure, Object Deputy Path Index (ODPI), for path expression evaluation in OODB. Experiments proves that ODPI can support path expression evaluation efficiently. The remainder of this paper is organized as follows. Section 2 describes related works. Section 3 introduces object deputy database. Object Deputy Path Index is described in section 4. Section 4 also gives method for computing path expression with ODPI. A serious experiments are given in section 5 to prove the efficiency of ODPI. Section 6 gives conclusions and challenges.

2 Related Works

The concept of path expression has been already proposed in query language of OODB [9] [10]. It's used for navigation between nesting objects. Currently, there are three methods to compute path expression: forward pointer tracking algorithm, reverse pointer tracking algorithm and explicit join algorithm. Literature [11] proposed a cost model to evaluate computation cost of different algorithms. And heuristic rules are employed to choose a combination evaluation strategy which has great performance. Unlike the three algorithms described above, literature [12] proposed a parallel algorithm for path expression evaluation. This algorithm converts a path expression to a cascade half-join expression and computes it. Path expression of OODB is quite different from the one of OODB. The former is not used to traverse between nesting objects but to navigate between objects which have deputy relationship. These two kind of path expressions have different evaluation methods.

Index techniques are widely used to support and optimize nesting object query in OODB. Literature [13] proposed Multiple Index built on any two classes which have reference relationship. Multiple Index is used to map referenced object to reference object. Join Index proposed in [14] is similar to Multiple Index, except that there are two mapping between referenced and reference objects in Join Index. Nested Index [15] is designed to support nesting predication evaluation. It enables users to map a value of nesting attribute to the root object. Path Index [16] is more complex than Nested Index. Its index item stores not only root object but also every objects on the nesting path. Path Dictionary [17] proposed an index mechanism based on path dictionary to support objects traversal and

related query. Access Support Relation proposed in [18] is a normalized Join Index. Such index techniques used in OODB are not fit for path expression of ODDB. And until now, there is no research on index techniques for path expression evaluation in ODDB.

3 Object Deputy Database

Object Deputy Database contains the core concepts: object, deputy object, class and deputy class, etc. The detailed definition of these concepts are described in [1]. This section will only give some definitions for path expression in ODDB.

3.1 Concepts for Path Expression

Definition 1. *Directed Graph composed by deputy relationship between classes (deputy classes) is called 'deputy level graph'. Each node in deputy level graph is a class or a deputy class. Directed edge means direct deputy relationship between two nodes. Let DH be a deputy level graph, $Class(DH)$ is a set of all classes and deputy classes in DH :*

$$Class(DH) = \{C|C \text{ is class or deputy class in } DH\}$$

Definition 2. *Given a deputy level graph DH , let $P = C_1 \rightarrow C_2 \rightarrow \dots \rightarrow C_n (n \geq 2)$. P is a path of DH if and only if P satisfies the following two conditions:*

1. $C_i \in Class(DH), 1 \leq i \leq n$;
2. C_i has direct deputy relationship with C_{i+1} , that is to say C_i is a deputy class of C_{i+1} or vice versa.

$len(P) = n - 1$ is length of Path P . $Class(P) = \{C|C \text{ in } P\}$ is a set of classes and deputy classes contained by path P . $P_1 = C_1 \rightarrow C_2 \rightarrow \dots \rightarrow C_n$ and $P_2 = C_n \rightarrow C_{n-1} \rightarrow \dots \rightarrow C_1$ are each other's reverse path.

Definition 3. *Given a path $P = C_1 \rightarrow C_2 \rightarrow \dots \rightarrow C_n$, let $p_i = o_1 \rightarrow o_2 \rightarrow \dots \rightarrow o_n$. p_i is an instance of P if and only if p_i satisfies the following conditions:*

1. o_i is an instance of $C_i, 1 \leq i \leq n$;
2. o_i has direct deputy relationship with o_{i+1} , that is to say o_i is a deputy object of o_{i+1} or vice versa ($1 \leq i < n$).

We name o_1 as 'start-point object' and o_n as 'end-point object'.

Definition 4. *Given a deputy level graph DH , $PE = (C_1\{pr_1\} \rightarrow C_2\{pr_2\} \rightarrow \dots \rightarrow C_n\{pr_n\}).expr (n \geq 2)$ is a path expression defined on path $P = C_1 \rightarrow C_2 \rightarrow \dots \rightarrow C_n$ if and only if the following conditions are satisfied:*

1. P is a path of DH ;
2. pr_i is a predication defined on attribute of C_i (pr_i can be NULL, $1 \leq i \leq n$);
3. $expr$ is a expression composed by constant and attributes of C_n .

Among them, $C_1\{pr_1\} \rightarrow C_2\{pr_2\} \rightarrow \dots \rightarrow C_n\{pr_n\}$ is called navigation path of PE, $expr$ is called target expression of PE.

We use a music database to give some examples of definitions described above. Figure 1 shows schema of the music database. It contains five classes: **Artist**, **Music**, **MTV**, **Lyrics** and **Picture**. There are also five deputy classes. **Personal** is a select deputy class on **Music**. **Music_Lyr**, **Music_MTV** and **Album_Pic** are all join deputy classes. **Album** is a group deputy class on **Music_Lyr**. We can say that schema in Fig 1 is a deputy level graph. And $P = Music \rightarrow Music_Lyr \rightarrow Album$ is a path of that deputy level graph. ($Music\{artist = 'U2'\} \rightarrow Music_Lyr \rightarrow Album$).*intro* is a path expression defined on path P.

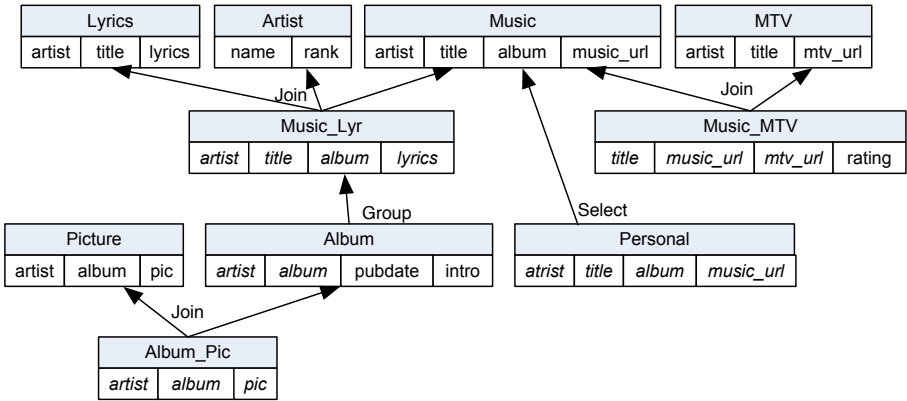


Fig. 1. Schema of Music Database

3.2 Path Expression Evaluation

Path expression evaluation is the key of query process in ODDB for it is core of deputy class query and Cross-Class Query. Path expression evaluation is depend on bilateral links between objects. Each objects has links point to its deputy objects and source objects. Bilateral links describe deputy relationship between objects. It is easily to get deputy objects or source objects for a given object according to bilateral links. In current implementation, a OID mapping table is used in ODDB to store bilateral links between objects. Bilateral links are maintained by DBMS automatically. The OID mapping table is a binary relation:

$$[SourceObject:OID, DeputyObject:OID]$$

SourceObject is source object's OID. *DeputyObject* is deputy object's OID.

There are two algorithms to evaluate path expression: pointer tracking algorithm and explicit join algorithm.

In pointer tracking algorithm, each object of the head class C_1 on the path will be accessed firstly. It will be checked whether it will satisfy predication p_1

of C_1 . For every object o_1 satisfied p_1 , getting next object o_2 on path instance according to o_1 's bilateral links. And then check whether o_2 will satisfy p_2 . This tracking process will continue until:

1. There is no object found according to current object's bilateral links.
2. Current object can not satisfy the predication.
3. Current object is an object of the tail class on the path.

Explicit join algorithm converts a path to an explicit join expression. By evaluating this expression, we can get all path instances satisfied all predications. Then projection can be done on these instances to get target expressions. If there is no predication on the path, we should only need to join the head class, the tail class and OID mapping table together to get result. Otherwise, we should add classes which have predications on them into the join operation. So n times self-join operations on OID mapping table are needed for the path which has n classes.

4 Object Deputy Path Index

This section introduces an index structure supporting path expression evaluation: Object Deputy Path Index (ODPI). Under the help of traditional index built on head or tail class of the path, ODPI can support path expression evaluation with predications on head and tail classes. ODPI can avoid the cost of objects traversal and predications checking in path expression evaluation.

4.1 Structure of ODPI

The idea of ODPI is very simple: materializing path instances. ODPI is different from traditional indices. Traditional indices use a key value to search the index tree and return pointers of result objects. Generally, the index item of traditional index is a pair: $\langle value, object's\ pointer \rangle$. But for ODPI, start-point objects are used to retrieve end-point objects. In order to get end-point objects according to start-point objects quickly, we need to design special index item for ODPI.

Each index item of ODPI is corresponding to a path instance. But we need not store all objects of the instance in the index item. In order to decrease storage cost, we should store the most necessary information in it. These information includes start-point object's OID, pointer of start-point object, end-point object's OID and pointer of end-point object. So an index item of ODPI looks like this:

$$\langle o_s.OID, o_s.pointer, o_e.OID, o_e.pointer \rangle$$

(o_s is start-point object, o_e is end-point object)

This design of index item enables sharing a same ODPI between a path and its reverse path.

Although ODPI's index item is much different from traditional indices, we can use classic B-Tree to organize these index items. Each ODPI has two B-Tree structures: Forward Index and Reverse Index. Forward index uses start-point object's OID as index key and reverse index uses end-point object's OID.

Obviously forward index and reverse index are designed for path and its reverse path respectively. So the pointers in forward index and reverse index are end-point object's pointer and start-point object's pointer respectively.

Besides index item, we still need to consider three issues for ODPI: creation, using and maintenance.

4.2 Creation of ODPI

Current ODQL provide CREATE INDEX to create indices on classes or deputy classes. But it does not support create indices on path. So we extend CREATE INDEX like this:

CREATE INDEX < index_name > ON < path >

Index_name refers to name of ODPI.

In order to save meta data of ODPI indices, we design a system catalog *od_pathindex* (Table 1).

Table 1. Attributes of *od_pathindex*

Name	Type	Notation
<i>startclassoid</i>	OID	OID of the head class on the path
<i>endclassoid</i>	OID	OID of the end class on the path
<i>seqoids</i>	OID Array	OID sequence consists of all classes on the path
<i>indexoid</i>	OID	OID of the ODPI index

In ODDB, it might exist more than one path between two classes. Storing only the head and tail classes is not enough to identify a path. So we need attribute *seqoids* to saving a OID sequence consisted of all classes on the path. For a given path, we filter *od_pathindex* by *startclassoid* and *endclassoid*. If there is only one path remained, the index identified by *indexoid* can be used. Otherwise there are more than one tuple remained, we need *seqoids* to identify the index. Certainly, *startclassoid* and *endclassoid* are not necessary for ODPI identification. But with these two attributes, the searching process can be more efficient.

ODPI creation on path $P = C_1 \rightarrow C_2 \rightarrow \dots \rightarrow C_n$ runs as following steps:

1. Check whether P is a valid path.
2. Initialize forward index FI and reverse index RI.
3. Use pointer tracking algorithm to find all path instances of path P.
4. For each path instance like $o_1 \rightarrow o_2 \rightarrow \dots \rightarrow o_n$ got in Step 1, construct an index item and insert it into forward index and reverse index.
5. Build two tuples of *od_pathindex*:
 $\langle C_1.OID, C_n.OID, \{C_1.OID, \dots, C_n.OID\}, FI.OID \rangle$
and $\langle C_n.OID, C_1.OID, \{C_n.OID, \dots, C_1.OID\}, RI.OID \rangle$. Insert these two tuples into *od_pathindex*.

4.3 Using ODPI

After creation of ODPI index, we can use this index to evaluate path expressions. For a given path expression, we should firstly check whether it has a valid path. Then `od_pathindex` and OIDs of classes on the path are used to search a valid ODPI index. If there was a valid index, it can be used to complete the evaluation. The whole process of path expression evaluation using ODPI is runs as follow steps:

1. Search a valid index in `od_pathindex` according to OIDs of classes on the path. If there is a valid index then go to Step 2. Otherwise go to step 4.
2. Get all start-point objects satisfied predication on head class (If there are traditional indices on the predication attributes, use them). Use OIDs of these objects to search end-point objects in ODPI index got in Step 1. OIDs and pointers of end-point objects will lead us to end-point objects. Filter out the end-point objects which didn't satisfy the predication on tail class.
3. Execute projection on remaining end-point objects and return the result objects.
4. Use pointer tracking algorithm to evaluate the path expression.

In steps mentioned above, predications on head and tail classes are considered. So path expression evaluation based on ODPI can support path expressions with predications on head or tail class.

4.4 Maintenance of ODPI

Like traditional indices, ODPI needs to be maintained after update operations in ODDB. There are two types of update operations affected ODPI: deletion of classes on path and updating objects of classes on path.

A path consists of classes. If any class on a path is dropped, the path itself can not exist any more. And then the ODPI on this path becomes invalid. So we must examine the index when dropping class is in progress. It's worth noting that propagation mechanism of ODDB can cascade drop related classes. This means that if we want to drop a class then its deputy classes would be all dropped. We must examine the index for each dropping operation.

Updating objects of classes on path may create new path instances or delete some old path instances. Correspondingly we need to insert or delete index items of these path instances. It's a complex process. This section mainly introduces this process.

In order to maintain ODPI, we should save temp information about changing of objects and bilateral links between them. These information can be used to find out path instances have been created and deleted. Further more, the corresponding index items will be inserted into or deleted from the index. For propagation mechanism processes updating objects one by one, we will do index maintenance after each object's propagation finished.

The maintenance operation of ODPI includes four steps:

1. Create Objects Propagation Graph (OPG). Objects Propagation Graph describes relationship between updating object and its related objects. In this OPG, objects are nodes and their deputy relationship is edge. Objects have been created or deleted in propagation are all remained in OPG. New nodes will be labelled by symbol '+' and deleted objects will be labelled by symbol '-'. Edges without changes, new edges and deleted edges will be labelled by '0', '1' and '-1' respectively. Figure.2 shows an example of OPG.

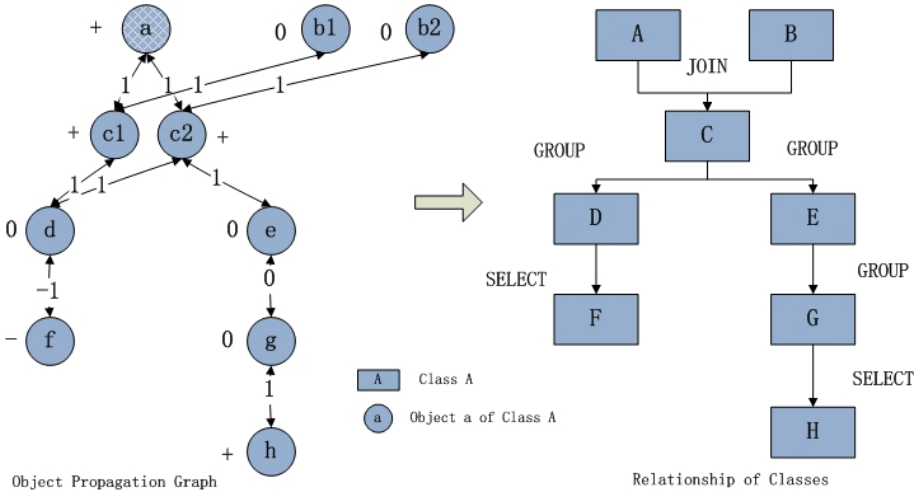


Fig. 2. Example for OPG

2. Find out ODPI indices influenced by this update operation. For each path in od_pathindex, check whether it contains any class appeared in OPG. If the path has no class appeared in OPG, indices on this path do not be influenced. Indices must be maintained if any one of the following conditions is satisfied:
 - (a) Head class appears in OPG but tail class does not.
 - (b) Tail class appears in OPG but head class does not.
 - (c) Head class and tail class all appear in OPG.
 - (d) Head class and tail class don't appear in OPG but part classes of the path do.
3. Search path instances created or deleted in OPG. For each index influenced, check all path instances of their path:
 - (a) If a path instance only has '0' labelled on its edges, this path instance has never been changed. So its index item should not be update.
 - (b) If a path instance has one or more '-' and '-1' labelled on its nodes and edges, this path instance has already been deleted. So its index item should be deleted from the index.

- (c) If a path instance only has '+' , '0' or '1' labelled on its nodes and edges, this pat instance is new. So its index item should be inserted into the index.
4. Maintain the indices. According to the result of Step 3, insert or delete corresponding index items.

5 Performance Evaluation

ODPI has been implemented in TOTEM, a ODDDB developed by our group. This section gives several experiments to prove efficiency of ODPI and analyze factors influence the performance. All experiments are finished in TOTEM.

The experiments are carried out on a PC of Intel Celeron 2.0 GHz CPU, 2GB RAM, 200GB hard disk, Ubuntu 9.10 and TOTEM 2.0. A music database is used in all experiments, its schema is described in Fig.1. Different size of data sets (DS1 to DS6) are used in experiments. Table 2 describes the size of each data set.

Table 2. Number of objects

	Artist	Music	MTV	Lyrics	Picture	Album	Music_Lyr	Music_MTV	Album_Pic
DS1	500	5000	5000	5000	1000	1000	5000	5000	1000
DS2	1000	10000	10000	10000	2000	2000	10000	10000	2000
DS3	2000	20000	20000	20000	4000	4000	20000	20000	4000
DS4	3000	30000	30000	30000	6000	6000	30000	30000	6000
DS5	5000	50000	50000	50000	10000	10000	50000	50000	10000
DS6	8000	80000	80000	80000	16000	16000	80000	80000	16000

Currently, TOTEM uses pointer tracking algorithm(PT) introduced in section 2.3 to evaluate path expressions. In this section, we compare ODPI with PT. We also implemented Multiple Join Index(MJ-Index), Nested Index(N-Index) and Path Index(P-Index) according to the algorithms introduced in section 2.3. These implementations are used in experiments to compare with our ODPI.

5.1 Exp-1: Varying Length of Path without Predications

The first set of experiments evaluate performance of ODPI, PT, MJ-Index, N-Index and P-Index with different length of path without predications and different data sets. Two no-predication path expressions whose length is 3 and 6 respectively are used in these experiments. And the response time of all path expression queries are recorded (Fig.3 and Fig.4). We can see that the response time of every index increases with the size of data sets and the length of path. Performance of PT is lowest in these methods. Because PT needs huge number of I/O operations to access objects used in evaluation. MJ-Index is more efficient to short path expressions. But it is not suitable for long path expressions, because several join operations are needed. Performance of ODPI is similar to the one of P-Index. These two indices all materialized the path instances in their

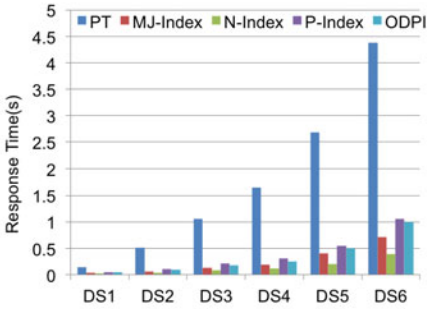


Fig. 3. Exp-1: Length=3

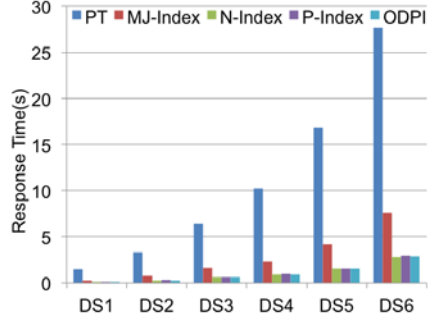


Fig. 4. Exp-1: Length=6

structures. But they all performed worse than N-Index. N-Index only stores head object and tail object for each path instance. So number of objects accessed by N-Index is smallest.

5.2 Exp-2: Fixed Length of Path with Predications

The second set of experiments evaluate performance of these indices with predications in path expression. N-Index does not support predications in path expression, so only ODPI, PT, MJ-Index and P-Index are used in these experiments. Path expressions with one predication on real attribute, one predication on virtual attribute and two predications on virtual attributes respectively are used. The last one has two predications on virtual attributes of its head and tail classes respectively. Figure 5, Fig 6 and Fig 7 show response time of these indices. We can see that ODPI performed best in four indices. This owes to traditional index on head or tail class of the path. It can avoid predication computation in path expression evaluation. ODPI performed more better with path expressions with

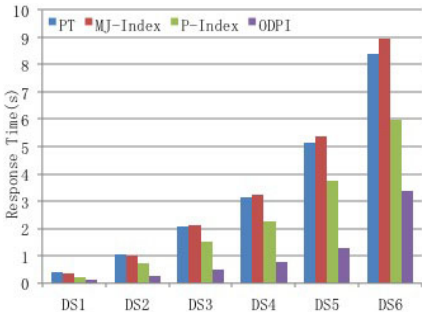


Fig. 5. Exp-2: One Predication on Real Attribute

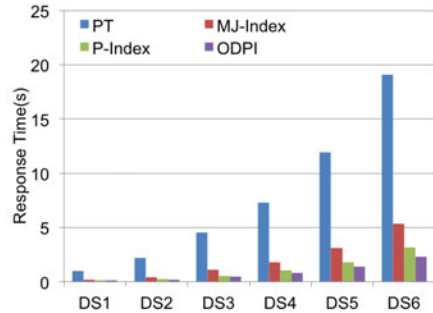


Fig. 6. Exp-2: One Predication on Virtual Attribute

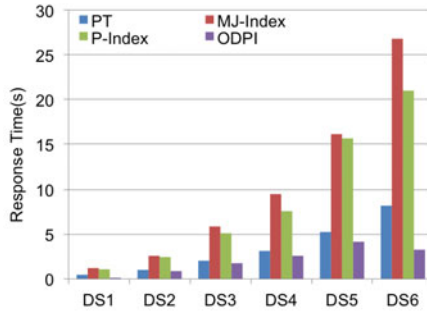


Fig. 7. Exp-2: Two Predication on Virtual Attribute

two predications, because indices on its head and tail classes can help to reduce objects access greatly.

6 Conclusions

This paper proposed a new index structure, ODPI, for path expression evaluation in Object Deputy Database. ODPI materializes path instances and uses traditional indices to assist computing predications. ODPI can effectively reduce objects access in object navigation stage of path expression evaluation. Cross-Class Query which is implemented by path expression evaluation can benefit from ODPI greatly. Results of experiments proved this. Maintenance method for ODPI is also given in this paper.

Although ODPI can increase performance of path expression evaluation, it supports only predications on head or tail class of the path. If there are some predications on middle classes, ODPI should be useless. This will be a further research point in future.

References

1. Peng, Z.Y., Kambayashi, Y.: Deputy mechanisms for object-oriented databases. In: 11th International Conference on Data Engineering (ICDE), pp. 333–340. IEEE Press, New York (1995)
2. Kambayashi, Y., Peng, Z.Y.: Object deputy model and its applications. In: 4th International Conference on Database Systems for Advanced Applications (DAS-FAA), pp. 1–15. World Scientific Press, New York (1995)
3. Peng, Z.Y., Li, Q., Feng, L., Li, X.H., Liu, J.Q.: Using object deputy model to prepare data for data warehousing. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 17(9), 1274–1288 (2005)
4. Peng, Z.Y., Luo, Y., Shan, Z., Li, Q.: Realization of workflow views based on object deputy model. *Chinese Journal of Computers* 28(4), 651–660 (2005)
5. Wang, L.W., Huang, Z.Q., Luo, M., Peng, Z.Y.: Data provenance in a scientific workflow service framework integrated with object deputy database. *Chinese Journal of Computers* 31(5), 721–732 (2008)

6. Peng, Z.Y., Shi, Y., Zhai, B.X.: Realization of biological data management by object deputy database system. In: Priami, C., Hu, X., Pan, Y., Lin, T.Y. (eds.) *Transactions on Computational Systems Biology V. LNCS (LNBI)*, vol. 4070, pp. 49–67. Springer, Heidelberg (2006)
7. Peng, Z.Y., Peng, Y.W., Zhai, B.X.: Using object deputy database to realize multi-representation geographic information system. In: *15th Annual ACM International Symposium on Advances in Geographic Information Systems (ACM GIS)*, pp. 43–46. ACM Press, New York (2007)
8. Zhai, B.X., Shi, Y., Peng, Z.Y.: Object deputy database language. In: *4th International Conference on Creating, Connecting and Collaborating through Computing (C5)*, pp. 88–95. IEEE Press, New York (2006)
9. Cattell, R.G.G.: *The object database standard: ODMG-93*. Morgan Kaufmann, San Francisco (1993)
10. Kifer, M., Kim, W., Sagiv, Y.: Querying object-oriented databases. In: *18th ACM SIGMOD International Conference on Management of Data*, pp. 393–402. ACM Press, New York (1992)
11. Gardarin, G., Gruser, J.R., Tang, Z.H.: Cost-based selection of path expression processing algorithms in object-oriented databases. In: *22th International Conference on Very Large Data Bases*, pp. 390–401. Morgan Kaufmann, San Francisco (1996)
12. Wang, G.R., Yu, G., Zhang, B., Zheng, H.Y.: THE parallel algorithms for path expressions. *Chinese Journal of Computers* 22(2), 126–133 (1992)
13. Maier, D., Stein, J.: Indexing in an object-oriented database. In: *Proceedings of IEEE Workshop on Object-Oriented Data Base Management Systems*, pp. 171–182. IEEE Press, New York (1986)
14. Xie, Z., Han, J.: Join index hierarchies for supporting efficient navigations in object oriented databases. In: *20th International Conference on Very Large Data Bases*, pp. 522–533. Morgan Kaufmann, San Francisco (1994)
15. Bertino, E., Foscoli, P.: Index organizations for object-oriented database systems. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 7(2), 193–209 (1995)
16. Bertino, E., Guglielmina, C.: Path-Index: An approach to the efficient execution of object-oriented queries. *Data and Knowledge Engineering* 10(1), 1–27 (1993)
17. Lee, W.C., Lee, D.L.: Path dictionary: A new access method for query processing in object-oriented databases. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 10(3), 371–388 (1998)
18. Kemper, A., Moerkotte, G.: Access support in object bases. In: *16th ACM SIGMOD International Conference on Management of Data*, pp. 364–374. ACM Press, New York (1990)

Ensemble Pruning via Base-Classifier Replacement*

Huaping Guo and Ming Fan

School of Information Engineering, Zhengzhou University, P.R. China
hpguo.gm@gmail.com, mfan@zzu.edu.cn

Abstract. Ensemble pruning is a technique to increase ensemble accuracy and reduce its size by choosing an optimal or suboptimal subset of ensemble members to form subensembles for prediction. A number of greedy ensemble pruning methods that are based on greedy search policy have recently been proposed. In this paper, we contribute a new greedy ensemble pruning method, called EPR, based on replacement policy. Unlike traditional pruning methods, EPR searches for the optimal or suboptimal subensemble with predefined size by iteratively replacing the least important classifier in it with current classifier. Especially, replacement would not occur if the current classifier was the least important one. Also, we adopt diversity measure [1] to theoretically analyze the properties of EPR, based on which a new metric is proposed to guide EPR's search process. We evaluate the performance of EPR by comparing it with other advanced greedy ensemble pruning methods and obtain very promising results.

Keywords: Ensemble Pruning, Greedy Search, Replacement Policy.

1 Introduction

Ensemble of multiple learning machines, i.e. a group of learners that work together as a committee, has been a very popular research topic during the last decade in machine learning and data mining. The main discovery is that an ensemble is potential to increase classification accuracy beyond the level reached by an individual classifier alone. The key to the success of ensembles is to create a collection of diverse and accurate base classifiers [2].

Many approaches can be used to create accurate and diverse ensemble members, for example, by manipulating the training set to create diverse training sets for base classifiers (e.g. bagging [3] and boosting [4]), by manipulating the input features to map the training set into different feature spaces so that diverse training sets are constructed for base classifiers (random forest [5], rotation forest [6] and COPEN [7]), and by manipulating the parameters or the structure of corresponding learning algorithm to create accurate and diverse ensembles.

* Supported by the National Science Foundation of China (No. 60901078).

One problem existing in these methods is that they tend to train a very large number of classifiers which decrease the response time for prediction. The minimization of run-time overhead is crucial in some application, such as stream mining. Equally important problem is that they tend to construct an ensemble with both high and low accurate classifiers. Intuitively, the latter may affect the expected performance of the ensemble. In terms of these problems, a technique called ensemble pruning [8,9], also named ensemble selection or ensemble thinning, is proposed to prune the low-performing members while maintaining the diversity in remaining members to reduce the response time and increase the expected performance.

In principle, there are exponentially many combination candidates of ensemble members. While some of the candidates performs better than others, finding the best candidate is computational infeasible because of the exponential size of the search space. Many ensemble pruning methods via greedy search [10,11,12,13,14,15,16,17,18,19] have been proposed: given a subensemble S which is initialized to be empty (or full), searching for the space of different combination candidates by iteratively adding into (or removing from) S the classifier $h \in H \setminus S$ (or $h \in S$) that optimizes a cost function. Empirical results show that these methods induce ensemble with small size and promising generalization performance.

In this paper, we propose a new greedy ensemble pruning algorithm called EPR (Ensemble Pruning via Base-Classifier Replacement) to reduce ensemble size and increase its accuracy. Unlike traditional greedy ensemble pruning algorithms, EPR adopts replacement policy to search for optimal or suboptimal subensembles with predefined number of members, which is much smaller than the original ensemble size. In addition, we theoretically analyze the properties of EPR using diversity measure. Based on these properties, a new metric is proposed to supervise the search of EPR. EPR is evaluated by comparing it with other advanced greedy ensemble pruning methods and very promising results are obtained.

The rest of this paper is organized as follows. After the background introduction in section 2, section 3 introduces the basic idea of EPR (the proposed ensemble pruning method), analyzes its properties and presents a new metric to guide its search process, followed by experimental results shown in section 4. Finally, section 5 concludes this paper and presents some future work.

2 Related Work

In 1997, Margineantu and Dietterich [10] introduce heuristics to calculate the benefit of adding a classifier to a subensemble, using forward selection in a number of them. They experiment with boosting algorithm and conclude that ensemble pruning can reduce ensemble size and improve its accuracy. Similar approach is adopted by Tamon and Xiang in 2000 to prune ensembles constructed by boosting algorithm [11].

Fan et al. [12] use forward selection, like in [10], to prune an ensemble of classifiers, where the search is guided by the benefit that is obtained by evaluating the

combination of the selected classifiers with majority voting. Experimental results show that ensemble pruning increase the generalization ability of an ensemble with C4.5 as base classifier.

Caruana et al. [13] construct an ensemble pool (2000 classifiers) using different algorithms and sets of parameters for these algorithms. Then they use forward selection, like in [10], to iteratively add into the subensemble the classifier in the pool that can optimize some cost function. In this way, they expect to construct an ensemble with better performance.

Martinez-Munoz et al. [14,15] reorder the ensemble members constructed by bagging into a list, through which a subensemble with arbitrary size can be formed by selecting the top p percent of ensemble members. A similar approach is adopted by Lu et al. [16] based on a measure called individual contribution (IC) of a classifier.

Banfield et al. [17] propose a method that selects subensembles in a backward manner. The authors reward each classifier according to its decision with regard to the ensemble decision. The method iteratively removes the classifier with the lowest accumulated reward. Similar approach is adopted by Partalas et al. [18,19]. In [18], Partalas et al. identify that the prediction of a classifier h and ensemble S on example \mathbf{x}_i can be categorized into four cases: (1) $e_{tf} : h(\mathbf{x}_i) = y_i \wedge S(\mathbf{x}_i) \neq y_i$, (2) $e_{tt} : h(\mathbf{x}_i) = y_i \wedge S(\mathbf{x}_i) = y_i$, (3) $e_{ft} : h(\mathbf{x}_i) \neq y_i \wedge S(\mathbf{x}_i) = y_i$, and (4) $e_{ff} : h(\mathbf{x}_i) \neq y_i \wedge S(\mathbf{x}_i) \neq y_i$, where y_i is a real label associated with \mathbf{x}_i . They conclude that considering all four cases is crucial to design ensemble diversity metrics and ensemble selection techniques. In 2010, they extend the work and propose a metric called Uncertainty Weighted Accuracy (UWA) [19]. Based on the four cases, Lu et al. [16] design a new diversity based measure called individual classifier contribution (IC) to supervise ensemble pruning.

In this paper, we propose a new greedy ensemble pruning method based on replacement policy: searching for the optimal/suboptimal subensemble whose size is predefined by iteratively replacing the least important classifier in it with current classifier. The details of this method are presented in next section.

3 Ensemble Pruning via Base-Classifier Replacement

In this section, we first introduces the basic idea of EPR in section 3.1, then discuss about EPR's properties in section 3.2, based on which a new metric is proposed in section 3.3 to guide the search of EPR, and finally, present the EPR's pseudo-code in section 3.4.

3.1 Problem Definition and the Idea of EPR

Let $H = \{h_1, h_2, \dots, h_M\}$ be an ensemble with M members, our problem is: how to select the subensemble S from H , such that $|S| = L \ll M$ and S 's performance is similar to or better than the original ensemble H .

Many methods have been proposed, as discussed in section 2. Here, we use replacement policy to prune ensemble H : initialize subensemble S using L members of H and iteratively replace the "least-important" classifier using current

classifier $h \in H \setminus S$ until the maximum iteration number is achieved. Especially, replacement would not occur if the current classifier was the least-importance one. We name this method as EPR (Ensemble Pruning via Base-Classifier Replacement).

The major issue in EPR is to design an effective metric to evaluate the importance of a classifier with respect to an ensemble S . Before the introduction of the details of the proposed metric, we first use diversity measure [1] to obtain the properties that supervise the design of the metric in next section.

3.2 Property Analysis

In this subsection, we first introduce the notations and the diversity measures used in this paper, and then derive properties that supervise the design of the metric used in EPR.

Let $D = \{(\mathbf{x}_i, y_i) | i = 1, 2, \dots, N\}$ be an example set where each example \mathbf{x}_i is associated with a label $y_i \in \{1, \dots, T\}$. Suppose $S = \{h_t | t = 1, 2, \dots, L\}$ is an ensemble with L classifiers and suppose each member $h_t \in S$ maps each example \mathbf{x}_i in D to a label y , $h_t(\mathbf{x}_i) = y \in \{1, \dots, T\}$. Assume that $V = \{v^{(\mathbf{x}_1)}, v^{(\mathbf{x}_2)}, \dots, v^{(\mathbf{x}_N)} | v^{(\mathbf{x}_i)} = [v_1^{(\mathbf{x}_i)}, \dots, v_T^{(\mathbf{x}_i)}], i = [1, N]\}$ is a set of vectors where $v_j^{(\mathbf{x}_i)}$ is the number of votes for label j of example \mathbf{x}_i of the ensemble S combined by majority voting.

Many measures [20] exist to evaluate the diversity (or distance) of classifiers or the diversity of an ensemble. In this paper, we employ a 0/1 loss based disagreement measure, which was proposed by Ho [1], to characterize the pair-wise diversity for ensemble members.

Definition 1. Given two classifiers h_i and h_j . Suppose $N_{ij}^{(01)}$ is the number of the examples in D incorrectly classified by h_i and correctly classified by h_j , and $N_{ij}^{(10)}$ is the opposite of $N_{ij}^{(01)}$, then the diversity of h_i and h_j with respect to D , denoted by $Div_D(h_i, h_j)$, is defined as

$$Div_D(h_i, h_j) = \frac{N_{ij}^{(01)} + N_{ij}^{(10)}}{N} \tag{1}$$

where N is the size of the example set D .

Definition 2. The diversity contribution of a classifier h with respect to ensemble S and example set D , denoted by $ConDiv_D(h, S)$, is defined as

$$ConDiv_D(h, S) = \sum_{h_j \in S} Div_D(h, h_j) = \sum_{h_j \in S \setminus \{h\}} Div_D(h, h_j) \tag{2}$$

Theorem 1. The diversity contribution of a classifier h , defined in Eq. 2, is equal to Eq. 3.

$$Con_D(h, S) = \frac{1}{N} \sum_{\mathbf{x}_i \in D} \left[I(h(\mathbf{x}_i) = y_j) \left(L - v_{y_i}^{(\mathbf{x}_i)} \right) + I(h(\mathbf{x}_i) \neq y_i) v_{y_i}^{(\mathbf{x}_i)} \right] \tag{3}$$

where $I(true) = 1$, $I(false) = 0$, L is the size of S , and $v_{y_i}^{(\mathbf{x}_i)}$ is the number of the classifiers in S that correctly classify the example \mathbf{x}_i .

Proof. Since $v_{y_i}^{(\mathbf{x}_i)}$ is the number of the classifiers in S that correctly classify the example \mathbf{x}_i , $L - v_{y_i}^{(\mathbf{x}_i)}$ is the number of classifiers that incorrectly classify \mathbf{x}_i . Therefore, the value in square bracket is exactly the number of the classifiers in S that disagree with h 's prediction on \mathbf{x}_i . Thus, the sum of the disagreement on each example of D is exactly the diversity contribution of h , defined in Eq. 2. \square

Theorem 1 is the extension of lemma 1 in [16] which only presents another form of Eq. 2 for two-class problem. Based on this theorem, we design a metric to evaluate the contribution of a classifier with respect to an ensemble and an example set in next section.

Definition 3. *The diversity of an ensemble S , denoted by $Div_D(S)$, is defined by the sum of the diversity contribution of each classifier in S , as shown in Eq. 4*

$$Div_D(S) = \sum_{h_i \in S} ConDiv_D(h_i, S) \quad (4)$$

Lemma 1. *Let $h' \notin S$ be a base-classifier and let $S' = S \setminus \{h\} \cup \{h'\}$ be an ensemble, i.e., S' is obtained from S by replacing h with h' . If $ConDiv_D(h', S') > ConDiv_D(h, S)$, then S' is better than S , i.e., $Div_D(S') > Div_D(S)$.*

Proof.

$$\begin{aligned} Div_D(S') &= \sum_{h_i \in S'} ConDiv_D(h_i, S') = ConDiv_D(h', S') + \sum_{h_i \in S' \setminus \{h'\}} ConDiv_D(h_i, S') \\ &= ConDiv_D(h', S') + \sum_{h_i \in S' \setminus \{h'\}} \sum_{h_j \in S' \setminus \{h_i\}} Div_D(h_i, h_j) \end{aligned} \quad ,$$

where

$$\sum_{h_j \in S' \setminus \{h_i\}} Div_D(h_i, h_j) = Div_D(h', h_i) + \sum_{h_j \in S' \setminus \{h_i, h'\}} Div_D(h_i, h_j).$$

Therefore

$$\begin{aligned} Div_D(S') &= ConDiv_D(h', S') + \sum_{h_i \in S' \setminus \{h'\}} Div_D(h', h_i) \\ &\quad + \sum_{h_i \in S' \setminus \{h'\}} \sum_{h_j \in S' \setminus \{h_i, h'\}} Div_D(h_i, h_j) \\ &= 2ConDiv_D(h', S') + \sum_{h_i \in S' \setminus \{h'\}} \sum_{h_j \in S' \setminus \{h_i, h'\}} Div_D(h_i, h_j). \end{aligned} \quad (5)$$

Similarly, we can prove

$$Div_D(S) = 2ConDiv_D(h, S) + \sum_{h_i \in S \setminus \{h\}} \sum_{h_j \in S \setminus \{h_i, h\}} Div_D(h_i, h_j). \quad (6)$$

Since $S' = S \setminus \{h\} \cup \{h'\}$,

$$S' \setminus \{h'\} = S \setminus \{h\}, S' \setminus \{h', h_i\} = S \setminus \{h, h_i\}. \quad (7)$$

Combining Eq. 5 Eq. 6 and Eq. 7, we have

$$Div_D(S') - Div_D(S) = 2 (ConDiv_D(h', S') - ConDiv_D(h, S)). \tag{8}$$

Therefore, $Div(S') > Div(S)$ if $ConDiv_D(h', S') > ConDiv_D(h, S)$. \square

Theorem 2. *Let $h' \notin H \setminus S$ be a base-classifier and let $S' = S \setminus \{h\} \cup \{h'\}$ be another ensemble, i.e., S' is obtained from S by replacing h with h' . Then the Eq. 9 holds*

$$Div_D(S') \propto ConDiv_D(h', S \setminus \{h\}) - ConDiv_D(h, S \setminus \{h\}) \tag{9}$$

Proof. Eq. 8 in the proof of lemma 1 results in

$$\begin{aligned} Div_D(S') &= Div_D(S) + 2 (ConDiv_D(h', S') - ConDiv_D(h, S)) \\ &= Div_D(S) + 2(ConDiv_D(h', S' \setminus \{h'\}) - ConDiv_D(h, S \setminus \{h\})). \end{aligned} \tag{10}$$

Then Eq. 9 results from Eq. 10 and $S' = S \setminus \{h\} \cup \{h'\}$ immediately. \square

Theorem 2 indicates that, searching for the best candidate S' with largest diversity in space $\{S' | S' = S \setminus \{h\} \cup \{h'\}, h \in S\}$ is equivalent to searching for the classifier $h \in S$ that maximizes the right item of Eq. 9. More generally, we can use Eq. 11 to select the candidate classifier h as the classifier to be replaced by h' so that $Div_D(S')$ is largest. If the classifier $h = h'$, then replacement will not occur. In addition, property 1 holds on the condition that EPR adopts Eq. 11 as the metric that guides its search.

$$h = \arg \max_{h \in S \cup \{h'\}} (ConDiv_D(h', S \setminus \{h\}) - ConDiv_D(h, S \setminus \{h\})) \tag{11}$$

Proposition 1. *After each iteration of EPR guided by Eq 11, the subensemble generated by EPR is better than before if the replacement happened, i.e., the diversity of the subensemble is larger than before.*

Proof. This proposition results from Eq. 9 and Eq. 11 immediately. \square

Based on theorem 1 and theorem 2, we define a new metric to guide the search of EPR. The specific details are present in next section.

3.3 The Proposed Metric

In this section, we design the metric according to theorem 1 and theorem 2 to supervise the search process of EPR .

The notations in this section follow the introduction in section 3.2. According to theorem 2, we define the *Contribution Gain* (CG) when h is replaced by h' as

$$CG_{D_{pr}}(h', h) = Con_{D_{pr}}(h', S \setminus \{h\}) - Con_{D_{pr}}(h, S \setminus \{h\}) \tag{12}$$

where $Con_{D_{pr}}(h, S)$ is h 's *contribution* with respect to ensemble S and pruning set D_{pr} , defined as

$$Con_{D_{pr}}(h, S) = \frac{1}{N} \sum_{\mathbf{x}_i \in D_{pr}} (I(h(\mathbf{x}_i) = y_i)(L - v_{y_i}^{(\mathbf{x}_i)}) + I(h(\mathbf{x}_i) \neq y_i)(v_{y_i}^{(\mathbf{x}_i)} - v_{max}^{(\mathbf{x}_i)})) \tag{13}$$

Algorithm 1. The proposed method in pseudocode

Input: L : subensemble size;
 H : the full ensemble;
 D_{pr} : the pruning set;
Output: the subensemble S with size L

- 1: Initialize S with L members of H ;
- 2: $H' = H \setminus S$;
- 3: **for** each $h' \in H'$ **do**
- 4: $h_r = \arg_h \max_{h \in S \cup \{h\}} CG_{D_{pr}}(h', h)$
- 5: **if** $h_r \neq h'$ **then**
- 6: $S = S \setminus \{h_r\} \cup \{h'\}$;
- 7: **end if**
- 8: **end for**
- 9: **return** S ;

where

- N is the size of pruning set D_{pr} ,
- $I(b)$ is equal to 1(or 0) if $b = \text{true}$ (or $b = \text{false}$),
- L is the size of ensemble S ,
- $h(\mathbf{x}_i)$ is the label of h 's prediction on example \mathbf{x}_i ,
- $v_{max}^{(\mathbf{x}_i)}$ is the number of majority votes of S on example \mathbf{x}_i ,
- $v_{y_i}^{(\mathbf{x}_i)}$ is the number of the votes on the label y_i .

The value of Eq. 13 regardless of the item $-v_{max}^{(\mathbf{x}_i)}$ is exactly h 's diversity contribution defined in Eq. 3 of theorem 1. The item, $-v_{max}^{(\mathbf{x}_i)}$, is to minus h 's contribution when h incorrectly classifies \mathbf{x}_i , that follows the conclusion in [16]: correct predictions make positive contributions and incorrect predictions make negative contribution.

EPR uses Eq. 14, which results from Eq. 12 and Eq. 11, to select the classifier h to be replaced by h' . If h is exactly h' , then replacement does not occur. The specific details of EPR are presented in next section.

$$h = \arg \max_{h \in S \cup \{h'\}} CG_{D_{pr}}(h', h) \tag{14}$$

3.4 Algorithm

The specific details of EPR (Ensemble Pruning via Base-Classifier Replacement) are shown in algorithm 1. EPR first initializes the subensemble S using L members of H (line 1), which can be selected randomly or obtained by some ensemble selection method, such as UWA [19]. We randomly initialize the subensemble S . Then EPR searches for the optimal or suboptimal subset by iteratively replacing the classifier that minimizes Eq. 14. If the classifier is exactly the current classifier, replacement does not occur. The complexity of EPR is $O(N|H|^2)$, where N is the pruning set size and $|H|$ is the original ensemble size.

4 Experiments

4.1 Data Sets and Experimental Setup

16 data sets, of which the details are shown in table 1, are randomly selected from UCI repository [21]. For each data set, a tenfold cross validation is performed: a data set is randomly split into ten disjunctive folds. For each fold, the other nine folds are treated as training set to train models and the corresponding fold as test set to evaluate the constructed models. For each trial, a bagging with 200 decision trees is trained. The base classifier is J4.8, which is a Java implementation of C4.5 [22] from Weka [23].

Two experiments are designed: the first evaluates the validation of CG (contribution gain, the proposed metric) and the second the performance of EPR. For the first experiment, CG, IC [16], UWA [19] and CON [17] are selected as the metrics that guide EPR's search process (denoted as EPR-CG, EPR-IC, EPR-UWA and EPR-CON respectively). With respect to the second experiment, EPR is compared with forward and backward greedy ensemble pruning methods, where CG and IC are selected as the representative metrics guiding the search process of these methods. We denote F- \star (B- \star) as forward (backward) ensemble pruning method which adopts the metric \star to guide its search process. For example, F-CG (B-CG) indicates that CG is used to guide the search of forward (backward) ensemble pruning method.

All pruning methods use training set as pruning set and adopt simple majority voting for predicting unseen example. To avoid overfitting, the size of subensemble selected by each pruning method is set to be 30% of the original ensemble size.

4.2 Experimental Results

The Proposed Metric Performance. This section reports the performance of the proposed metric CG. The corresponding results are shown in Table 2. For reference, we displace the accuracy of bagging (the whole ensemble) as well. The accuracy of the winning algorithm on each data set is highlighted with bold typeface. As shown in Table 2, EPR-CG achieves the best performance in most of the data sets (8), followed by followed by EPR-IC (5), EPR-UWA(3), EPR-CON(2), and finally bagging(0).

Table 1. Characteristics of the 16 Data Sets Used in experiments

Dataset	Size	Class	Attributes	Dataset	Size	Class	Attributes
australian	609	2	14	machine	209	8	7
autos	205	7	25	mofn-3-7-10	300	2	11
breast-wisconsin	699	2	9	page-blocks	5473	5	10
car	1728	6	4	promoters	106	2	57
cars	406	3	8	segment	2310	7	19
german-credit	1000	2	20	vehicle	846	4	18
heart-statlog	270	2	13	vowel	909	11	12
lymphography	148	4	18	zoo	101	7	17

Table 2. The mean accuracy of EPR-CG, EPR-IC, EPR-UWA, EPR-CON and bagging. The accuracy of the winning algorithm on each data set is highlighted with bold typeface.

Dataset	EPR-CG	EPR-IC	EPR-UWA	EPR-CON	bagging
australian	85.48	85.68	85.30	85.51	85.59
autos	75.61	76.30	75.52	75.23	75.13
breast-wisconsin	96.37	96.34	96.22	96.34	96.25
car	90.67	90.73	90.76	90.56	89.92
cars	82.76	82.32	82.66	82.02	82.02
german-credit	73.90	73.86	73.54	74.40	74.34
heart-statlog	81.78	81.78	81.93	81.56	81.33
lymphography	78.92	80.00	78.65	78.92	78.92
cpu	87.56	86.99	87.37	86.12	85.65
mofn-3-7-10	86.60	86.60	86.33	85.93	85.80
page-blocks	97.20	97.20	97.20	97.22	97.01
promoters	85.47	85.28	85.09	81.51	83.40
segment	96.60	96.58	96.64	96.50	96.42
vehicle	73.90	74.07	73.85	73.40	73.78
vowel	85.29	85.20	85.29	84.55	84.57
zoo	92.88	92.49	92.85	91.69	91.30

Table 3. The rank of EPR-CG, EPR-IC, EPR-UWA, EPR-CON and bagging on each data set.

Dataset	EPR-CG	EPR-IC	EPR-UWA	EPR-CON	bagging
australian	4	1	5	3	2
autos	2	1	3	4	5
breast-wisconsin	1	2.5	5	2.5	4
car	3	2	1	4	5
cars	1	3	2	4.5	4.5
german-credit	3	4	5	1	2
heart-statlog	2.5	2.5	1	4	5
lymphography	3	1	5	3	3
cpu	1	3	2	4	5
mofn-3-7-10	1.5	1.5	3	4	5
page-blocks	3	3	3	1	5
promoters	1	2	3	5	4
segment	1	2	3	4	5
vehicle	2	1	3	4	5
vowel	1.5	3	1.5	5	4
zoo	1	3	2	4	5
Average Rank	1.97	2.22	2.97	3.69	4.16

According to [24], the appropriate way to compare two or more algorithms on multiple data sets is based on their average rank across all datasets. On each dataset, the algorithm with the highest accuracy gets rank 1.0, the one with the second highest accuracy gets rank 2.0 and so on. In case two or more algorithms tie, they all receive the average of the ranks that correspond to them.

Table 3 presents the rank of each algorithm on each data set, along with the average ranks. We notice that the best performing algorithm is EPR-CG with average rank 1.97, while EPR-IC, EPR-UWA, EPR-CON and bagging follow up with average ranks 2.22, 2.97, 3.69 and 4.16, respectively.

The results above indicate that: (1) EPR outperforms bagging, no matter which metric is adopted to guide the search of EPR, and (2) compared with the other metrics, CG is a better metric for EPR.

Table 4. The mean accuracy and rank of EPR-CG, F-CG and B-CG on each data set

Data Set	Accuracy			Rank		
	EPR-CG	F-CG	B-CG	EPR-CG	F-CG	B-CG
australian	85.48	85.42	85.42	1	2.5	2.5
autos	75.61	75.42	75.42	1	2.5	2.5
breast-wisconsin	96.37	96.37	96.37	2	2	2
car	90.67	90.59	90.66	1	3	2
cars	82.76	82.56	82.66	1	3	2
german-credit	73.90	73.80	73.88	1	3	2
heart-statlog	81.78	81.33	81.85	2	3	1
lymphography	78.92	79.19	78.92	1.5	3	1.5
cpu	87.56	87.47	87.47	1	2.5	2.5
mofn-3-7-10	86.60	86.53	86.60	1.5	3	1.5
page-blocks	97.20	97.19	97.19	1	2.5	2.5
promoters	85.47	85.09	85.28	1	3	2
segment	96.60	96.60	96.61	2.5	2.5	1
vehicle	73.90	73.95	73.71	2	1	3
vowel	85.29	85.32	85.01	2	1	3
zoo	92.88	92.88	92.68	1.5	1.5	3

Table 5. The mean accuracy and rank of EPR-IC, F-IC and B-IC on each data set

Data Set	Accuracy			Rank		
	EPR-IC	F-IC	B-IC	EPR-IC	F-IC	B-IC
australian	85.68	85.57	85.42	1	2	3
autos	76.30	76.59	76.10	2	1	3
breast-wisconsin	96.34	96.31	96.31	1	2.5	2.5
car	90.73	90.66	90.69	1	3	2
cars	82.76	82.56	82.66	1	3	2
german-credit	73.86	73.86	73.82	1.5	1.5	3
heart-statlog	81.78	81.48	81.70	2	3	1
lymphography	80.00	80.00	79.86	1.5	3	1.5
cpu	86.99	86.89	86.89	1	2.5	2.5
mofn-3-7-10	86.60	86.53	86.60	1.5	3	1.5
page-blocks	97.20	97.19	97.20	1.5	3	1.5
promoters	85.28	85.09	85.47	2	3	1
segment	96.58	96.56	96.56	1	2.5	2.5
vehicle	74.07	73.90	74.14	2	3	1
vowel	85.20	85.34	85.09	2	1	3
zoo	92.49	92.49	92.49	2	2	2

The Performance of EPR. This section reports the performance of EPR compared with forward and backward ensemble pruning methods, where CG and IC are adopted as the metrics guiding the search of these pruning methods. The corresponding results are shown in Table 4 and Table 5, where Table 4 is the mean classification accuracy and ranks of EPR-CG, F-CG and B-CG, and table 5 is the results of EPR-IC, F-IC and B-IC.

Intuitively, EPR, F- \star and B- \star performs comparable to each other, since the idea of these pruning methods is similar. The results in Table 4 and Table 5 validate this intuition: the difference between classification accuracy of the three pruning methods on each data set is small, no matter whether the methods supervised by CG or by IC.

Table 4 and Table 5 also show that, the rank of EPR is smallest on most of the 16 data sets, although EPR performs comparable to the other two pruning methods. This results indicates that, compared with other state-of-the-art ensemble pruning methods, EPR can induce small subensemble with better performance.

5 Conclusion

This paper contributes a new greedy ensemble pruning method called EPR based on replacement policy. EPR searches for the optimal or suboptimal subensemble whose size is predefined by iteratively replacing the least important classifier in it with current classifier. In addition, this paper theoretically analyzes the properties of EPR using diversity measure. Based on these properties, a new metric is proposed to guide the search of EPR. EPR is evaluated by comparing it with other advanced ensemble pruning methods and very promising results is obtained.

The key to the success of EPR is to design an effective metric to evaluate the contribution gain when a classifier is replaced with current classifier. Although experiments show that the measure proposed in this paper can reasonably evaluate the contribution gain, the study in this field is just beginning and better metrics are expected to be proposed.

References

1. Ho, T.K.: The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20(8), 832–844 (1998)
2. Kuncheva, L.I.: *Combining Pattern Classifiers: Methods and Algorithms*. John Wiley and Sons, Chichester (2004)
3. Breiman, L.: Bagging predictors. *Machine Learning* 24, 123–140 (1996)
4. Freund, Y., Schapire, R.F.: A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* 55(1), 119–139 (1997)
5. Breiman, L.: Random forests. *Machine Learning* 45(1), 5–32 (2001)
6. Rodríguez, J.J., Kuncheva, L.I., Alonso, C.J.: Rotation forest: A new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28(10), 1619–1630 (2006)
7. Zhang, D., Chen, S., Zhou, Z., Yang, Q.: Constraint projections for ensemble learning. In: *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI 2008)*, pp. 758–763 (2008)
8. Zhou, Z.H., Wu, J., Tang, W.: Ensembling neural networks: Many could be better than all. *Artificial Intelligence* 137(1-2), 239–263 (2002)
9. Zhang, Y., Burer, S., Street, W.N.: Ensemble pruning via semi-definite programming. *Journal of Machine Learning Research* 7, 1315–1338 (2006)
10. Margineantu, D.D., Dietterich, T.G.: Pruning adaptive boosting. In: *Proceedings of the 14th International Conference on Machine Learning*, pp. 211–218 (1997)
11. Tamon, C., Xiang, J.: On the Boosting Pruning Problem. In: Lopez de Mantaras, R., Plaza, E. (eds.) *ECML 2000. LNCS (LNAI)*, vol. 1810, pp. 404–412. Springer, Heidelberg (2000)
12. Fan, W., Chun, F., Wang, H.X., Yu, P.S.: Pruning and dynamic scheduling of cost-sensitive ensembles. In: *Proceeding of Eighteenth National Conference on Artificial intelligence, AAAI*, pp. 145–151 (2002)
13. Caruana, R., Niculescu-Mizil, A., Crew, G., Ksikes, A.: Ensemble Selection from Libraries of Models. In: *Proceedings of the Twenty-First International Conference (2004)*

14. Martinez-Muverbnoz, G., Suarez, A.: Aggregation ordering in bagging. In: Proceeding of The International Conference on Artificial Intelligence and Applications (IASTED), pp. 258–263. Acta press, Calgary (2004)
15. Martinez-Muverbnoz, G., Suarez, A.: Pruning in ordered bagging ensembles. In: Proceeding of the 23rd International Conference on Machine Learning, pp. 609–616 (2006)
16. Lu, Z.Y., Wu, X.D., Zhu, X.Q., Bongard, J.: Ensemble Pruning via Individual Contribution Ordering. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 871–880 (2010)
17. Banfield, R.E., Hall, L.O., Bowyer, K.W., Kegelmeyer, W.P.: Ensemble diversity measures and their application to thinning. *Information Fusion* 6(1), 49–62 (2005)
18. Partalas, I., Tsoumakas, G., Vlahavas, I.P.: Focused Ensemble Selection: A Diversity-Based Method for Greedy Ensemble Selection. In: 18th European Conference on Artificial Intelligence, pp. 117–121 (2008)
19. Partalas, I., Tsoumakas, G., Vlahavas, I.P.: An ensemble uncertainty aware measure for directed hill climbing ensemble pruning. *Machine Learning*, 257–282 (2010)
20. Kuncheva, L.I., Whitaker, C.J.: Measures of diversity in classifier ensemble and their relationship with the ensemble accuracy. *Machine Learning* 15(2), 181–207 (2003)
21. Asuncion, D.N.A.: UCI machine learning repository (2007)
22. Quinlan, J.R.: C4.5: programs for machine learning. Morgan Kaufmann, San Francisco (1993)
23. Witten, I.H., Frank, E.: *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd edn. Morgan Kaufmann, San Francisco (2005)
24. Demsar, J.: Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research* 7, 1–30 (2006)

Efficient Approximate Similarity Search Using Random Projection Learning

Peisen Yuan¹, Chaofeng Sha¹, Xiaoling Wang², Bin Yang¹, and Aoying Zhou²

¹ School of Computer Science, Shanghai Key Laboratory of Intelligent Information Processing, Fudan University, Shanghai 200433, P.R. China

² Shanghai Key Laboratory of Trustworthy Computing, Software Engineering Institute, East China Normal University, Shanghai 200062, P.R. China

{peiseny, cfsha, byang}@fudan.edu.cn

{xlwang, ayzhou}@sei.ecnu.edu.cn

Abstract. Efficient similarity search on high dimensional data is an important research topic in database and information retrieval fields. In this paper, we propose a random projection learning approach for solving the approximate similarity search problem. First, the random projection technique of the locality sensitive hashing is applied for generating the high quality binary codes. Then the binary code is treated as the *labels* and a group of SVM classifiers are trained with the *labeled* data for predicting the binary code for the similarity queries. The experiments on real datasets demonstrate that our method substantially outperforms the existing work in terms of preprocessing time and query processing.

1 Introduction

The similarity search, also known as the k -nearest neighbor query, is a classical problem and core operation in database and information retrieval fields. The problem has been extensively studied and applied in many fields, such as content-based multimedia retrieval, time series and scientific database, text documents etc. One of the common characteristics of these kinds of data is high-dimensional.

Similarity search on the high-dimensional data is a big challenge due to the time and space demands. However, in many real applications, the approximate results obtained in a more constrained time and space setting can also satisfy the users' requirements. For example, in the content-based image retrieval problem, a similar image can be returned as the result. Recently, researchers propose the approximate way for the similarity query, which can provide the satisfying results with much efficiency improvement [1–5].

The locality sensitive hashing (LSH for short) [1] is an efficient way for processing similarity search approximately. The principle of LSH is that the more similar the objects, the higher probability they fall into the same hashing bucket. The random projection technique of LSH is designed to approximately evaluate the cosine similarity between vectors, which transforms the high-dimensional data into much lower dimensions with the compact bit vectors.

However, the LSH is a data-independent approach. Recently, the learning-based data-aware methods, such as semantic hashing [6], are proposed and improve the search

efficiency with much shorter binary code. The key of the learning-based approaches is designing a way to obtain the binary codes for the data and the query. For measuring the quality of the binary code, the *entropy maximizing* criterion is proposed [6].

The *state-of-the-art* of learning-based technique is self-taught hashing (STH for short) [7], which converts the similarity search into a two stage learning problem. The first stage is the unsupervised learning of the binary code and the second one is supervised learning with the two-classes classifiers. In order to obtain the binary code satisfying the *entropy maximizing* criterion, the adjacency matrix of the k -NN graph of the dataset is constructed firstly. After solving the matrix by the binarised Laplacian Eigenmap, the median of the eigenvalues is set as the threshold for getting the bit labels: if the eigenvalue is larger than the threshold, the corresponding label is 1, otherwise is 0. In the second stage, binary code of the objects is taken as the class label, then classifiers are trained for *predicting* the binary code for the query. However, the time and space complexity of the preprocessing stage is considerably high.

In this paper, based on the *filter-and-refine* framework, we propose a random projection learning (RPL for short) approach for the approximate similarity search problem, which requires much less time and space cost for acquiring the binary code in the preprocessing step. Firstly, the random projection technique is used for obtaining the binary code of the data objects. The binary codes are used as the labels of the data objects and then the l SVM classifiers are trained subsequently, which are used for predicting the binary *labels* for the queries. We prove that the binary code after random projection satisfies the *entropy maximizing* criterion which is required by semantic hashing. Theoretical analysis and empirical experiment study on the real datasets are conducted, which demonstrate that our method gains comparable effectiveness with much less time and space cost.

To summarize, the main contributions of this paper are briefly outlined as follows:

- Random projection learning method for similarity search is proposed, and the approximate k -nearest neighbor query is studied.
- Properties of random projection of LSH that satisfying the *entropy maximizing* criterion needed by the semantic hashing is proved.
- Extensive experiments are conducted to demonstrate the effectiveness and efficiency of our methods.

The rest of paper is organized as follows. The random projection learning method for approximate similarity search is presented in Section 2. An extensive experimental evaluation is introduced in Section 3. In Section 4, the related work is briefly reviewed. In Section 5, the conclusion is drawn and future work is summarized.

2 Random Projection Learning

2.1 The Framework

The processing framework of RPL is described in Figure 1(a). Given data set S , firstly, the random projection technique of LSH is used for obtaining the binary code. After that, the binary code is treated as the labels of the data objects. Then l SVM classifiers

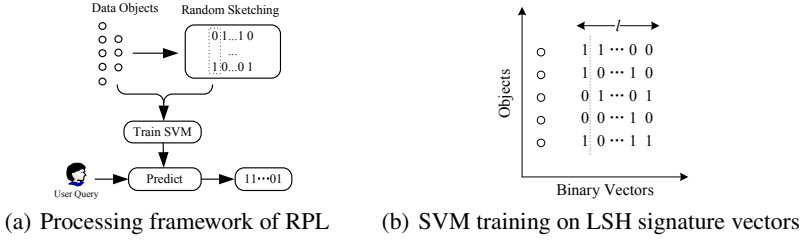


Fig. 1. The processing framework and the training with signature

are trained subsequently, which are used for predicting the binary *labels* for the queries. The binary code after random projection satisfying the *entropy maximizing* criterion in section 5

To answer a query, the binary labels of the query are *learned* with these l classifiers firstly. Then the similarities are evaluated in the hamming space, and hamming distance between the query less than a threshold is treated as the candidates. The distances or similarities are evaluated on the candidate set. Results are re-ranked and returned finally. In this paper, we take the approximate k -NN query into account.

Since there is no need of computing the k -NN graph, LapEng and median, our framework can efficiently answer the query with much less preprocessing time and space consumption comparing with STH [7].

In this paper, the cosine similarity and Euclidean distance are used as the similarity metric and the distance metric without pointing out specifically.

2.2 Random Projection

M. Charikar [2] proposes the random projection technique using random hyperplanes, which preserves the cosine similarity between vectors in lower space. The random projection is a method of locality sensitive hashing for dimensionality reduction, which is a powerful tool designed for approximating the cosine similarity.

Let \mathbf{u} and \mathbf{v} be two vectors in \mathbb{R}^d and $\theta(\mathbf{u}, \mathbf{v})$ be the angle between them. The cosine similarity between \mathbf{u} and \mathbf{v} is defined as Eq. 1

$$\text{cosine}(\theta(\mathbf{u}, \mathbf{v})) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \|\mathbf{v}\|} \quad (1)$$

Given a vector $\mathbf{u} \in \mathbb{R}^d$, a random vector \mathbf{r} is randomly generated with each component randomly choosing from the standard normal distribution $N(0, 1)$. Each hash function h_r of the random projection LSH family \mathcal{H} is defined as Eq. 2

$$h_r(\mathbf{u}) = \begin{cases} 1 & : \text{if } \mathbf{r} \cdot \mathbf{u} \geq 0; \\ 0 & : \text{otherwise.} \end{cases} \quad (2)$$

Given the hash family \mathcal{H} and vectors \mathbf{u} and \mathbf{v} , Eq. 3 can be obtained [2].

$$\text{Pr}[h_r(\mathbf{u}) = h_r(\mathbf{v})] = 1 - \frac{\theta(\mathbf{u}, \mathbf{v})}{\pi} \quad (3)$$

Form Eq. 3, the cosine similarity between \mathbf{u} and \mathbf{v} can be approximately evaluated with Eq. 4¹

$$\text{cosine}(\theta(\mathbf{u}, \mathbf{v})) = \text{cosine}((1 - Pr[h_r(\mathbf{u}) = h_r(\mathbf{v})])\pi) \quad (4)$$

For the random projection of LSH, l hash functions h_{r_1}, \dots, h_{r_l} are chosen from \mathcal{H} . After hashing, the vector \mathbf{u} can be represented by the signature s as Eq. 5

$$s = \{h_{r_1}(\mathbf{u}), \dots, h_{r_l}(\mathbf{u})\}. \quad (5)$$

The more similar of the data vectors, the higher probability they are projected with the same labels.

2.3 The Algorithm

The primary idea of RPL is that: (1) similar data vectors have similar binary code after random projection, and the disparity of binary code predicted for the query with the data vector set should be small; (2) The smaller distance between the vectors, the high chance they belong to the same class.

Before introducing the query processing, the definitions used in the following sections are introduced firstly.

Definition 1. Hamming Distance. Given two binary vectors \mathbf{v}_1 and \mathbf{v}_2 with equal length L , their Hamming distance is defined as $H_{dist}(\mathbf{v}_1, \mathbf{v}_2) = \sum_{i=1}^L \mathbf{v}_1[i] \neq \mathbf{v}_2[i]$.

Definition 2. Hamming Ball Coverset. Given an integer R , a binary vector \mathbf{v} and a vector set \mathbf{V}_b , the Hamming Ball Coverset relative to R of \mathbf{v} is denoted as $BC_R(\mathbf{v}) = \{\mathbf{v}_i | \mathbf{v}_i \in \mathbf{V}_b, \text{ and } H_{dist}(\mathbf{v}, \mathbf{v}_i) \leq R\}$, where R is Radius.

The intuitive meaning of Hamming Ball Coverset is the set of all the binary vectors in the set \mathbf{V}_b whose Hamming distance to \mathbf{v} is less than or equal to R .

Obtaining the Binary Vector. The algorithm of random projection used for generating the binary code is illustrated in Algorithm 1.

First a random matrix $\mathbf{R}^{d \times l}$ is generated (line 1). The entry of the vector of the matrix is chosen from $N(0, 1)$ and normalized, i.e., $\sum_{i=1}^d r_{ij}^2 = 1, j = 1, \dots, l$. For each vector \mathbf{v} of the set V , the inner products with each vector of the matrix $\mathbf{R}^{d \times l}$ are evaluated (lines 3 - 4). Therefore, the signatures of $\mathbf{v} \in V$ can be obtained as Eq. 5.

After projecting all the objects, a signature matrix $\mathcal{S} \in \mathbf{R}^{n \times l}$ can be obtained. Row of the signature matrix represents the object vector and the column represents the LSH signatures. In order to access the signature easily, an inverted list of the binary vector is built based the LSH signatures (line 5). Each signature vector \mathbf{v}_b of the corresponding vector is a binary vector $\in \{0, 1\}^l$. Finally, the inverted list of binary vectors returned (line 6).

¹ The similarity value of Eq. 4 can be normalized between 0 and 1.

Algorithm 1. Random Projection of LSH

Input: Object vector set $\mathcal{V}, \mathcal{V} \subseteq \mathbb{R}^d$.
Output: Inverted List of Binary Vector (*ILBV*).

```

1  $ILBV = \emptyset$ ;
2 generate a normalized random matrix  $\mathbf{R}^{d \times l}$ ;
3 foreach  $v \in \mathcal{V}$  do
4    $v_b = \emptyset$ ;
5   foreach vector  $r$  of  $\mathbf{R}^{d \times l}$  do
6      $h = r \cdot v$ ;
7      $val = \text{sgn}(h)$ ;
8      $v_b.append(val)$ ;
9    $ILBV.add(v_b)$ ;
10 return  $ILBV$ ;
```

Training Classifiers. The support vector machine (SVM) classifying technique is a classic classifying technique in data mining field. In this paper, the SVM classifiers are used and trained with the labeled data vectors.

Given labeled training data set $(\mathbf{x}_i, y_i), i = 1, \dots, n$, where $\mathbf{x}_i \in \mathbb{R}^d$ and $y \in \{-1, 1\}^l$. The solution of SVM is formalized as the following optimization problem.

$$\min_{\mathbf{w}, b, \xi_i \geq 0} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^n \xi_i \quad (6)$$

subject to $y_i \mathbf{w}^T \mathbf{x}_i \geq 1 - \xi_i$

The learning procedure after obtaining the binary labels is presented in Figure 1(b). The x -axis represents the binary vectors of the LSH signatures, and the y -axis represents the data object set². As illustrated in Figure 1(b), the column of the binary vectors is used as the class labels of the objects, then the first SVM classifier is trained. Since the length of signature vector is l , l SVM classifiers can be trained in the same manner.

The classifier training algorithm used in RPL is described in Algorithm 2. For each column of the signature matrix, an SVM classifier is trained at first (lines 1-4). In the same manner, l classifiers are trained. Finally, these l classifiers are returned, which are denoted as (\mathbf{w}_j, b_j) , for $j = 1, \dots, l$ (line 5).

Processing Query. The procedure for processing the approximate k -NN query is summarized in Algorithm 3. The algorithm consists of two stages: filter and refine. In the filter stage, the binary vector for the query is obtained with the l SVM classifiers firstly (lines 4-6). After that, the Hamming distances of the query with each binary vector in *ILBV* are evaluated (lines 7-9) and the distances are sorted (line 10). The objects whose Hamming distance larger than R are filtered in this stage. In the refine stage, the Euclidean distances are evaluated on the candidate set which fall in the Hamming Ball Coverset with radius R (lines 11-14). The top- k results are returned after being sorted finally (lines 15-16).

² label $\{0,1\}$ can be transformed to $\{-1,1\}$ plainly.

Algorithm 2. SVM Training of RPL

Input: Object vector $\mathbf{v}_i \in \mathcal{V}, i = 1, \dots, n, \mathcal{V} \subseteq \mathbb{R}^d; ILBV$.
Output: l SVM Classifiers.

- 1 **for** ($j = 1; j \leq l; j++$) **do**
- 2 **foreach** $\mathbf{v}_i \in \mathcal{V}$ **do**
- 3 $\mathbf{v}_b[i] = \text{get}(ILBV, \mathbf{v}_i);$
- 4 $SVMTrain(\mathbf{v}_i, \mathbf{v}_b[i]);$
- 5 **return** $(\mathbf{w}_j, b_j), j = 1, \dots, l;$

Algorithm 3. Approximate k -NN Query Algorithm

Input: Query $q; l$ SVM Classifiers; $ILBV$, Hamming Ball Radius R , Integer k .
Output: Top- k Result List.

- 1 $HammingResult = \emptyset;$
- 2 $Result = \emptyset;$
- 3 Vector $\mathbf{q}_v = \text{new Vector}();$ //The query binary vector
- 4 **for** ($i = 1; i \leq l; i++$) **do**
- 5 $b = SVMPredict(q, svm_i);$
- 6 $\mathbf{q}_v = \mathbf{q}_v.append(b);$
- 7 **foreach** $\mathbf{v}_b \in ILBV$ **do**
- 8 $H_{dist} = HammingBallDist(\mathbf{v}_b, \mathbf{q}_v);$
- 9 $HammingResult = HammingResult \cup H_{dist};$
- 10 $\text{sort}(HammingResult);$
- 11 Select the vectors $\tilde{\mathbf{V}} = BC_R(\mathbf{q}_v);$
- 12 **foreach** $\mathbf{v} \in \tilde{\mathbf{V}}$ **do**
- 13 $distance = \text{dist}(\mathbf{v}, q);$
- 14 $Result = Result \cup distance;$
- 15 $\text{sort}(Result);$
- 16 **return** Top- k result list;

2.4 Complexity Analysis

Suppose the object vector $\mathbf{v} \in \mathbb{R}^d$, and the length of the binary vector after random projection is l . In the algorithm 1 for generating the matrix $\mathbf{R}^{d \times l}$, its time and space complexity is $O(dl)$ and $O(dl)$ respectively. Let z be the number of non-zero values per data object. Training l SVM classifiers takes $O(lzn)$ or even less [8].

For the query processing, algorithm 3 predicts l bits with the l SVM classifiers takes $O(lzn \log n)$ time complexity. Let the size of the Hamming Ball Coverset $|BC_R(q)| = C$, the evaluation of the candidate with the query takes $O(Cl)$. The sort step takes $O(n \log n)$. Therefore, the query complexity is $O(lzn \log n + Cl + n \log n)$.

The time and space complexity of the preprocessing step of STH is $O(n^2 + n^2k + lnkt + ln)$ and $O(n^2)$ respectively [7].

Thus the time complexity of the preprocessing of our method $O(dl) \ll O(n^2 + n^2k + lnkt + ln)$, and space complexity $O(dl) \ll O(n^2)$ also, where l is much smaller than n .

3 Experimental Study

3.1 Experimental Setup

All the algorithms are implemented in Java SDK1.6 and run on an Ubuntu 10.04 enabled PC with Intel duo core E6550 2.33GHz CPU and 4G main memory. The SVM Java Package [9] is used and the kernel function is configured to linear kernel function with other default configurations. In term of the dataset, the following two datasets are used.

WebKB [10] contains web pages collected from the computer science departments of various universities by the World Wide Knowledge Base project, which is widely used for text mining. It includes 2803 pages for training, which are mainly classified into *faculty*, *staff*, *department*, *course* and *project* 5 classes.

Reuters-21578 [11] is a collection of news documents used for text mining, which includes 21578 documents and 17 topics. Due to space limited when solving the k -NN matrix problem, we randomly select 2083 documents which include *acq*, *crude*, *earn*, *grain*, *interest*, *money-fx*, *ship*, *trade* 8 classes.

The vectors are generated with TF-IDF vector model [12] after removing the stop-words and stemming. In total, 50 queries are issued on each dataset and the average results are reported. The statistics of the datasets are summarized in the table 1.

Table 1. Statistics of the datasets

Dataset	Document No.	Vector Length	Class No.
WebKB	2803	7287	5
Reuters	2083	14575	8

For evaluating the effectiveness, the k -precision metric is used and defined as k -precision = $\frac{|aNN_k \cap eNN_k|}{k}$, where aNN_k is the result list of the approximate k -NN query, and eNN_k is the k -NN result list by the linear scanning.

3.2 Effectiveness

To evaluate the query effectiveness of the approximate k -NN query algorithm, we test the k -precision varying the bit length L and the radius R . In this experiment, the parameter k of the k -precision is set to 40. The binary length L varies from 4 to 28, the radius R is selected between 1 to 12. The experiments results of the effectiveness of RPL on the two datasets are presented in Figure 2.

From the Figure 2, we can observe that (1) for a certain radius R , with the increasing of the parameter L , the precision drops quickly. For example, when $R = 8$, the k -precision drops from 1.0 to 0.2 when the length L varies from 10 to 28 on the two datasets. (2) given the parameter L , increasing the radius can improve the precision. Considering when the length L is 15, the precision increases from 0.1 to 1.0 with the radius R varies from 1 to 12. The reason that the bigger the radius R , more candidates are survived after filtering, thus the higher precision.

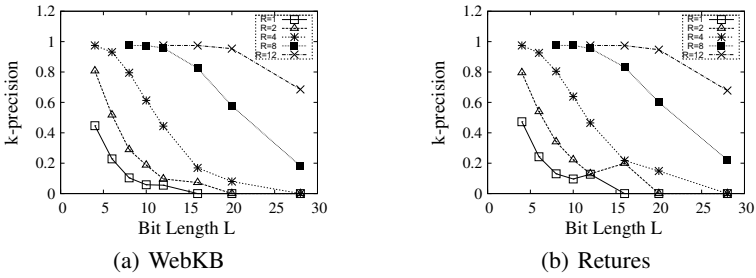


Fig. 2. *k*-Precision on two datasets

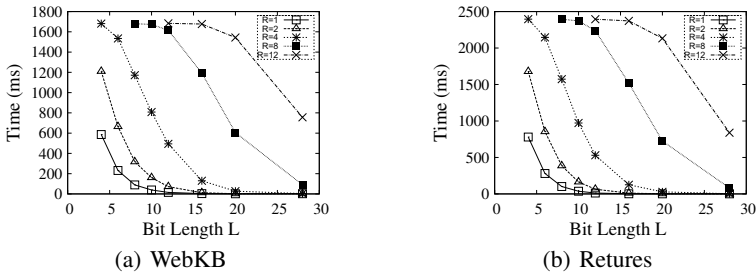


Fig. 3. Query efficiency on the two Datasets

3.3 Efficiency

To evaluate the efficiency of RPL, the approximate *k*-NN query processing time is tested when varying the radius *R* and the bit length *L*. The test result is demonstrated in Figure 3. In this experiment, the radius *R* varies from 1 to 12, and the bit length is selected between 4 and 30.

From the performance evaluation, conclusions can be drawn that: (1) with the increasing of the bit length *L*, the query processing time drop sharply, because there are fewer candidates; (2) the bigger the radius are, the higher query processing time for there are more candidates. But it is much smaller than the linear scan, which takes about 1737 and 2641 ms on the WebKB and Reuters datasets respectively.

3.4 Comparison

Comparisons with STH [7] are made on both of effectiveness and efficiency. STH takes about 9645 and 7106 seconds on WebKB and Retures datasets respectively for preprocessing, i.e., evaluating *k*-NN graph and the the Eigenvectors of the matrix. However, the preprocessing time of our method is much less than STH, only takes only several milliseconds, i.e., only a random matrix is needed to generate. In this comparison, the parameter *k* of the *k*-NN graph used in STH is set to 25.

The effectiveness comparison with STH is shown in Figure 4. The radius *R* is set to 3, and the parameter *k* to 3 and 10. The *k*-precision is reported varying bit length *L* on the two datasets. The experiment results show that the precision of RPL is a bit lower

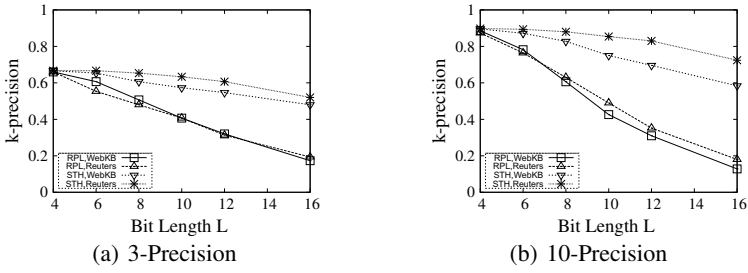


Fig. 4. Precision comparison

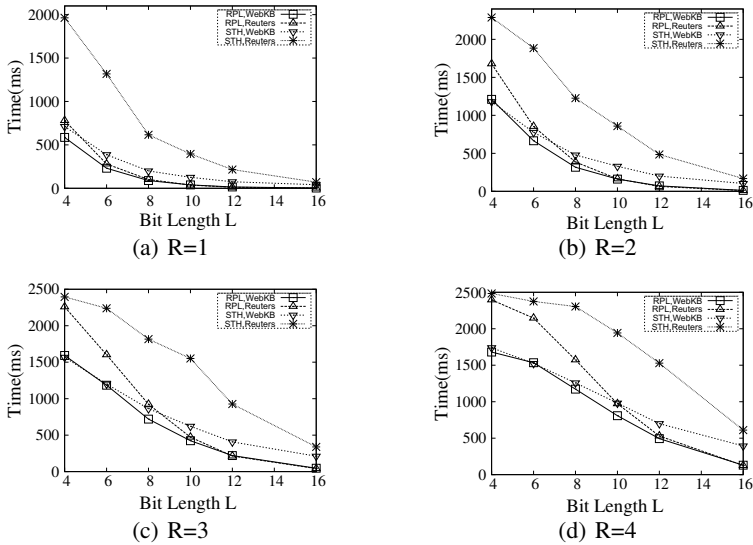


Fig. 5. Performance comparison with different R

than the STH, because STH use the k -NN graph of the dataset, and this take the data relationship into the graph.

The efficiency comparison with STH is conducted and the results are shown in Figure 5. In this experiment, the bit length varies from 4 to 16, results show the query processing time when varying the the radius R and different bit length L . Results demonstrate that the performance of our method is better than STH, which indicates better filtering skill.

4 Related Work

The similarity search problem is well studied in low dimensional space with the space partitioning and data partitioning index techniques [13–15]. However, the efficiency

degrades after the dimension is large than 10 [16]. Therefore, the researches propose the approximate techniques to process the problem approximately.

Approximate search method improves the efficiency by relaxing the precision quality. MEDRANK [17] solves the nearest neighbor search by sorted list and TA algorithm [18] in an approximate way. Yao et al. [19] study the approximate k -NN query in relation database. However, both MEDRANK and [19] focus on much lower dimensional data (from 2 to 10).

Locality sensitive hashing is a well-known effective technique for approximate nearest neighbor search [1, 3], which ensures that the more similar of the data objects, the higher probability they are hashed into the same buckets. The near neighbor of the query can be found in the candidate bucket with high probability.

The space filling techniques convert high-dimensional data into one dimensional while preserving their similarity. One kind of space filling is Z -order curve, which is built by connecting the z -value of the points [20]. Thus the k -NN search for a query can be translated into range query on the z -values. Another space filling technique is Hilbert curve [21]. Based on Z -order curve and LSH, Tao et al. [4] propose *lsb-tree* for fast approximate nearest neighbor search in high dimensional space.

The random projection method of LSH is designed for approximately evaluating similarity between vectors. Recently, CompactProjection [5] employs it for the content-based image similarity search.

Recently, the data-aware hashing techniques are proposed which demand much less bits with the machine learning skill [6, 7, 22]. Semantic hashing [6] employs the Hamming distance of the compact binary codes for semantic similarity search. Self-taught hashing [7] proposes a two stage learning method for similarity search with much less bit length. Nevertheless, the preprocessing stage takes too much time and space for evaluating and storing k -NN graph and solving the LagMap problem. It's not proper for the data evolving setting and also its scalability is not well especially for the large volume of data.

Motivated by the above research works, the random projection learning method is proposed, which is proved to satisfy the *entropy maximizing* criterion nicely.

5 Conclusions

In this paper, a learning based framework for similarity search is studied. According to the framework, the data vectors are randomly projected into binary code firstly, and then the binary code are employed as the labels. The SVM classifiers are trained for *predicting* the binary code for the query. We prove that the binary code after random projection satisfying the *entropy maximizing* criterion. The approximate k -NN query is effectively evaluated within this framework. Experimental results show that our method achieves better performance comparing with the existing technique. Though the the query effectiveness is bit lower than the STH, RPL takes much smaller time and space complexity than STH in the preprocessing step and the query gains better performance, and this is very important especially in the data-intensive environment.

Acknowledgments. This work is supported by NSFC grants (No. 60925008 and No. 60903014), 973 program (No. 2010CB328106), Shanghai International Cooperation

Fund Project (Project No. 09530708400), Program for New Century Excellent Talents in University (No. NCET-10-0388) and Shanghai Leading Academic Discipline Project (No. B412).

References

1. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: STOC, pp. 604–613 (1998)
2. Charikar, M.S.: Similarity estimation techniques from rounding algorithms. In: STOC, pp. 380–388 (2002)
3. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In: FOCS, pp. 459–468. MIT, Cambridge (2006)
4. Tao, Y., Yi, K., Sheng, C., Kalnis, P.: Quality and efficiency in high dimensional nearest neighbor search. In: SIGMOD, pp. 563–576 (2009)
5. Min, K., Yang, L., Wright, J., Wu, L., Hua, X.S., Ma, Y.: Compact Projection: Simple and Efficient Near Neighbor Search with Practical Memory Requirements. In: CVPR, pp. 3477–3484 (2010)
6. Salakhutdinov, R., Hinton, G.: Semantic Hashing. *International Journal of Approximate Reasoning* 50(7), 969–978 (2009)
7. Zhang, D., Wang, J., Cai, D., Lu, J.: Self-taught hashing for fast similarity search. In: SIGIR, pp. 18–25 (2010)
8. Joachims, T.: Training linear SVMs in linear time. In: SIGKDD, pp. 217–226 (2006)
9. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines (2001), <http://www.csie.ntu.edu.tw/~cjlin/libsvm>
10. World Wide Knowledge Base project (2001), <http://www.cs.cmu.edu/~webkb/>
11. Reuters21578 (1999), <http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html>
12. Baeza-Yates, R.A., Ribeiro-Neto, B.A.: *Modern Information Retrieval*. Addison Wesley, Reading (1999)
13. Bentley, J.L.: Multidimensional binary search trees used for associative searching. *Communications of the ACM* 18(9), 517 (1975)
14. Guttman, A.: R-trees: a dynamic index structure for spatial searching. In: SIGMOD, pp. 47–57 (1984)
15. Beckmann, N., Kriegel, H.P., Schneider, R., Seeger, B.: The R*-tree: an efficient and robust access method for points and rectangles. *SIGMOD* 19(2), 322–331 (1990)
16. Weber, R., Schek, H.J., Blott, S.: A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In: VLDB, pp. 194–205 (1998)
17. Fagin, R., Kumar, R., Sivakumar, D.: Efficient similarity search and classification via rank aggregation. In: SIGMOD, pp. 301–312 (2003)
18. Fagin, R., Lotem, A., Naor, M.: Optimal aggregation algorithms for middleware. *Journal of Computer and System Sciences* 66(4), 614–656 (2003)
19. Yao, B., Li, F., Kumar, P.: k-nearest neighbor queries and knn-joins in large relational databases (almost) for free. In: ICDE, pp. 4–15 (2010)
20. Ramsak, F., Markl, V., Fenk, R., Zirkel, M., Elhardt, K., Bayer, R.: Integrating the UB-tree into a database system kernel. In: VLDB, pp. 263–272 (2000)
21. Liao, S., Lopez, M., Leutenegger, S.: High dimensional similarity search with space filling curves. In: ICDE, pp. 615–622 (2001)
22. Baluja, S., Covell, M.: Learning to hash: forgiving hash functions and applications. *Data Mining and Knowledge Discovery* 17(3), 402–430 (2008)
23. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. *NIPS* 21, 1753–1760 (2009)

Appendix: Theoretical Proof

The entropy H of a discrete random variable X with possible values x_1, \dots, x_n is defined as $H(X) = \sum_{i=1}^n p(x_i)I(x_i) = -\sum_{i=1}^n p(x_i) \log p(x_i)$, where $I(x_i)$ is the self-information of x_i .

For the binary random variable X , assume that $P(X = 1) = p$ and $P(X = 0) = 1 - p$, the entropy of X can be represented as Eq.7

$$H(X) = -p \log p - (1 - p) \log(1 - p), \tag{7}$$

when $p = 1/2$, Eq.7 attains its maximum value.

Semantic hashing [6] is an effective solution for similarity search with learning technique. For ensuring the search efficiency, an elegant semantic hashing should satisfy *entropy maximizing* criterion [6, 7]. The intuitive meaning of *entropy maximizing* criterion is that the datasets are uniformly represented by each bit, thus maximizing the information of each bit, i.e., bits are uncorrelated and each bit is expected to *fire* 50% [23]. Thus, we propose the following property for semantic hashing.

Property 1. Semantic hashing should satisfy *entropy maximizing* criterion to ensure efficiency, i.e., the chance of each bit to occur is 50% and each bit is uncorrelated.

In the following section, we prove that the binary code after random projection of LSH naturally satisfies the *entropy maximum* criterion. This is the key difference between our method and STH [7] in the generating binary code step. The latter needs considerable space and time consumption.

Let \mathbb{R}^d be a d -dimensions real data space and $\mathbf{u} \in \mathbb{R}^d$ is normalized into the unit vector, i.e., $\sum_{i=1}^d \mathbf{u}_i^2 = 1$. Suppose a random vector $\mathbf{r} \in \mathbb{R}^d$, \mathbf{r}_i is the i -th component of \mathbf{r} and chosen randomly from the standard normal distribution $N(0, 1)$, $i = 1, \dots, d$. Let $v = \mathbf{u} \cdot \mathbf{r}$ and v is random variable. Then the following lemma [8] holds.

Lemma 1. *Let random variable $v = \mathbf{u} \cdot \mathbf{r}$, then $v \sim N(0, 1)$.*

Proof. Since $v = \mathbf{u} \cdot \mathbf{r}$, thus $v = \sum_{i=1}^d \mathbf{u}_i \cdot \mathbf{r}_i$.

Since \mathbf{r}_i is a random variable and drawn randomly and independently from the standard normal distribution $N(0, 1)$, accordingly we have

$$\mathbf{u}_i \cdot \mathbf{r}_i \sim N(0, \mathbf{u}_i^2).$$

Thus, according to the property of standard normal distribution, the random variable $v \sim N(0, \sum_{i=1}^d \mathbf{u}_i^2) = N(0, 1)$. Hence the lemma is proved.

From lemma [8], the corollary [9] can be derived, which means that the binary code after random projection is uncorrelated.

Corollary 1. *The binary code after random projection is independent.*

Lemma 2. *Let $f(x) = \text{sgn}(x)$ be a function defined on the real data set \mathbb{R} , a random variable $v = \mathbf{u} \cdot \mathbf{r}$. Set random variable $v' = f(v)$, then $P(v' = 0) = P(v' = 1) = \frac{1}{2}$.*

The prove of Lemma 2 is omitted here due to space limit. The lemma 2 means that the 0 and 1 occur with the same probability in the signature vector. Hence, on the basis of corollary 1 and lemma 2, we have the following corollary.

Corollary 2. *The binary code of after random projection satisfies entropy maximizing criterion.*

Informed Prediction with Incremental Core-Based Friend Cycle Discovering*

Yue Wang**, Weijing Huang, Wei Chen,
Tengjiao Wang, and Dongqing Yang

Key Laboratory of High Confidence Software Technologies (Peking University),
Ministry of Education, China
{eecswangyue, pekingchenwei, tjwang, dqyang}@pku.edu.cn

Abstract. With more and more new social network services appearing, the volumes of data they created are continuous increasing at an astonishing speed. These data represent a snapshot of what real social network happening and evolving, and they contain the basic relationships and interacted behaviors among users. Core-based friend cycles are connected nodes around given "core node", and their interaction pattern with core node may reveal potential habits of users. This may be useful for online personalized advertising, online public opinion analysis, and other fields. To search core-based friend cycles by global method needs to scan the entire graph of social network every time, and thus its efficiency is low. This study (1) modeled the core-based friend cycles with core-based subgraphs;(2) provided algorithms to find structure and evolving interaction pattern of friend cycles around a given core node in online social network; (3) discussed and analyzed the design of incremental search algorithm theoretically; (4)applied the provided model to do informed prediction between node and its core-based friend cycles and received hit rate over 77.6%;(5) provided sufficient experiments and proven the newly proposed approach with good scalability and efficiency.

Keywords: social network , graph mining, evolving pattern, friend cycle.

1 Introduction

By the studies of real human life, one person may have many different friend cycles in his life, and he treat each one differently[1]. We observed similar phenomenon in the online social networks (OSN): people may behave quite different to informed his online friend cycles by receiving new information, usually, they prone to informed specific group of their friends intensively. This knowledge may be useful for online personalized advertising or online public opinion analysis,

* This work is supported by the National Key Technology R&D Program of China(No. 2009BAK63B08), National High Technology Research and Development Program of China('863' Program)(No.2009AA01Z150), China Postdoctoral Science Foundation.

** Corresponding author.

and administrator of OSN service providers or authorities may be interested about the different interaction patterns between specific nodes and their friend cycles on online social network. For example, they may ask questions about a specific node like: 1) how other nodes interact with a specific node? 2) How many different groups of friends are there related to specific node? Or 3) if a message were sent to one node, which related set of friends would be informed first?

To explore this kind of knowledge needs to analyze the topological structure of OSN, and it brings challenges: (1) efficiency problem: as the naive methods find the communities by processing information of all nodes in OSN, their processing efficiency is low; (2) result uncertainty: too many information retrieved from global methods may confuse the user. This paper provided notations about core-based friend cycles and studied friend cycles finding problem by focusing on "core node" and proposed algorithms to track their evolving interaction pattern. By considering the efficiency on huge volumes of OSN data, the proposed method was also designed in incremental style and thus can suit for large-scale evolving structure.

In order to track the online friend cycles, we implemented a system to crawl data from bulletin board system (BBS). Our system downloads data every day in following schema: (1) $post_info = \{ article_id, article_info, author_id, t_{post} \}$; (2) $reply_info = \{ article_id, replier, t_{reply} \}$. For every day running, the system get newly-added data in time span $[t_{start}, t_{end}]$, that means the created time of new records satisfy $t_{start} \leq (t_{post} \text{ or } t_{reply}) \leq t_{end}$, and this kind of data accumulate every day. Figure 1 visualized the results of core-based friend cycles found around a specific node during 15 days.

To study such knowledge, the main contribution of this paper includes: 1) proposed notations about core-based friend cycle and the former problem definitions; 2) provided global and incremental algorithms to extract the model with theoretical discussion for the reasons of incremental design, (3) applied the provided model to do informed prediction, and (4) gave sufficient experiments to discuss the effectiveness, efficiency and the parameters' impact of our model.

The remaining of this work is organized as follows: Chapter 2 discussed the related work recently; Chapter 3 gave the basic definitions and formalized the friend cycle finding problems; Chapter 4 proposed data structure of retrieved core-based friend cycles and searching algorithms for it; Chapter 5 gave sufficient experiments for the approach on practical data and applied our method to do informed prediction; Chapter 6 made a conclusion about the whole work and gave the future working directions.

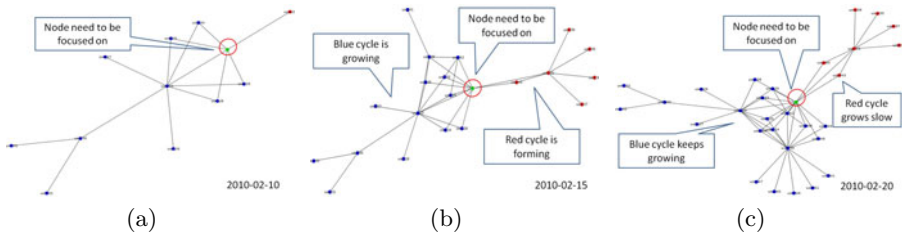


Fig. 1. Friend cycles evolving with data coming

2 Related Works

Mining the social network data requires the achievements from database technologies, graph theory, social psychology, physics, etc. Its primary applications include: business analysis, psychological guidance, epidemiology studying, public security, and anti-terrorist detection [2]. Technically, there 2 major way to analyze the social network data with these models: **(1) Social network analysis** This way allows people to consider social network as a huge stationary graph, and apply graph theory and other theories to analyze it. Christian Bird [3] used a small world model to analyze the social network established by email conversation, and got some useful information from the source CVS repositories. B. Gallagher et al. [4] proposed a notation "ghost edge" based on semi-supervised method to study the classification problem in sparse network. X. Ying and Xintao Wu et al. [5] provide a measurement to describe the graph random degree in social network and they applied this measurement to analyze the relationship between individual and communities. These method provided good advices about the topological analysis of the social network graph. **(2)Dynamic social network analysis** Barabasi et al. [6] analyzed the evolving pattern of social network by using theories of complex network. Davidsen et al. [7] studied the communication patterns between individuals in a social network, and the phenomenon of small communities emergence. EmilyM et al. [8] discussed the process of network topological structure changing with the time increasing. L. Zou et al. [9] tried to use a dynamic way to preserving network publication privacy. D. Liben-Nowell and J. Kleinberg [10] introduced a network linkage prediction problem in social network, and gave some corresponding methods to predict the future possible links between nodes. Nathan Eagle et. al [11] treated the mobile phone as personal sensor and used the data collected from telecom area to study rapid communication patterns between people. Golbeck [12] analyzed the data from popular social network services like Youtube, facebook, et. al, and he got some qualitative result. Yang Ning et. al [13] proposed a evolve graph model for events detection in data stream form. Most methods need directly applied on the entire network graph, and thus lacked efficiency when analyzing evolving structure of specific node in large-scale social network.

3 Problem Definition

This work uses the number of middlemen to establish the basic measurement between two strangers. And also, our work start the social network exploring from a "core node". Before discussing details of the approach, we give a formal definitions of the notations will be used in this paper.

Definition 1. Social Network Graph, a social network is a undirected graph $G=\langle V, E \rangle$, where V is the set of active users (users who post or reply at least one article in BBS) during time span $[t_s, t_e]$, and E is the set of interaction relationships between active users, $\forall e \in E, e=(v_a, v_b), v_a, v_b \in V (a \neq b)$.

Definition 2. Core-based Subgraph. Give vertex $v_{core} \in V$, the smallest number of middlemen between $v (\forall v \in V)$ and v_{core} is n , then the hop distance of v and v_{core} in G $Hop_G(v_{core}, v) = n + 1$. If we analyze G starting from v_{core} , then v_{core} is core vertex for this task, and a core-based subgraph $Sub(G)_i$ of v_{core} is a subgraph of G , and it must satisfy $Hop_{G'}(v_a, v_b) < h$, $Hop_G(v_{core}, v_b) < h$ and $Hop_G(v_{core}, v_a) < h (\forall v_a, v_b \in V_i (a \neq b))$. Where $G' = \langle V - v_{core}, E - e(v_{core}) \rangle$, and $e(v_{core})$ is the edges related to v_{core} .

Core node choosing: With the former definitions, the main task of this paper can be described as: given a core node v_{core} (in the BBS application, we chosen nodes who posted articles frequently as "core node", for these nodes may affect the behavior of the whole social network a lot), to track and get distribution principle of its friend cycles (core-based subgraph), and to analyze the interaction patterns of information based on this friend cycle model. And formally, the core-based subgraph discovering process can be described as below.

Problem 1. Core-based Subgraph Finding: Give a evolving social network graph $G = \langle V, E \rangle$, and vertex $v_{core} \in V$, to find $\forall Sub(G)_i \subset G$, for each $Sub(G)_i$ is core-based subgraph of v_{core} .

Under most practical circumstances: as the large-scale OSN data in a continue updating data stream form, to obtain $Hop(v_{core}, v)$ in a global view for everyday run is difficult. By following our system's data schema, we design an incremental way to discover the friend cycles, and formally, it can be described as:

Problem 2. Incremental Core-based Subgraph Finding: Given core vertex $v_{core} \in V$, and time window is Δt , Let $G_t = \langle V_t, E_t \rangle$ be the subset of G within the time span $[t, t + \Delta t]$ which satisfies $G = \bigcup_{t_0}^{t_{now}} G_t$ (t_0 is the starting time of data collect process and t_{now} is current time stamp); to find $\forall Sub(G)_i \subset \bigcup_{t_0}^{t_{now}} G_t$, $Sub(G)_i = \langle V_i, E_i \rangle$, for each $Sub(G)_i$ is core-based subgraph of v_{core} .

With the two basic problems defined upon, to predict information interaction patterns is to find out which friend cycle of vertex v will be informed first by giving v is informed. We assume that information would propagate from v_1 to v_2 if v_1 posted an article on BBS and v_2 replied it ($G = \langle V, E \rangle$, $v_1, v_2 \in V$, and $v_1 \neq v_2$). And the more, if $\forall v \in Sub(G)$ replied v_1 's article, then information will propagate from v_1 to $Sub(G)$. With this assumption, the informed prediction can be described as:

Problem 3. Informed prediction. Let $Sub(G)_i (i=0,1,2,...)$ be core-based subgraph of v_{core} in G , and its weight $W(Sub(G)_i) = \sum_{\forall v \in Sub(G)_i, t \in T} \frac{R}{v.t - t_s}$, where T is set of interaction time stamp between v and v_{core} , and R is a constant for scale adjusting. The informed probability $p_j = \frac{W(Sub(G)_j)}{\sum_{\forall Sub(G)_i \in G(v_{core})} W(Sub(G)_i)}$, where $G(v_{core})$ is set for all core-based subgraph of v_{core} in G . With the formal descriptions, we use friend cycle with the biggest informed probability as the estimation result of the predicted next informed friend cycle.

The following part will discuss approaches to find the core-based subgraph (friend cycle) and do informed prediction.

4 Core-Based Friend Cycle Searching Algorithms

Given a specific core vertex $v_{core}(v_{core} \in V)$, in social network graph $G = \langle V, E \rangle$, denote $Sub(G)_i^t$ is i -th core-based subgraph of v_{core} at time t . The core-based friend cycle searching's result can be denoted as a 2-tuple: $S_t(v_{core}) = \langle v_{core}, C_t = \{Sub(G)_0^t, Sub(G)_1^t, Sub(G)_2^t, \dots\} \rangle$ Where C_t is a snapshot for all core-based subgraphs of v_{core} at t ; If the context is specific, $S_t(v_{core})$ can simplify as S_t .

In the rest sections, we proposed two algorithms for the core-based friend cycle searching task: (1) global algorithm, and (2) incremental algorithm.

4.1 Global Friend Cycle Finding Algorithm

As our goal is to find the friend cycles around one core vertex, the basic steps to find core-based subgraphs globally is: 1) collect data D from online social network services; 2) establish a global graph G from D ; 3) build the topological structures by the interaction among vertexes (people) in G ; 4) check $\forall v \in G.V$ whether it satisfy the conditions of problem 1, and add v to v_{core} 's specific friend cycle. The pseudo-code of algorithm is listed in algorithm 1 gFC-find.

```

Data:  $v_{core}$ ,  $h$ ,  $D$  (all the data collected until time  $t$ );
Result:  $S$ -set= $\{S_0, S_1, S_2, \dots, S_t\}$ ;
begin
   $S \leftarrow \phi$ 
   $G \leftarrow \text{Graph-generate}(D)$ ; /* Generate graph  $G$  from  $D$  */
  TopBuild( $G$ ); /* Topological structure establish */
  foreach  $Vertex\ v \in G$  do
    |  $S_t \leftarrow \text{InsertVertex}(S_{t-1}, v, h)$ ; /*  $h$  is hop-distance threshold */
  end
end
Function InsertVertex( $S, v, h$ )
begin
   $i \leftarrow S.\text{CheckCycle}(v)$ ; /*  $v \in \text{cycle } C_i$  when  $\forall v_i \in C, \text{hop}(v_i, v) < h$  */
  if  $i \neq -1$  and  $\text{hop}(v_{core}, v) < h$  then Insert( $S.C_i, v$ );
  else if  $\text{hop}(v_{core}, v) < h$  then
    |  $C \leftarrow \text{newCycle}()$ ;  $C.\text{add}(v)$ ;  $S.\text{addCycle}(C)$ ;
  end
end

```

Algorithm 1. gFC-find

In gFC-find, we evoked a modified version of Dijkstra algorithm to generate the topological structure in the graph. And for the input parameters, h is the threshold for hop distance between the two vertexes in a same friend cycle (As the six degree separation theory [14], practically, we set $h \in [2, 4]$ in the experiments). In this algorithm, the function InsertVertex is actually a process of accumulated clustering. As the result of Dijkstra algorithm is optimal, it can find the shortest hop-distance between two vertexes. However because its complexity are $O(n^2)$,

when D is huge, the efficiency of gFC is low. The worst, for every new time stamp t, gFC-find needs to regenerate G and its topological structure. This make gFC-find's complexity is $O(n^2)$ too, and it can hardly be applied to a big network. Thus, we propose an incremental algorithm in the following.

4.2 Rapid Incremental Algorithm

Cast Away Strategy: Beside the $O(n^2)$ time complexity of gFC-find, too many vertexes may cause combinational exploding problem, that also reduce gFC-find's efficiency. To conquer this, we design a strategy to cast away some vertexes and edges when the new vertexes are loaded in order to keep an appropriate number of vertexes in G_t thus to control the time and space cost.

We use **shortest hop distance matrix** A to record the topological structure generated by Dijkstra algorithm, A is a matrix recording the shortest hop steps between every two vertex in graph G, and A is denoted as:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{pmatrix} \tag{1}$$

Where $\forall a_{ij} \in A$, a_{ij} represents the shortest hop distance for every pair of v_i and v_j in G, and $n=|G|$. When the dynamic graph $G=(V, E)$ needs to cast away vertexes, it must check the affection may be caused by casting away for $\forall v \in V$ one for each time, then cast away the least topological affection vertexes. We provide following methods to fulfill the rapid checking.

An eigenvector-based strategy. This strategy based on the eigenvector calculation on A, and its process can be described as follows: 1) get the shortest hop distance matrix A of G; 2) calculate the eigen values and the eigen vectors about A; 3) find the biggest eigen value and its corresponding eigen vector $eigv_k=[a_{1k}, a_{2k}, a_{3k}, \dots, a_{nk}]$ of A; 4) find the biggest a_{ik} ($1 \leq i \leq n$) in $[a_{1k}, a_{2k}, a_{3k}, \dots, a_{nk}]$, and cast away i-th vertex v_i in G. This eigenvector-based strategy can find out the least topological affection vertex through a single eigenvectors computation, we give following theorem to prove this strategy is effective.

Theorem 1. *Let A be the shortest hop distance matrix of graph G, $eigv_k=[a_{1k}, a_{2k}, a_{3k}, \dots, a_{nk}]$ is the corresponding eigenvector of the biggest eigen value about A, then the proposition of finding out the current least topological affection vertex $v \in G$ is equivalent to finding $v_i \in G$ ($1 \leq i \leq n$) where its correspondinge a_{ik} is the biggest value in $eigv_k$.*

Proof. As defined in former sections, $\forall a_{ij} \in A$ represents the shortest hop distance between v_i and v_j ($v_i, v_j \in G.V$). Suppose information propagate from v_i to v_j and the cost of information propagated at t step on v_i is $c_i^{(t)}$, the information propagate cost from v_i to v_j is denoted as $c_i^{(0)} \times a_{ij}$. Denote the average cost

for information propagating from v_i to $\forall v_j \in G.V$ ($i \neq j$) the next iterate step as $c_i^{(t+1)}$, then, $c_i^{(t+1)}$ can be obtained by equation 2.

$$c_i^{(t+1)} = \frac{\sum_{j=1}^n a_{ij} c_j^{(t)}}{n}, n = |A| \tag{2}$$

Information propagated from v_i to $\forall v_j \in G.V$. When $t \rightarrow \infty$, $c_i^{(t)}$ describes the information propagating difficulty from v_i to any other vertex in $G.V$, thus the bigger $c_i^{(t+1)}$ is, the more difficult information propagating from v_i to any other vertex will be. So the vertex with biggest $c_i^{(t+1)}$ is the least topological affection vertex in G . The information propagate costs for every vertex form a vector $V_{cost}^{(t)} = [c_1^{(t)}, c_2^{(t)}, \dots, c_n^{(t)}]^T$, and it can be calculated by equation 2.

$$V_{cost}^{(t)} = \frac{A \times V_{cost}^{(t-1)}}{n} = \frac{A^{t-1} \times V_{cost}^{(t_0)}}{n} \tag{3}$$

Suppose the isolate vertexes are deleted before this calculation, and A is a irreducible Matrix, then according to the Markov chain central limit theorem [15]: when $t \rightarrow \infty$, A^t will tend to a definite Matrix A_u (u means ultimate). Thus equation 3 can be transformed into equation 4.

$$V_{cost}^{(t)} = \frac{A_u \times V_{cost}^{(t_0)}}{n} \tag{4}$$

Thus when $t \rightarrow \infty$, $V_{cost}^{(t)}$ will tend to a constant ($V_{cost}^{(t+1)} = V_{cost}^{(t)} = V_c$). According to equations 3 and 4.

$$\lambda \times V_c = A \times V_c \tag{5}$$

Where λ is a constant for value adjusting. Thus, $n(n=|G|)$ -dimensional vector V_{cost} for all vertexes in $G.V$ is the eigenvector of A . And according to the concept of information propagated cost, current least topological affection vertex $v \in G$ will have the biggest cost in $\forall v_i \in G$. **Proof End**

iFC-find algorithm with the former cast-away strategy, we proposed an incremental core-based friend cycle finding algorithm in algorithm 2: iFC-find. In algorithm iFC-find, the functions Graph-generate, TopBuild, and InsertVertex is the same as the ones in gFC-find. The difference is iFC-find use ΔD in every Δt as source to establish a graph ΔG which was added to G_t for TopBuild, and apply eigenvector-based strategy to cast away operation. This makes iFC-find search core-based friend cycles of v_{core} incrementally without regenerate the whole G for every Δt . We provided more detail discussion about this castaway strategy in the experiment section.

5 Experiments and Discussion

5.1 Data Sets and Experiments Configuration

BBS Data: We use BBS data collected from sinaBBS and TecentBBS (2009-1-12 ~ 2010-07-26). And the target is to find different friend cycles of one specific node(account ID). **Actors Data:** To test the general use of the algorithms,

```

Data:  $v_{core}$ ,  $h$ ,  $G_0, \Delta D$  (data collected in  $\Delta t$ );
Result: S-set= $\{S_0, S_1, S_2, \dots, S_t\}$ ;
begin
   $S_0 \leftarrow \phi; G_t \leftarrow G_0;$ 
  for  $t \leftarrow 0; t < t_{now}; t \leftarrow t + \Delta t$  do
     $G_t \leftarrow \text{Graph-generate}(\Delta D, G_t);$ 
    if  $|G_t| > k$  then
      |  $G_t \leftarrow \text{VertexesCastAway}(G_t)$  /* do cast away operation */
    end
    TopBuild( $G_t$ ); /* Establish the topological structure of  $G_t$  */
    foreach Vertex  $v \in G_t.V$  do
      |  $S_t \leftarrow \text{InsertVertex}(S_t, v, h);$  /*  $h$  is hop-distance threshold */
    end
    S-set.add( $S_t$ );
  end
  return S-set;
end

Function VertexesCastAway( $G$ )
begin
  G-set  $\leftarrow \phi$ 
  A  $\leftarrow \text{getHopMatrix}(G);$ 
  eigenvalues  $\leftarrow \text{eig}(A);$  eigenvectors  $\leftarrow \text{eigv}(A);$ 
   $i \leftarrow \text{max}(\text{eigenvalues});$ 
  weights  $\leftarrow \text{eigenvectors}[i];$ 
  Find  $k \leftarrow \text{arg max}(\text{weights}(k))$ 
   $G' \leftarrow G.\text{castaway}(v_k)$ 
  return  $G'$ 
end

```

Algorithm 2. iFC-find

we also downloaded Actors data from IMDB [16], and extracted 72,371 actor records in schema actor_relation = {movie_name, leading_role, supporting_role, filmed_time} from year 2000 to 2012. And the target is try to find different friend cycles of one specific actor. Table 1 shows a summary of BBS data. All experiments are completed on a PC with a 2.93 GHZ CPU and 2 GB RAM. The prototype is implemented by C#, and all data stored in a PC with Microsoft SQL server 2005 database server.

Table 1. A Summary of Data Sets

	rows	poster no.	replier no.
Sina	3,046,814	28,359	127,088
Tencent	311,483	4,712	107,776
Actors	72,374	7,392	39,105

5.2 Effectiveness Experiments

We selected the top 5 articles posted authors (for each posted over 47 articles in TencentBBS) to analyze gFC. As gFC’s efficiency is rather low, we chosen the active time span (continue posting or replying articles for 5 days) from 2010-1-28 to 2010-2-28 for experiment, and set hop=2. As the result shown in figure 2: the average number of friend cycles is 1.4.

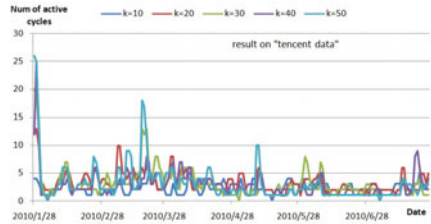
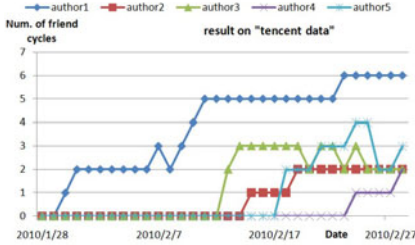


Fig. 2. Friend cycles of top 5 posters **Fig. 3.** Active cycles results of diff. k by iFC

iFC analysis experiment. We used all the authors in data set "Tencent BBS" as the vertexes of graph $G(|V| = 110,002)$. As our data is extracted from BBS, by considering the post-reply time span of people, we treated friend cycles c_i with $w(c_i) > 0.01$ as the active friend cycle. According to the definition, $w(c_i) > 0.01$ here means the average interaction happened between 2 people in c_i is within 1,000 seconds. Figure 3 shows the active friend cycles found result by iFC algorithm, and the number of this author’s average active friend cycles is 3.083. This result shows that although people may interact with many people in different friend cycles everyday, but the main contact friend cycles are few.

Informed prediction application. Figure 4 shows the weight varying pattern of 4 big cycles of "author1", and their weights is 0.40955, 0.00127, 0.00025, 0.00011 after 2010-7-1. This means friend cycle1 may be author1’s most possible information receiver, and probability $p = \frac{0.4095}{0.4095 + 0.0012 + 0.0002 + 0.0001} = 99.629\%$ (the ellipsis represents weights of other cycles). That means when author1 post a new article, cycle1 may be informed with probability 99.629%. We tested the accuracy of prediction by: suppose the biggest weighted friend cycle is "cycle1" at time t, then at t+1, if the next vertex v_n was inserted into cycle1, the prediction is hit. And hit rate R_{hit} can be denotes as: $R_{hit} = \frac{N_{hit}}{N_{in}}$; Where N_{in} means total number of inserted vertexes within time [t, t+1], and N_{hit} means the number of vertexes inserted into the friend cycle practically.

Figure 5 shows hit rate results with time windows $W = 24$ and 48 hours, and average hit rate exceeds 77.6%. With such result, we found people may prone to informed few specific friends group of them intensively during given period.

Results found on Actors Data. We applied iFC on Actors data [16] set with hop=3, k=20, and chosen "Goldblum, Jeff" as core vertex. The result of

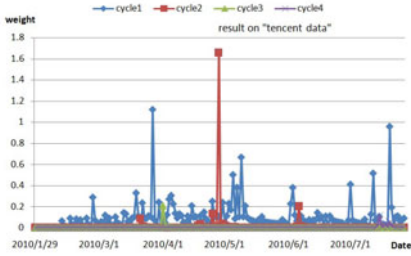


Fig. 4. friend cycle evolving pattern

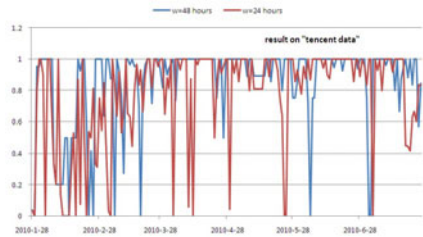


Fig. 5. R_{hit} result for iFC

”Goldblum, Jeff” is shown in figure 6, and iFC found 3 major friend cycles: $c_1 = \{Goldwyn, Tony; Desotell, Michael A.; Barber, Lance (I); Cashen, Brian; Bricchetto, Gary\}$, $c_2 = \{Fogarty, Brud; Dickerson, Briana; Goldston, Gregg; Capadona, Tom; Fogarty, Brud\}$; $c_3 = \{Embry, Ethan; Ansley, Zachary; Banks, Linden; Boileau, Daniel...(more actors)\}$. By checking the members’ ”known for” result in IMDB [16] for each cycle, we found that actors in c_1, c_2, c_3 are tend to play in ”comedy, {crime, thriller}, {sci-fi, thriller, adventure }” styles of movies respectively. Following this pattern, if $v (v \in c_1)$ and Goldblum Jeff cooperate in one movie, this movie with great possibility will be a comedy!

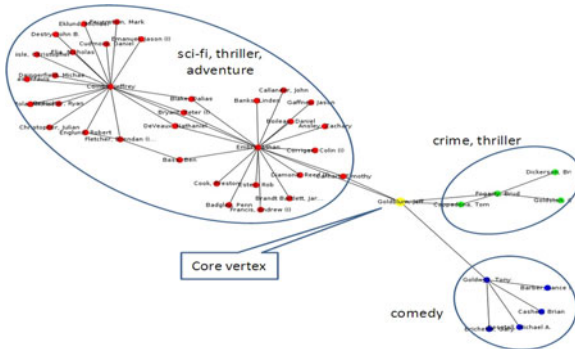


Fig. 6. Friend cycles of Goldblum, Jeff

5.3 Scalability Experiments

Time cost comparison experiment. We ran iFC algorithm 180 times on Tencent BBS data(32,340 rows) with $k=10 \sim 50$. As it is shown in figure 7, time cost is increasing with k , and the average time costs are: $\bar{t}_{k=10} = 2.11s$, $\bar{t}_{k=20} = 3.89s$, $\bar{t}_{k=30} = 8.20s$, $\bar{t}_{k=40} = 21.53s$, $\bar{t}_{k=50} = 45.00s$. We also compared the efficiency of iFC and gFC, as the result shown in the figure 8, when processing data over 2,281 rows, gFC costed over 800 seconds for a single run.

Space cost comparison experiment. The space comparison results are shown in figure 9, as iFC keep an relative constant vertexes in RAM, its space cost is within 2,000KB, and gFC cost over 8,000KB space for 2,356 rows data.

Scalability of iFC. We ran iFC on Sina data with $k=20$ iteratively for 82 times to simulate a extreme communication happened environment (totally over 2.66 million interaction happened, and almost 200 new happened every day). As the result in figure 10, the average process time everyday is 7,899 ms.

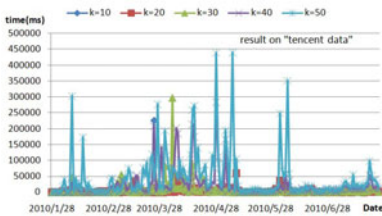


Fig. 7. Time cost for different k of iFC

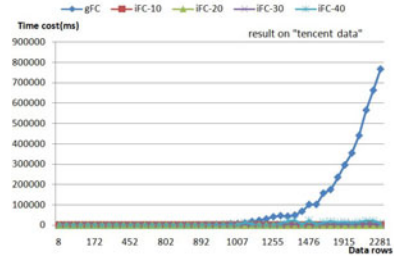


Fig. 8. Efficiency of iFC v.s. gFC

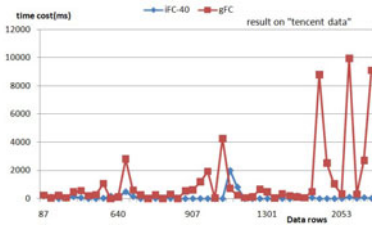


Fig. 9. Space of iFC & gFC

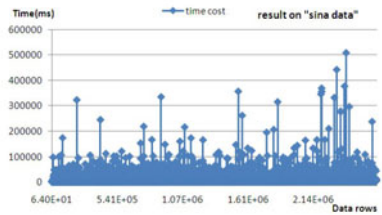


Fig. 10. Time cost for iFC on 2.66 mil. data

6 Conclusions

In this study, we proposed methods to analyze the core-based friend cycles around specific node and do informed prediction in OSN, and our contributions are: (1) modeled core-based friend cycles in OSN; (2) proposed a global algorithm gFC to find core-based friend cycles; (3) proposed a cast-away strategy with an efficient incremental algorithm iFC; (4) provided sufficient experiments on real data to discuss effectiveness and scalability of algorithms, and do informed prediction. Our future work will focus on studying information propagate dynamics around core node and generalizing our methods on micro-blog applications.

References

1. Hill, R.A., Dunbar, R.I.M.: Social network size in humans. *Human Nature* 14, 53–72 (2003)
2. Wu, X., Ying, X., Liu, K., Chen, L.: A Survey of Algorithms for Privacy-Preservation of Graphs and Social Networks. In: Aggarwal, C.C., Wang, H. (eds.) *Managing and Mining Graph Data*. Kluwer Academic Publishers, Dordrecht (August 2009)
3. Bird, C., Gourley, A., Devanbu, P., Gertz, M., Swaminathan, A.: Mining Email Social Networks. In: *Proceedings of the Third International Workshop on Mining Software Repositories*, pp. 137–143. ACM, New York (2006)
4. Gallagher, B., Tong, H., Eliassi-Rad, T., Faloutsos, C.: Using ghost edges for classification in sparsely labeled networks. In: *Proc. of the 14th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining* (2008)
5. Ying, X., Wu, X.: Graph generation with prescribed feature constraints. In: *Proc. of the 9th SIAM Conference on Data Mining* (2009)
6. Barabasi, A.L., Jeong, H., Néda, Z., Ravasz, E., Schubert, A., Vicsek, T.: Evolution of the social network of scientific collaboration. *Physica A* 311(3-4), 590–614 (2002)
7. Davidsen, J., Ebel, H., Bornholdt, S.: Emergence of a small world from local interactions: Modeling acquaintance networks. *Physical Review Letters* 88 (128701) (2002)
8. Jin, E.M., Girvan, M., Newman, M.E.J.: The structure of growing social networks. *Physical Review Letters* E 64(046132) (2001)
9. Zou, L., Chen, L., Ozsu, M.T.: K-automorphism: A general framework for privacy preserving network publication. In *Proc. of 35th International Conference on Very Large Data Base* (2009)
10. Liben-Nowell, D., Kleinberg, J.: The Link Prediction Problem for Social Networks. In: *Proc. 12th International Conference on Information and Knowledge Management, CIKM* (2003)
11. Eagle, N., Pentland, A.: Reality mining: sensing complex social systems. *Personal and Ubiquitous Computing* 10(4), 255–268 (2006)
12. Golbeck, J.: The dynamics of online social networks: Membership, relationships, and change. *ACM Trans. Inform. Stud.* (2006a)
13. Yang, N., Tang, C.-J., Wang, Y., Chen, Y., Zheng, J.-L.: TDCA: A Novel Clustering Algorithm on Data Stream with Skew Distribution Based on Temporal Density. *Journal of Software* 5 (2010)
14. Newman, M., Barabasi, A.-L., Watts, D.J.: *The Structure and Dynamics of Networks*. Princeton University Press, Princeton (2006)
15. Nagaev, S.V.: Some limit theorems for stationary Markov chains. *Theory of Probability and its Applications* 11(4), 378–406 (1957)
16. <http://www.imdb.com>

Early Prediction of Temporal Sequences Based on Information Transfer*

Ning Yang, Jian Peng, Yu Chen, and Changjie Tang

College of Computer Science, Sichuan University, Chengdu, China
{yangning, jiangpeng, yuchen, cjtang}@scu.edu.cn

Abstract. In recent years, early prediction for ongoing sequences has been more and more valuable in a large variety of time-critical applications which demand to classify an ongoing sequence in its early stage. There are two challenging issues in early prediction, i.e. why an ongoing sequence is early predictable and how to reasonably determine the parameter $k_{optimal}$, the minimum number of elements that must be observed before an accurate classification can be made. To address these issues, this paper investigates the kinetic regularity of the information transfer in sequence data set. As a result, a new concept of Accumulatively Transferred Information (ATI) and its kinetic model in early predictable sequences are proposed. This model shows that the information transfer in early predictable sequences follows *Inverse Heavy-tail Distribution*(IHD), and the most uncertainty of an early predictable sequence is eliminated by only few of its preceding elements, which is exactly the intrinsic and theoretically sound ground of the feasibility of early prediction. Based on the kinetic model, a heuristic algorithm is proposed to learn the parameter $k_{optimal}$. The experiments are conducted on real data sets and the results validate the reasonableness and effectiveness of the proposed theory and algorithm.

Keywords: Early Prediction, Temporal Sequence, Information Transfer.

1 Introduction

With the increase of temporal data, there is a growing demand of early prediction that classifies an ongoing sequence as early as possible, preferably after only few of its preceding elements occurred. For example, it is always desirable to predict an intrusion as soon as few preceding events of a sequence happen. For another example, early prediction of a disease makes early treatment possible, which is extremely important for patients. The traditional methods for sequence classification, such as support vector machines (SVM) and artificial neural networks (ANN) [1,2], cannot be applied straight to early prediction for an ongoing

* Supported by the Basic Research Foundation for Central Universities under Grant No. 2010SCU11053.

sequence since they are mostly whole-sequence based, i.e. they presuppose that all the elements of a sequence to be classified are observed in advance.

Essentially, there are two key issues that should be addressed for successful early prediction. One issue is to identify the ongoing sequences that are early predictable. The problem behind the curtain is to seek the underlying reason why the early prediction is possible. The other issue is to reasonably determine the minimum number of the preceding elements that must be observed before an ongoing sequence can be classified with an acceptable accuracy. Obviously, the early prediction will be no sense if the number is too great.

This paper addresses these two key issues and reveals the intrinsic and theoretically sound ground of early prediction by taking a new approach rooted in information theory. Our idea includes two points. The first is to regard an ongoing sequence as a flow where the information is accumulatively transferred from one element to the next, and meanwhile the remaining uncertainty of the sequence decreases. It is obvious that the less the remaining uncertainty, the higher the determinacy of a sequence. The second point is that a sequence is early predictable if and only if its most information is accumulatively transferred by only few of its preceding elements. In other words, the determinacy of a early predictable sequence only depends on its few preceding elements, which makes it possible to classify it in early stage, even when only few preceding elements of it occurred. The main contributions of this paper can be summarized as the following:

- (1) A new concept of Accumulative Transferred Information (shortened as ATI) is proposed to characterize the information movement in a sequence. ATI is defined based on entropy and conditional entropy of a sequence and represents the total amount of information having been transferred until a specified element occurred which quantifies the eliminated uncertainty.
- (2) The kinetic regularity of ATI of early predictable sequences is investigated on real data sets. The results show that the kinetic regularity of ATI can be approximated by an inverse heavy-tail distribution, which reveals that the most information of an early predictable sequence is transferred, and consequently the most uncertainty is eliminated, by only few of preceding elements.
- (3) A heuristic algorithm, Learn K , is suggested to learn the parameter $k_{optimal}$, the minimum number of preceding elements that must be observed before an ongoing sequence can be accurately predicted. Learn K tries to locate the point on which the variation of the slope of the ATI's curve increases abruptly.
- (4) The feasibility and effectiveness of our theory and algorithm are verified on real data sets.

The rest of this paper is organized as follows: Section 2 reviews the related work. Section 3 introduces the concept of ATI of a sequence and investigates its kinetic model. Section 4 presents the algorithm Learn K to search the parameter $k_{optimal}$. Section 5 introduces the approach to build a classifier for the early

prediction for an ongoing sequence. Section 6 shows the experimental evaluation and Section 7 concludes.

2 Related Work

Three domains are relevant to our work, namely sequential pattern mining, sequence classification and early prediction for sequences.

Sequential Pattern Mining. The goal of sequential pattern mining is to discover the frequently occurring ordered elements or subsequences as patterns [1]. There have been several typical algorithms including GSP [3], PrefixSpan [4], Spam [5], Spade [6] and SeqIndex [7]. Additionally, the sequence alignment has been investigated as the extension of sequence analysis [8,9].

Sequence classification. Sequence classification is an extensively studied problem. Some classifiers are built based on frequent sequential patterns [2, 10, 11]. Some other classifiers are built with artificial neural networks (ANN) [12,13]. In addition, [14,15,16] propose a series of classification methods by the combination of the support vector machines (SVM) and feature selection.

Early Prediction for Sequences. [17] introduces the concept of early prediction. In [17], early prediction is achieved through the linear combination of features, which can tolerate the miss of some features but suffers the deterioration of accuracy. [18] studies the construction of sequence classifier towards early prediction. The methods proposed by [18] take a parameter p_0 of the expected accuracy as the input specified beforehand, and minimize the length of the prefix that the classifier must check for the prediction, and strongly depend on the feature selection.

The existing work can not answer the question why a sequence is early predictable. This paper takes a strategy fundamentally distinct from them, which first investigates the underlying reason why a sequence is early predictable from the perspective of the information movement in a sequence, and derives the algorithm logically and naturally to adaptively learn the parameter $k_{optimal}$, the minimum number of preceding elements that should be checked before an accurate early prediction is made.

3 Kinetic Model of Information Transfer

This section investigates the kinetic model of the information transfer of the predictable sequences. Let X_n be a set of n -length sequences, where each sequence $x_n = (e_1, e_2, \dots, e_n)$ and $e_i (i = 1, 2, \dots, n)$ is an element. Let X_i be the set of sequences where each sequence consists of the first i elements of the corresponding x_n in X_n , and obviously $|X_n| = |X_i|$. The base of logarithms is 2.

As stated in section 1, the uncertainty of a sequence decreases due to the information is accumulatively transferred along successive elements e_i, e_{i+1} . So, how

much total information is transferred and how much uncertainty is consequently eliminated up till the i th element e_i ? Accumulatively Transferred Information is exactly the key to these questions.

Definition 1. Given X_n , the **Accumulatively Transferred Information** (shortened as **ATI**) up till e_i , denoted by A_i , is defined as $A_i = H(X_n) - H(X_n|X_i)$.

The entropy $H(X_n)$ evaluates the uncertainty of a whole sequence when no element is observed, and the conditional entropy $H(X_n|X_i)$ evaluates the remaining uncertainty of a sequence after its i th element is checked. Therefore the difference between $H(X_n)$ and $H(X_n|X_i)$ naturally indicates the amount of information having been transferred up till i th element. The following proposition states that ATI increases monotonically.

Proposition 1. Given X_n , the ATI increases monotonically with the increase of i , i.e. $A_1 < A_2 < \dots < A_n$.

Proof. $H(X_n) > H(X_n|X_i)$ and $H(X_n|X_i)$ decreases monotonically with the increase of i because the conditional variable reduces the entropy [19]. So according to Definition 1, $A_{i-1} = H(X_n) - H(X_n|X_{i-1}) < A_i = H(X_n) - H(X_n|X_i)$. \square

Proposition 1 reflects the fact that the uncertainty of a sequence decreases along with the transfer of information, while the following proposition 2 provides a convenient way to calculate AIT beyond its definition.

Proposition 2. The following formula holds

$$A_i = \sum_{x_n \in X_n} \frac{|x_n|}{|X_n|} \log \frac{|X_n|}{|x_i|}, \tag{1}$$

where $|x_n|$ is the number of occurrences of x_n in X_n , $|x_i|$ is the number of occurrences of x_i in X_i , and $|x_1|$ is the number of occurrences of x_1 in X_1 .

Proof. By the definition 1 and the definition of conditional entropy [19], we have

$$\begin{aligned} A_i &= H(X_n) - H(X_n|X_i) \\ &= - \sum_{x_n \in X_n} p(x_n) \log p(x_n) + \sum_{x_i \in X_i} \sum_{x_n \in X_n} p(x_n, x_i) \log p(x_n|x_i) \\ &= - \sum_{x_n \in X_n} \left(\sum_{x_i \in X_i} p(x_n, x_i) \log p(x_n|x_i) - p(x_n) \log p(x_n) \right). \end{aligned}$$

Note that x_i is the prefix of x_n , so

$$\begin{aligned} p(x_n, x_i) &= p(x_n) = \frac{|x_n|}{|X_n|}, \\ p(x_n | x_i) &= \frac{|x_n|}{|x_i|}. \end{aligned}$$

Consequently,

$$A_i = \sum_{x_n \in X_n} \left(\sum_{x_i \in X_i} \frac{|x_n|}{|X_n|} \log \frac{|x_n|}{|x_i|} - \frac{|x_n|}{|X_n|} \log \frac{|x_n|}{|X_n|} \right).$$

Each $x_n \in X_n$ has only one i -length prefix x_i , so

$$A_i = \sum_{x_n \in X_n} \frac{|x_n|}{|X_n|} \left(\log \frac{|x_n|}{|x_i|} - \log \frac{|x_n|}{|X_n|} \right) = \sum_{x_n \in X_n} \frac{|x_n|}{|X_n|} \log \frac{|X_n|}{|x_i|}. \quad \square$$

Proposition 2 enables us to directly compute ATI without the knowledge of conditional entropy $H(X_n|X_i)$, as shown in the following example.

Example 1. Given the set X_5 listed in table 1, compute A_3 .

Table 1. The set X_5, X_3

X_5	X_3
a, b, c, d, e	a, b, c
a, b, c, f, g	a, b, c
a, b, c, d, e	a, b, c
a, b, c, f, g	a, b, c
c, d, e, j, q	c, d, e
a, b, c, d, e	a, b, c
c, d, e, f, h	c, d, e
b, c, d, g, h	b, c, d
c, d, e, j, q	c, d, e
b, c, d, e, f	b, c, d

In X_5 , $|(a, b, c, d, e)| = 3$, $|(a, b, c, f, g)| = 2$, $|(c, d, e, j, q)| = 2$, $|(c, d, e, f, h)| = 1$, $|(b, c, d, g, h)| = 1$, $|(b, c, d, e, f)| = 1$. In X_3 , $|(a, b, c)| = 5$, $|(c, d, e)| = 3$, $|(b, c, d)| = 2$. Then

when $x_5 = (a, b, c, d, e), x_3 = (a, b, c), \frac{|x_5|}{|X_5|} \log \frac{|X_5|}{|x_3|} = \frac{3}{10} \log \frac{10}{5} = 0.30$,

when $x_5 = (a, b, c, f, g), x_3 = (a, b, c), \frac{|x_5|}{|X_5|} \log \frac{|X_5|}{|x_3|} = \frac{2}{10} \log \frac{10}{5} = 0.20$,

when $x_5 = (c, d, e, j, q), x_3 = (c, d, e), \frac{|x_5|}{|X_5|} \log \frac{|X_5|}{|x_3|} = \frac{2}{10} \log \frac{10}{3} = 0.35$,

when $x_5 = (c, d, e, f, h), x_3 = (c, d, e), \frac{|x_5|}{|X_5|} \log \frac{|X_5|}{|x_3|} = \frac{1}{10} \log \frac{10}{3} = 0.17$,

when $x_5 = (b, c, d, g, h), x_3 = (b, c, d), \frac{|x_5|}{|X_5|} \log \frac{|X_5|}{|x_3|} = \frac{1}{10} \log \frac{10}{2} = 0.16$,

when $x_5 = (b, c, d, e, f), x_3 = (b, c, d), \frac{|x_5|}{|X_5|} \log \frac{|X_5|}{|x_3|} = \frac{1}{10} \log \frac{10}{2} = 0.16$.

So, according to the proposition 2,

$$A_3 = \sum_{x_5 \in X_5} \frac{|x_5|}{|X_5|} \log \frac{|X_5|}{|x_3|} = 1.34. \quad \square$$

Before investigating the kinetic model of ATI, we firstly use $H(X_n)$ to normalize ATI to avoid the influence from the different scales of data sets. The Normalized-ATI (shortened as NATI) is defined as the following.

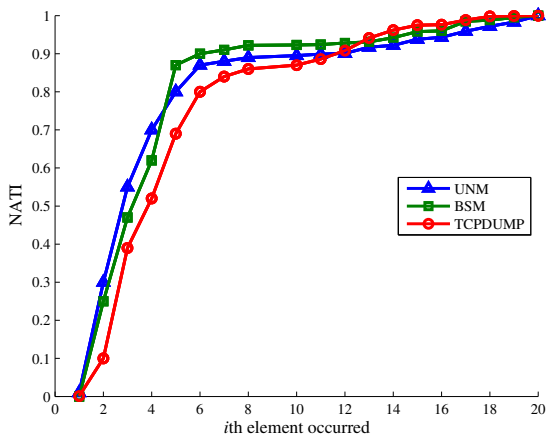


Fig. 1. The Kinetic Model of Information Transfer

Definition 2. The **Normalized-ATI** (shortened as **NATI**) up till *i*th element e_i , denoted by a_i , is defined as $a_i = \frac{|A_i|}{|A_0|}$, where $A_0 = H(X_n)$ and A_i is the ATI up till e_i .

We explore three real data sets, the University of New Mexico sendmail data (shortened as UNM) [20], MIT Lincoln Lab BSM sendmail data (shortened as BSM) and tcpdump data (shortened as TCPDUMP) [21]. For each data set we compute and depict the curve of NATI over a_1 to a_{20} in Fig.1. Fig.1 shows that the curves increase rapidly before a preceding element (no later than the 7th element) and become stable after it, which implies that the most information is accumulatively transferred by only few preceding elements of the sequences. This discovery inspires the following hypothesis about the relationship between the early prediction and the kinetic model of ATI.

Hypothesis. A n -length sequence can be predicted in early stage if and only if the kinetic regularity of its NATI can be approximately modeled by the following equation:

$$a_i = 1 - i^{-b}, \tag{2}$$

where $i = 1, 2, \dots, n$, and b is the accumulative factor.

We call the formula (2) *Inverse Heavy-tail Distribution* (shortened as IHD). Fig.2 illustrates several examples of IHD and shows the dependency of the information transfer speed on the accumulative factor b . It is easy to see that the greater accumulative factor, the faster transfer of information. In practice, the accumulative factor is often restricted in the real interval of $(1,2]$. In section 6, we will verify that the kinetic regularity of ATI modeled by formula (2) is exactly the underlying ground of early prediction.

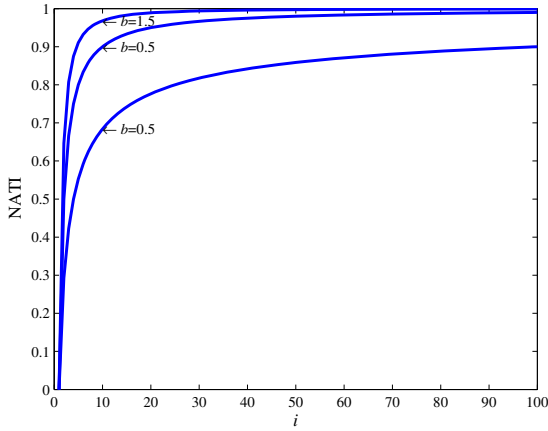


Fig. 2. Dependency of the information transfer speed on the accumulative factor b

4 Learning Parameter $k_{optimal}$

The conclusion of section 3 implies that the most uncertainty is eliminated by preceding few elements of a sequence if it is predictable, so the next question is how to determine the parameter $k_{optimal}$, the minimum number of elements that should be examined before an accurate prediction for an ongoing sequence is made. The value of $k_{optimal}$ should be small enough to make early prediction meaningful as well as large enough to ensure the accuracy. This section proposes a heuristic and iterative algorithm to learn $k_{optimal}$ which finds an optimal tradeoff between the two contradictory objectives.

By studying Fig.1 and Fig.2 more carefully, it can be found that the $k_{optimal}$ th element is exactly the point around which the slope of the NATI curve varies sharply. We call such point *Slope Change Point* and define it formally as follows.

Definition 3. *The Slope Change Point (shortened as SCP) of a function curve is the point on which the variation rate of slope is maximum.*

So the learning of parameter $k_{optimal}$ is reduced to the search of the SCP. Note that what the definition 3 emphasizes is the variation rate of the slope instead of the absolute difference. Thus it is the second rather than the first difference of slopes that should be used.

Given the curve of NATI, the key point of the algorithm to learn parameter $k_{optimal}$ is to separately compute s_{ir} which is the slope of the right line segment of the point (i, a_i) , and s_{il} which is the slope of the left line segment of that point. These two segments are linearly fitted by w points before and after the point (i, a_i) respectively, where w is the width of fitting window. Then $\Delta s_i = s_{ir} - s_{il}$ is the first difference of slopes, and $\Delta^2 s_i = \Delta s_i - \Delta s_{i-1}$ is the second difference of slopes. The $\Delta^2 s_i$ evaluates precisely the acceleration of slope

variation around the point (i, a_i) . Therefore according to definition 3, the desirable value of $k_{optimal}$ is exactly the sequence number i on which $\Delta^2 s_i$ achieves the maximum, i.e.

$$k_{optimal} = \arg \max_i \Delta^2 s_i, w \leq i \leq n - w, \tag{3}$$

where n is the length of the sequence. The details of algorithm to learn parameter $k_{optimal}$ is given in Algorithm 1.

Algorithm 1. Learn $K(X_n, W)$

Require: The set of n -length sequences, X_n ; The max width of fitting window, W ;

Ensure: $k_{optimal}$;

- 1: compute NATIs over $i = 1, \dots, n$ and store them into $a[1 \dots n]$; // by proposition 2
 - 2: **for** $i = 1 + W$ to $n - W$ **do**
 - 3: **for** $w = 2$ to W **do**
 - 4: make linear regression of points $(x, a[x]), x = i - w, \dots, i$;
 - 5: store the regression coefficient into $l(w)$;
 - 6: make linear regression of points $(x, a[x]), x = i, \dots, i + w$;
 - 7: store the regression coefficient into $r(w)$;
 - 8: **end for**
 - 9: $\sigma = \sum_{w=2}^W w^2$;
 - 10: $s_{il} = \sum_{w=2}^W \frac{w^2 l(w)}{\sigma}$; $s_{ir} = \sum_{w=2}^W \frac{w^2 r(w)}{\sigma}$;
 - 11: $\Delta[i] = s_{il} - s_{ir}$;
 - 12: **end for**
 - 13: **for** $i = 2$ to length of $\Delta[]$ **do**
 - 14: $\Delta^2[i] = \Delta[i] - \Delta[i - 1]$; // compute second difference
 - 15: **end for**
 - 16: $\Delta^2[u] = \max(\Delta^2[1 \dots \text{length of } \Delta^2])$;
 - 17: return $k_{optimal} = u$; // by formula (2)
-

The trick here is the choice of the width of the fitting window w . The linear regression of successive w points tells much more about the local slope if w is small, otherwise the global slope if w is great. Our solution is to set the argument W with the value of 4 in practice. Line 3 ~ 8 compute the linear regressions with $w = 2, 3, 4$, and the coefficients are respectively stored in $l(w)$ and $r(w)$, $w = 2, 3, 4$. The final slope s_{ir} is the average of $l(w)$ weighted by w^2 , and s_{il} can be calculated in the same way instead using $r(w)$ (Line 9 ~ 10). At last, the index of the maximum element of $\Delta^2[]$, u , is returned as result (Line 16, 17).

5 The Construction of Classifier

The first $k_{optimal}$ elements $(e_1, e_2, \dots, e_{k_{optimal}})$ can be regarded as features of x_n . So the early prediction is completed by a classifier with the following form:

$$y = f(e_1, e_2, \dots, e_k) \tag{4}$$

where $f(\cdot)$ is the classifier and y is the class label.

Since the element e_i is discrete and categorical, we apply SLIPPER algorithm [22] to the training data to build the classifier. SLIPPER is a simple, fast and effective rule learner which generates a rule set of compact and comprehensible by repeatedly boosting a simple and greedy rule-builder. In addition to its simplicity and effectiveness, SLIPPER is highly scalable since its complexity of $O(n \log n)$, where n is the number of data items. So it is very suitable for the task of early prediction.

It should be noted that there exists a few of qualified classification algorithms besides SLIPPER. We choose SLIPPER simply for its facilitation and scalability.

6 Experimental Evaluation

This section presents the experimental results evaluating our theory and methods on two real data sets, namely SPLICE and PROMOTER, which are all both available at UCI machine learning repository [23]. SPLICE and PROMOTER are two collections of gene sequences with length of 66 and 60 respectively, and each sequence is associated with a class label. In the experiments, we randomly choose the first 80% of each data set as the training data and the last 20% as the testing data. All the experiments are conducted on a PC with a Intel 2.2GHz CPU and 2GB main memory. The algorithm LearnK is implemented in Matlab R2007b and the classifiers based on SLIPPER are implemented in C.

We first compute the NATI for each data set, and the results are depicted in Fig.3. As we expected, the NATI of each data set approximately obeys the inverse heavy-tail distribution introduced in section 3, which means that the early predictions for the sequences in SPLICE and PROMOTER are feasible. From Fig.3, it can be observed that the NATI curves of PROMOTER and SPLICE

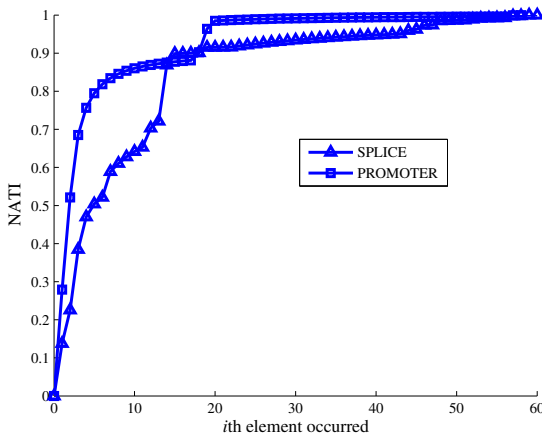


Fig. 3. The NATI of SPLICE and PROMOTER

come close to one around 25th element and 15th element respectively, so intuitively the $k_{optimal}$ of SPLICE and PROMOTER should be around 25 and 16 respectively.

As the second step of the experiment, we apply the algorithm LearnK proposed in section 4 onto the data sets to learn $k_{optimal}$. The results are $k_{optimal} = 16$ for SPLICE and $k_{optimal} = 23$ for PROMOTER, which are consistent with the observation on the Fig.3.

At the third step of the experiment, we apply the algorithm SLIPPER [22] on the training data of each data set to build the classifiers which take the first $k_{optimal}$ elements as input features, and then test them on the testing data. To compare the performance between our classifiers and other ones with different values of k , we additionally build a series of classifiers for each data set with k from 1 to 30. Fig.4 shows the misclassification rate on different data sets. Several discussions on Fig.4 are as follows.

1. The misclassification rate monotonically decreases with the increase of k . This indicates that the more elements observed, the higher deterministic the sequence, which precisely evidences the first point of our idea introduced in section 1 that the information is accumulatively transferred among the elements of an ongoing sequence and the remaining uncertainty is being gradually eliminated as the information is transferred from one element to the next.
2. The misclassification rate declines sharply before $k_{optimal}$, and begins to gently approach to zero after $k_{optimal}$, which coincides with the trend of the NATI curves in Figure 3 where the NATI increases rapidly before $k_{optimal}$ and begins to gently approach to one after $k_{optimal}$. This indicates that the ATI really imposes influence on remaining uncertainty of a sequence, and it is reasonable to use the formula (2) as the criterion for the construction of classifier.

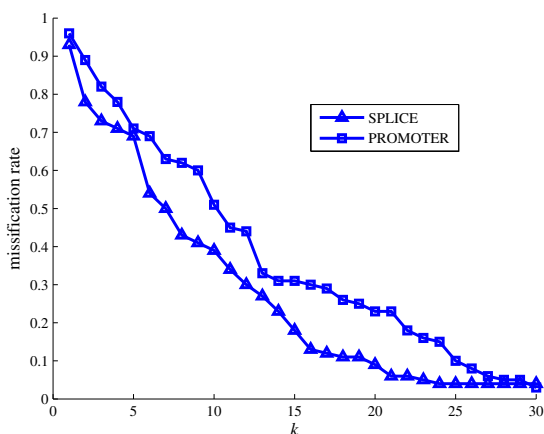


Fig. 4. The variation of misclassification rate on different k

3. The value of $k_{optimal}$ is far smaller than the whole length of a sequence (66 and 60, respectively), which confirms another point of our idea introduced in section 1 that the most information is accumulatively transferred by only few preceding elements of an early predictable sequence, which is exactly the intrinsic ground of the feasibility of the early prediction for an ongoing sequence.

In summary, the results shown in Figure 4 verify the hypothesis presented in section 3 and convince us that the intrinsic and theoretically sound ground of the capability of early prediction for an ongoing sequence is exactly the one that the kinetic regularity of its NATI can be approximated by an inverse heavy-tail distribution (formula (2)) so that the most information of it can be transferred in its early stage.

7 Conclusions

To reveal the intrinsic ground of early prediction, we first propose a new concept of ATI and its normalized form NATI to characterize the information movement in a sequence. Then we investigate the kinetic regularity of NATI of early predictable sequences on real data sets, and discover the fact that the kinetic regularity of NATI of early predictable sequences can be approximately modeled by an inverse heavy-tail distribution, which indicates that the most information is accumulatively transferred, and consequently the most uncertainty is gradually eliminated, by only few of preceding elements of an early predictable sequence.

Based on the kinetic model of NATI, we propose a heuristic algorithm, LearnK, to learn the parameter $k_{optimal}$, which is the minimum number of the preceding elements that must be observed before an ongoing sequence can be accurately classified. At last, the results of the experiments performed on real data sets verify that the kinetic regularity of ATI of sequences can be modeled by inverse heavy-tail distribution is exactly the underlying reason why early prediction for an ongoing sequence is feasible, and the early prediction based on the theory and algorithm proposed by this paper is reasonable and effectiveness.

References

1. Dongand, G., Pei, J.: Sequence Data Mining. Springer, Heidelberg (2007)
2. Lesh, M.O.N., Zaki, M.J.: Mining features for sequence classification. In: Proc. of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 342–346. ACM, New York (1999)
3. Srikant, R.R.: Mining sequential patterns: Generalizations and performance improvements. In: Apers, P.M.G., Bouzeghoub, M., Gardarin, G. (eds.) EDBT 1996. LNCS, vol. 1057, pp. 3–17. Springer, Heidelberg (1996)
4. Pei, J., Han, J.: Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In: Proc. of the 17th International Conference on Data Engineering, pp. 215–226. IEEE, Los Alamitos (2001)

5. Ayres, T.J., Flannick, J.: Sequential pattern mining using a bitmap representation. In: Proc. of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 429–435. ACM, New York (2002)
6. Zaki, M.J.: Spade: An efficient algorithm for mining frequent sequences. *Machine Learning* 42(1), 31–60 (2001)
7. Cheng, J.H., Yan, X.: Seqindex: Indexing sequences by sequential pattern analysis. In: Proc. of 2005 SIAM International Conference on Data Mining, pp. 601–605. SIAM, Philadelphia (2005)
8. Parker, P.C., Fern, A.: Gradient boosting for sequence alignment. In: Proc. of the 21st National Conference on Artificial Intelligence, pp. 452–457 (2006)
9. Karwath, N.A.: Boosting relational sequence alignments. In: Proc. of the 8th IEEE International Conference on Data Mining, pp. 857–862. IEEE, Los Alamitos (2008)
10. Tseng, M.: Cbs: A new classification method by using sequential patterns. In: Proc. of 2005 SIAM International Conference on Data Mining, pp. 596–600. SIAM, Philadelphia (2005)
11. Exarchos, T.P., Tspouras, M.G.: A two-stage methodology for sequence classification based on sequential pattern mining and optimization. *Data and Knowledge Engineering* 66(3), 467–487 (2008)
12. Wu, C., Berry, M.: Neural networks for full-scale protein sequence classification: Sequence encoding with singular value decomposition. *Machine Learning* 21(1), 177–193 (1995)
13. Ma, Q., Wang, J.T.L.: Dna sequence classification via an expectation maximization algorithm and neural networks: a case study. *IEEE Transactions on Systems, Man and Cybernetics* 31(4), 468–475 (2001)
14. Park, M.K.J.: Prediction of protein subcellular locations by support vector machines using compositions of amino acids and amino acid pairs. *Bioinformatics* 19(13), 1656–1663 (2003)
15. She, R., Chen, F.: Frequent-subsequence-based prediction of outer membrane proteins. In: Proc. of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 436–445. ACM, New York (2003)
16. Sonnenburg, S., Rätsch, G., Schäfer, C.: Learning interpretable SVMs for biological sequence classification. In: Miyano, S., Mesirov, J., Kasif, S., Istrail, S., Pevzner, P.A., Waterman, M. (eds.) RECOMB 2005. LNCS (LNBI), vol. 3500, pp. 389–407. Springer, Heidelberg (2005)
17. Alonso, C.J., Rodriguez, J.J.: Boosting interval based literals: Variable length and early classification. In: Data Mining in Time Series Databases. World Scientific, Singapore (2004)
18. Xing, Z., Pei, J.: Mining sequence classifiers for early prediction. In: Proc. of 2008 SIAM International Conference on Data Mining, pp. 644–655. SIAM, Philadelphia (2008)
19. Cover, T.: Elements of Information Theory, 2nd edn. John Wiley, Chichester (2006)
20. <http://www.cs.unm.edu/~immsec/data/>
21. <http://www.ll.mit.edu/cst.html>
22. Cohen, W.W., Singer, Y.: A simple, Fast, and Effective Learner. In: Proc. of the 16th National Conference on Artificial Intelligence, pp. 335–342 (1999)
23. Asuncion, A., Newman, D.J.: UCI machine learning repository (2007)

Mining Event Temporal Boundaries from News Corpora through Evolution Phase Discovery*

Liang Kong¹, Rui Yan², Han Jiang², Yan Zhang^{1,**},
Yan Gao³, and Li Fu³

¹ Department of Machine Intelligence, Peking University, Beijing 100871, China
Key Laboratory on Machine Perception, Ministry of Education, Beijing 100871, China

² Department of Computer Science, Peking University, Beijing 100871, China

³ Service Software Chongqing Institute of ZTE Corporation, China
{kongliang,r.yan,billybob}@pku.edu.cn, zhy@cis.pku.edu.cn,
{gao.yan6,fu.li3}@zte.com.cn

Abstract. Currently news flood spreads throughout the web. The techniques of Event Detection and Tracking makes it feasible to gather and structure text information into events which are constructed online automatically and updated temporally. Users are usually eager to browse the whole event evolution. With the huge quantity of documents, it is almost impossible for users to read all of them. In this paper, we formally define the problem of event evolution phases discovery. We introduce a novel and principled model (called EPD), aiming at temporally outlining the entire news development. A news document is usually not atomic but consists of independent news segments related to the same event. Therefore we first employ a latent ingredients extraction method to extract event snippets. Unlike traditional clustering methods, we propose a novel metrics integrating content feature, temporal feature, distribution feature and bursty feature to measure the correlation between snippets along timeline in a specific event. Combined with bursty feature, we introduce a novel method to compute word weight. We employ HAC to group the news snippets into diversified phases. An optimization problem are utilized to decide the number of phases, which makes EPD applied. With our novel evaluation method, empirical experiments on two real datasets show that EPD is effective and outperforms various related algorithms. Automatic event chronicle generated is introduced as a typical application of EPD.

Keywords: Evolution Phases Discovery.

1 Introduction

Currently document flood which comes from different websites spreads throughout the web and users can be easily trapped in news sea. In some news websites,

* Supported by NSFC with Grant No. 61073081, National Key Technology R&D Pillar Program in the 11th Five-year Plan of China with Research No. 2009BAH47B05, and ZTE University Partnership Fund.

** Corresponding author.

the mess-up information could be gathered together, turned into event, issue, topic or special manually by editors, and displayed to users, for instance, *Yahoo! News Topics* and *CNN Special Coverage & Hot Topics*. With the development of the techniques of Topic Detection and Tracking (TDT) [1], some web service can gather news from many websites and structure it into news topics which are constructed online automatically and updated temporally, such as *GoogleNews*.

The techniques of event detection and tracking makes it feasible for a user to know “what’s happening” and “what’s new”. However, there are always an event evolution over time, which can be summarized by several main phases. With the huge quantity of documents constructed and updated at all times, it is almost impossible for users to read all of them, which hinders users from learning from the the ins and outs of the event evolution. For example, *While a user query “H1N1”, he will receive a large number of documents from the search engine then he expects to find out the development history of the event*. This kind of scenario leads to a new question: **Can we automatically discover evolution for a specific event and generate the chronicle for users?** We believe users will be better satisfied if we can provide a humanized system which enables them to easily explore and exploit the event evolution. Hence, event evolution phases discovery is an applied and valuable research subject.

In this paper, we formally define the problem of event evolution phases discovery. Intuitively, the task can be regarded as a problem of clustering. However, there are some remarkable characteristics when the news documents are divided into diversified clusters along the timeline. Therefore the traditional clustering methods have poor performance. Motivated by above mentioned, we introduce a novel and principled model (called EPD) for discovering evolution phases for a specific event that consists of a quantity of documents information. EPD is composed of two major phases and effective to accomplish the task of evolution phases discovery.

Since EPD can be regarded as a clustering problem, there are three important questions to answer. The first question is what to be clustered. A document usually contains more than one latent ingredient. The similarity between documents can be influenced by some noisy latent ingredients, such as background information. Therefore, documents are unsuitable atomic units to be clustered. The second question is how to measure the correlation between texts. An evolutionary phase is a chunk for event evolution along timeline. The difference between phases can be caused by content feature, temporal feature, distribution feature, bursty feature and others. Thus, an fusion model for several important and useful features should be considered for improving similarity metrics. The third question is how many clusters to be decided. An applicable system requires a method to automatic decide the number of phases. In this paper, we will give the reasonable answers to those questions.

Indeed, our model can be combined with any existing event detection algorithms. We show that EPD is motivated by and well reflects the existing observations and findings. Empirical experiments on two real datasets show that EPD is effective and outperforms various related algorithms.

The rest of this paper is organized as follows. In Section 2, we revisit the related work. Section 3 gives the problem formulation. Section 4 introduces the model for event evolution phases discovery. Experiments in Section 5 show the performance of EPD. Section 6 summarizes contributions of this paper.

2 Related Work

Topic detection and tracking (TDT) research has been extensively studied in previous work. Pilot experiments in retrospectively and incrementally clustering of text documents have been done as a part of event detection task initiative [2] and query document like retrieval [3]. Others follow their work on event detection [4], improving clustering techniques [5] and novelty detection [6]. Our work is related with those work. Finding a new event requires technology of event detection. Event tracking helps us to access the documents stream. Nevertheless, our work is distinct from these work before. We focus on discovering diversified evolution phases rather than whether events are found.

Topic tracking also stimulates many researches because events are dynamic and evolutionary character is studied in [7]. Tracking captured dependencies in news topics, either casual or temporal in [8] and threads topics. This work was enriched by [9] into “incident threading”, aiming at expanding dependency variety. Probability generative model for topic mining came into use after Probabilistic Latent Semantic Analysis (PLSA) was proposed. In [10] Temporal Text Mining (TTM) was introduced to discover the latent theme from texts and construct an evolution graph. In [11] newswire evolution was constructed. These works focused on topic threads evolved from one to another. Based on previous work, we introduce EPD to divide the event evolution line into diversified phases. Readers can obtain a temporal outline of event evolution from a macro point of view, which cannot be solved from local views by event detection or evolution.

To the best of our knowledge, our approach has different characteristics from all previous work. In this paper, we formally define the problem of event evolution phases discovery and present a novel model EPD to solve the problem.

3 Problem Formulation

In this section, we formally define the related concepts and the task of evolution phases discovery. Let us begin with defining a few key concepts as follows.

Document. Let $D^\varepsilon = \{ \langle d_1^\varepsilon, t_1 \rangle, \langle d_2^\varepsilon, t_2 \rangle, \dots, \langle d_n^\varepsilon, t_n \rangle \}$ be the set of documents published about a specific event ε , where d_i^ε is the text document and t_i is the timestamp. d_i^ε is represented by a bag of words from a fixed vocabulary $W^\varepsilon = \{w_1^\varepsilon, w_2^\varepsilon, \dots, w_m^\varepsilon\}$. That is, $d_i^\varepsilon = \{c(d_i^\varepsilon, w_1^\varepsilon), c(d_i^\varepsilon, w_2^\varepsilon), \dots, c(d_i^\varepsilon, w_m^\varepsilon)\}$, where $c(d_i^\varepsilon, w^\varepsilon)$ denotes the number of occurrences of word w^ε in d_i^ε .

Snippet. Let $S^\varepsilon = \{ \langle s_1^\varepsilon, ts_1 \rangle, \langle s_2^\varepsilon, ts_2 \rangle, \dots, \langle s_k^\varepsilon, ts_k \rangle \}$ be the set of snippets extracted from documents set D^ε , where s_i^ε is the text of snippet i and ts_i is its timestamp.

Phase. Let $P^\varepsilon = \{ \langle p_1^\varepsilon, Ts_1, Te_1 \rangle, \langle p_2^\varepsilon, Ts_2, Te_2 \rangle, \dots, \langle p_l^\varepsilon, Ts_l, Te_l \rangle \}$ be the set of phases describing the evolution of event ε , where p_i^ε is a set of snippets in phases i , Ts_i is start time and Te_i is the end time. Given snippets set S^ε , $\forall s_i^\varepsilon, s_j^\varepsilon \in p_m^\varepsilon, \forall s_k^\varepsilon \in p_n^\varepsilon$, we have $\|coh(s_i^\varepsilon, s_j^\varepsilon)\| > \|coh(s_i^\varepsilon, s_k^\varepsilon)\|$ and $\|coh(s_i^\varepsilon, s_j^\varepsilon)\| > \|coh(s_j^\varepsilon, s_k^\varepsilon)\|$, where $\|coh(\cdot, \cdot)\|$ is the cohesiveness between any two snippets. The definition indicates two snippets within the same phase are more cohesive than those from different phases.

With the definitions of key concepts, we can now formally define the major task of evolution phases discovery. Given the input of a set of documents D^ε , the task is:

Evolution Phases Discovery. Formally, we want to infer the set of versions P^ε about the event ε . An event may consist of diversified phases. By inferring the set of phases, we expect to keep track of evolution about an event, give users multilateral understanding of an event and classify documents to the evolution phases of an event, etc.

4 Evolution Phases Discovery

4.1 System Framework for EPD

Intuitively, the task can be regarded as a problem of clustering. However, in a specific event, there is some remarkable characteristics when the news documents are divided into diversified clusters along the timeline. Therefore we cannot directly apply classic clustering methods. For solving the problem, we propose EPD. As shown in Fig. 1, the system framework of EPD is composed of two major steps. It works as follows:

1) Snippet Extraction. Since a document usually contains more than one latent ingredients, documents cannot be considered as atomic units for phases discovery. We employ an event snippets exacting algorithm to solve this problem.

2) Features Selection and Phases Classification. Based the snippets set obtained in step 1), we start to select and extract useful features. We discuss four features and propose a fusion model to compute the similarity between two snippets. Taking bursty feature into consideration, we propose the modified word weight. Afterwards, we employ the HAC clustering framework to cluster the event snippets to diversified phases. Furthermore, we suggest a feedback model to implement optimization problem to decide the number of phases.

In the following subsections, steps 1) and 2) are formally expressed.

4.2 Snippet Extraction

A document usually contains many latent segments. [12] firstly defines the concept of event snippet which is the different Latent Ingredients(LIs) in one document. They present the novel event snippet recognition models based on LIs extraction and exploit a set of useful features consisting of context similarity,

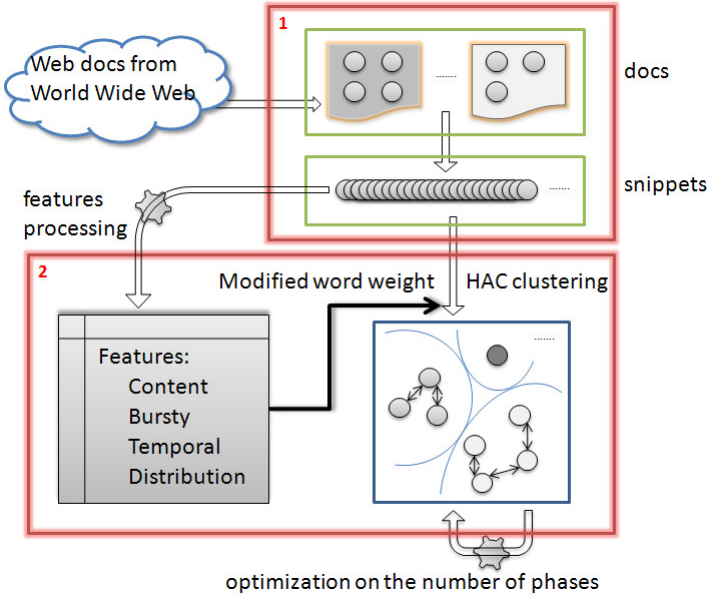


Fig. 1. The system framework for the model of EPD

distance restriction, entity influence from thesaurus and temporal proximity. In our model, we employ their algorithm to extract snippets for documents and we can obtain the set of snippets S^ϵ .

4.3 Features Selection

EPD is intrinsically a clustering problem. A phase is a chunk for event evolution, a cluster of snippets belong to the same evolution phase. When we classify snippets into different phases, we need to select suitable metrics to measure the similarity between them. In our model, we mainly utilize multiple event features such as content, temporal, and other information. In following part, we will describe four useful features.

Temporal Feature. Temporal information is an important dimension to decide snippet clusters, contemporary snippets tend to discuss closely related events and as a result, are likely to be at the same evolutionary stage. We denote $sim_t(s_i^\epsilon, s_j^\epsilon) = e^{-|ts_i - ts_j|}$, where $|ts_i - ts_j|$ is the temporal gap between two snippets. The farther two snippets are, the less similar they will be.

Distribution Feature. The distribution of snippets indicates related events. If there are too many snippets between two snippets (sorted by date), they are less likely to be connected. n_{ij} is the number of snippets between the two snippets and n_{ij} is 0 when $ts_i = ts_j$. The distribution similarity is defined as $sim_d(s_i^\epsilon, s_j^\epsilon) = e^{-n_{ij}}$.

Bursty Feature. A document burst reflects a concentrated topic discussion and indicates new updates occur here. A topic is usually discussed heavily within the center of an episode and in the transition period the topic temporarily fades and stays away from people’s attention. Suppose there are n_i snippets at the day of snippet s_i^ε and n_j for s_j^ε . $sim_b(s_i^\varepsilon, s_j^\varepsilon) = e^{-|n_i - n_j|}$.

Content Feature. Content information is basic dimension to decide snippet clusters. The content cohesion between two snippets is measured by the cosine similarity $sim(s_i^\varepsilon, s_j^\varepsilon)$. Every snippet is a bag-of-word (in raw words). Word weight is measured by traditional TFIDF. we have,

$$sim(s_i^\varepsilon, s_j^\varepsilon) = \frac{\sum_{w \in s_i^\varepsilon \cap s_j^\varepsilon} (weight_{tf.idf}(w_i, s_i^\varepsilon) \times weight_{tf.idf}(w_j, s_j^\varepsilon))}{\sqrt{\sum_{w_i \in s_i^\varepsilon} (weight_{tf.idf}(w_i, s_i^\varepsilon))^2} \times \sqrt{\sum_{w_j \in s_j^\varepsilon} (weight_{tf.idf}(w_j, s_j^\varepsilon))^2}} \tag{1}$$

$weight_{tf.idf}(w, s)$ indicate the TFIDF weight of word w in snippet s . In Section 4.4, by considering word bursty feature, we will propose a method to improve TFIDF for computing the word weight.

4.4 Phases Cluster

Modified Word Weight. When we consider both content feature and bursty feature, We can not ignore their interaction. The words in bursty time is usually more important. Thus, the bursty feature is modeled in term weighting. We propose the bursty factor \tilde{p}_t . The time t of a larger \tilde{p}_t is a burst time. A snippet in such burst time is more informative and words in it should gain larger weight.

Inspired by [15], we propose a method to decide the bursty factor \tilde{p} . According to [15], document volume is a function of time elapse for a specific. A topic thread reaches its peak volume at peak time t_p . There are two distinct behaviors: the volume within a t_p -centered time window can be well modeled by e^{-bx} , while outside of the window, the volume is best modeled by $a|\log(|x|)|$. Within the window the volume builds up and decays quickly while the outside part is a long tail. a and b are up to curve fitting. To integrate such unique character, we create a time windows Win_t . In a bursty time, the number of snippets within Win_t is more than around the Win_t . $N_{within(Win_t)}$ represents the number of snippets within windows Tw and $N_{around(Win_t)}$ represents the number of snippets around windows Win_t . Respectively, we compute the number of snippets in front and back of the time windows Win_t , and let $N_{around(Win_t)}$ equal the sum of them, where the size is the half of Win_t . Therefore, we have

$$\tilde{p}_t = \frac{N_{within(Win_t)}}{N_{around(Win_t)}} \tag{2}$$

Thus, the modified word weight is as follows.

$$weight_{modified}(w, s) = \frac{weight_{tf.idf}(w, s) \cdot \tilde{p}_t}{\sqrt{\sum_{w' \in s} (weight_{tf.idf}(w', s) \cdot \tilde{p}_t)^2}} \quad (3)$$

where $weight_{modified}(w, s)$ indicate the weight of word w in snippet s .

Fusion of Features. There are several drawbacks for utilizing merely one factor, which is incomplete to reflect the event evolution. We need connect snippets which are cohesive enough. Context similarity, time decay, snippet distribution and bursty all affect the cohesion measurement in balance and we combine them into compound similarity:

$$sim_c(s_i^\varepsilon, s_j^\varepsilon) = sim(s_i^\varepsilon, s_j^\varepsilon) \times sim_t(s_i^\varepsilon, s_j^\varepsilon) \times sim_d(s_i^\varepsilon, s_j^\varepsilon) \times sim_b(s_i^\varepsilon, s_j^\varepsilon) \quad (4)$$

Clustering. Based on the fusion metrics of similarity between two snippets, we traverse all snippets and group them by Hierarchical Agglomerate Clustering (*HAC*) until the number of phases c is reached. Every cluster is a set of snippets, and we sort their timestamp to get Ts_i and Te_i for phase i .

4.5 Decision of the Number of Phases

An actual system requires to automatic decide the number of phases. In processing course of HAC, for every step, we can get a partition U of phases: $U(S) = \{u_1 \cdots \cup u_k \cup \cdots \cup u_c\}$ and u_k represents the k -th temporary phase of snippet collection S . We optimize the best value of U :

$$V(U) = \frac{\max_{1 \leq i \leq c} \{ \max_{1 \leq j \leq c, j \neq i} \{ \Delta(u_i, u_j) \} \}}{\min_{1 \leq k \leq c} \{ \delta(u_k) \}} \quad (5)$$

$\Delta(u_i, u_j)$ defines the cohesion between u_i and u_j , which is measured by average pairwise similarity of two temporary phases and $\delta(u_k)$ denotes all possible similarities within an temporary phase u_k . The main goal of this measure is to minimize inter-phase similarity whilst maximizing intra-phase similarity. Thus small values of V correspond to good clusters and the number of clusters c that minimizes $V(U)$ is the optimal c^* to choose.

$$c^* = \arg \min_c V(U) \quad (6)$$

Based on the optimization, through continual feedback in HAC, the model can automatically decide the number of phases.

5 Experiments

In this section, we describe the experimental designs and evaluation of event evolution phases discovery. Since there is no existing standard test set for EPD,

we opt to construct a test set. We show the effectiveness of our model EPD with experiments both on a smaller dataset *H1N1 Pandemic* (including 2318 documents, call H1N1) and on a larger dataset *Dell's accounting problems* (including 4772 documents, called DeA). H1N1 is the 2009 flu pandemic. It began in Mexico and continued to spread globally, which kept a long time evolution. DeA is one of the largest financial scandal events from 2006 to 2008. The statistics of the datasets is shown in Table 1. We crawl documents from the mainstream news web portals, such as *BBC*, *Reuters*, *MSNBC*, *NYTimes*, *People*, to build the dataset. Since there is no existing standard test set for diversified versions discovery, we opt to construct our own test sets. We first present the evaluation for the performance of snippets clustering, which is the most important part in phases discovery. Then we illustrate and analyze the EPD results. Finally, we introduce an EPD application.

Table 1. Statistics of Datasets

Dataset	Quantity			Time Span		
	Docs	Snips	Variation	Docs	Snips	Variation
H1N1	2318	4052	+29.96%	03/13/2009-11/30/2009	03/08/2009-11/30/2009	+1.90%
DeA	4772	5290	+10.85%	08/17/2006-08/30/2008	07/01/2006-08/30/2008	+6.31%

From Table 1 we can see the number of texts is enlarged by Snippets extraction. Quantity is an important indicator which facilitates to distinguish an evolution phase. Enlarged datasets benefit EPD without extra human labor to read. Snippets extraction also helps extend the time span for a more complete topic review. For smaller topics such as H1N1, snippets greatly enlarges unit quantity but a smaller time span extension while for larger topics such as DeA, it is quite the opposite. Either phenomenon is helpful.

5.1 Performance of Snippets Phases Clustering

As mentioned above, an important part of EPD is to classify the snippets into diversified phases. Therefore, we firstly evaluate the snippets phases clustering.

Evaluation. With regard to an event, it is almost impossible to manually observe all the snippets and classify them into different phases. Hence, we evaluate the effectiveness of EPD in a pairwise judge task.

In the pairwise judge task, we focus on judging whether a pair of snippets belongs to the same phases. Firstly we construct the pairwise standard test sets. We randomly sample 200 pairs of snippets for H1N1 and 300 pairs of snippets for DeA. We make sure each snippet is different, which enhances the evaluation reliability. Afterwards, each pair of snippets is shown to volunteers and whether they belongs to the same phase depends on the voting result. If a result is hard to judge, the pair will be discarded and we will add a new pair to test set, which guarantees effectiveness of the test set. Finally, we can obtain the pairwise test

set, denoted as $T_\varepsilon = \{ \langle \langle s_{i_1}^\varepsilon, s_{i_2}^\varepsilon \rangle, v_i \mid v_i \in \{0, 1\}, s_{i_1}^\varepsilon \in S^\varepsilon, s_{i_2}^\varepsilon \in S^\varepsilon \}$, where $v_i = 1$ indicates that $s_{i_1}^\varepsilon$ and $s_{i_2}^\varepsilon$ belong to the same phase and $v_i = 0$ indicates otherwise situation. Specifically, we construct the pairwise test sets of H1N1 and DeA, which are abbreviated to T_{H1N1} and T_{DeA} respectively.

In our evaluation, we use the *Precision* criterion in the pairwise test task. Given the pairwise documents $\langle s_{i_1}^\varepsilon, s_{i_2}^\varepsilon \rangle$, each method can give a judge v'_i . Hence, we can define *Precision* in the pairwise test task as P_{score} , that is:

$$P_{score} = \frac{\sum_{\langle d_{i_1}, d_{i_2} \rangle} v_i \odot v'_i}{|T_\varepsilon|} \tag{7}$$

where $|T_\varepsilon|$ represents the size of the pairwise test set for event ε . \odot is XNOR.

Performance and Discussion. In this part, we discuss quantitatively the effectiveness of the snippets cluster. Comparisons against related algorithms are also conducted. The related algorithms studied include:

- **K-means**, which clusters the snippets based on the text similarity.
- **LDA**, which clusters the documents based on the word distribution.
- **EPD-c**, which implements a special version of EPD, but only the using content feature.
- **EPD-t**, which implements a special version of EPD, but only the using temporal feature.
- **EPD-d**, which implements a special version of EPD, but only using the distribution feature.
- **EPD-b**, which implements a special version of EPD, but only using the bursty feature.
- **EPD-m**, which implements a special version of EPD, without using modified TFIDF weight.

We compare the performance of EPD with related algorithms in the two built pairwise test sets, T_{H1N1} and T_{DeA} . The P_{score} performance of the pairwise judge task in T_{H1N1} and T_{DeA} is demonstrated in Fig. 2. From Fig. 2 we can see that K-means, LDA and EPD-c outperform the EPD-t, EPD-d and EPD-b.

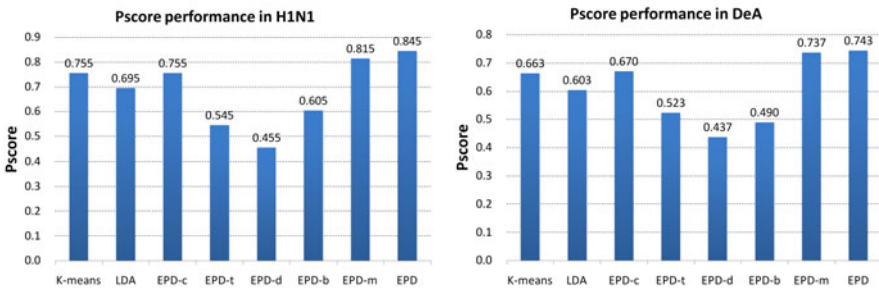


Fig. 2. P_{score} performance comparison in the pairwise test task of H1N1 and DeA

The results show that clustering only using the temporal, distribution or bursty feature is not satisfactory. EPD-m performs better than the K-means, LDA and EPD-c. Therefore, it illustrates two points. First point is that classic K-means and LDA are not satisfactory and since EPD-c only using content feature which is similar to K-means, it also doesn't perform well. Second point is that other three features are auxiliary for EPD clustering in that none of them is capable to achieve a satisfactory performance alone but they do help when combined. All features are useful to help understand event as shown above, EPD acquires the best performance, which also proves that the modified word weight is effective.

5.2 Performance of Event Phases Discovery

The time span is an important performance for a phase. In this section, we focus on evaluating the performance of EPD on time span.

Evaluation. Inspired by *Precision* and *Recall*, we propose *temporal Precision* (P) and *temporal Recall* (R) for test of time span. Given standard time span T_{gold} and tested time span T , we have,

$$P = \frac{T \cap T_{gold}}{T}, R = \frac{T \cap T_{gold}}{T_{gold}} \quad (8)$$

Then we can compute macro or micro F -measure by traditional methods.

Table 2. Results of Approaches on all Datasets. F⁺: macro-average; F^{*}: micro-average; P': average temporal Precision; R': average temporal Recall.

Data Set		K-means	EPD-c	EPD-ns	EPD	Standard
H1N1 2009	Phase1	04/28-05/07	04/23-05/07	04/29-06/03	04/27-05/23	04/28-05/15
	Phase2	06/11-07/29	04/29-08/13	06/09-09/04	06/01-08/03	05/01-07/31
	Phase3	07/27-09/14	07/08-10/01	07/15-10/11	07/09-09/27	07/05-09/30
	Phase4	08/30-11/06	10/17-11/26	10/15-11/30	10/03-11/30	10/13-11/30
	(P', R')	(84%, 54%)	(88%, 84%)	(74%, 91%)	(86%, 90%)	(100%, 100%)
	(F ⁺ , F [*])	(66%, 64%)	(86%, 85%)	(82%, 80%)	(88%, 86%)	(100%, 100%)
DeA	Phase1	08/06/2006- 12/19/2006	08/11/2006- 12/15/2006	09/10/2006- 09/22/2006	08/08/2006- 10/09/2006	08/01/2006- 10/01/2006
	Phase2	11/21/2006- 06/02/2007	11/16/2006- 08/12/2007	01/12/2007- 09/12/2007	11/01/2006- 07/01/2007	11/01/2006- 08/15/2007
	Phase3	03/30/2008- 07/03/2008	04/14/2008- 08/02/2008	05/19/2008- 06/30/2008	05/04/2008- 07/29/2008	05/19/2008- 08/29/2008
	(P', R')	(63%, 68%)	(68%, 84%)	(96% , 46%)	(90%, 81%)	(100%, 100%)
	(F ⁺ , F [*])	(66%, 61%)	(75%, 73%)	(62%, 58%)	(85%, 85%)	(100%, 100%)

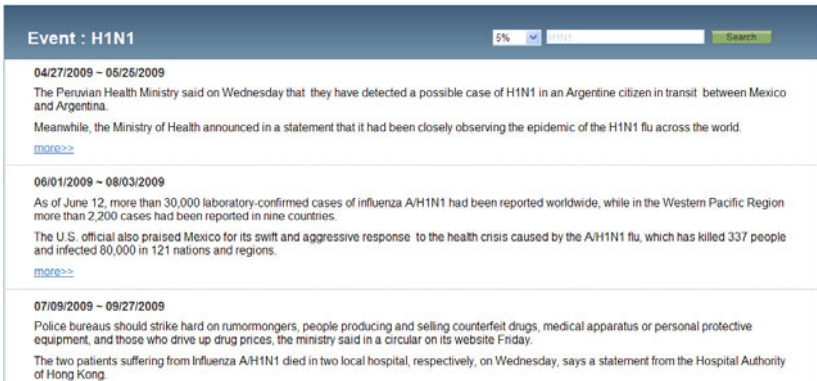
We invite five volunteers who major in Journalism and Communication to create the gold standard. Information which helps volunteers consists of three aspects: a) the amount of reports in a day; b) the format and length of reports; c) the report contents. They read and analyze answers especially at the boundaries. Once they have disputes, kappa statistics is used to measure agreements.

Performance and Discussion. We conduct comparisons against three related algorithms, K-means, EPD-c and EPD-ns. EPD-ns implements a special version of EPD, without extracting snippets. Results are listed in Table 2. Taking H1N1 as an example, we describe the mean of standard. During 04/28 to 05/15, media focused on the first wave of infection and warned a global flu. The first report on H1N1 protection came on 05/01 and hence the second phase began. From 05/01 to 07/31, reports focused on coping strategies against influenza A and H1N1 broke out throughout the world. H1N1 became quiet from 07/05 to 09/30 with few retrospective reviews. On 10/13 it returned and the second wave prevailed.

From Table 2, EPD has a better performance than EPD-ns. It shows that snippet is more effective atomic unit for phase discovery. Moreover, all perform better on smaller dataset due to the consecutiveness of data. Large dataset tends to be more discrete over time, which is an obstacle for phase discovery.

5.3 An Application Demonstration

Tangible benefit can be realized when applying our model to a demo system. An typical application is automatic produce event evolution phases chronicle. We combine EPD with a graph-based multi-document summarization algorithm (GMS) proposed by Wan et al [14]. We apply GMS on the snippet set of each phase. A snapshot from our demo system is present in Fig. 3.



The screenshot shows a web application interface for H1N1 event evolution phases. At the top, there is a header "Event : H1N1" and a search bar with "5%" and a "Search" button. Below the header, there are three event phases listed with their dates and descriptions. Each phase has a "more>>" link.

Event : H1N1
<p>04/27/2009 – 05/26/2009</p> <p>The Peruvian Health Ministry said on Wednesday that they have detected a possible case of H1N1 in an Argentine citizen in transit between Mexico and Argentina.</p> <p>Meanwhile, the Ministry of Health announced in a statement that it had been closely observing the epidemic of the H1N1 flu across the world.</p> <p>more>></p>
<p>06/01/2009 – 08/03/2009</p> <p>As of June 12, more than 30,000 laboratory-confirmed cases of influenza A/H1N1 had been reported worldwide, while in the Western Pacific Region more than 2,200 cases had been reported in nine countries.</p> <p>The U.S. official also praised Mexico for its swift and aggressive response to the health crisis caused by the A/H1N1 flu, which has killed 337 people and infected 80,000 in 121 nations and regions.</p> <p>more>></p>
<p>07/09/2009 – 09/27/2009</p> <p>Police bureaus should strike hard on rumormongers, people producing and selling counterfeit drugs, medical apparatus or personal protective equipment, and those who drive up drug prices, the ministry said in a circular on its website Friday.</p> <p>The two patients suffering from Influenza A/H1N1 died in two local hospital, respectively, on Wednesday, says a statement from the Hospital Authority of Hong Kong.</p>

Fig. 3. A snapshot of an application demonstration

6 Conclusion

In this paper, we present an event evolution phases discovery model, which helps in quickly mining the event development. Within our innovative model, we take two steps to achieve this task. First step is snippets extracting. We focus on second step, in which we use HAC to cluster these snippets. Based on the characteristic of event evolution, we propose four useful features for computing the

similarity. Since the bursty existing in event evolution, we propose a modified word weight. An automatic decision of the number of phases proposed in this paper makes it feasible to application. Empirical experiments on two real datasets show that EPD is effective and outperforms various related algorithms.

While the work in this paper exhibits good performance, it can be further improved. For simplicity, we do not import parameters in fusion model, which will be discussed in our future work. Besides, in the future, combined with event diversified version discovery techniques, we can detect the multi-aspects evolution for a specific event, which is very urgent and important in practical applications.

References

1. Allan, J., Carbonell, J., Doddington, G., Yamron, J., Yang, Y.: Topic detection and tracking pilot study: Final report, Tech. Rep. (1998)
2. Yang, Y., Pierce, T., Carbonell, J.: A study of retrospective and on-line event detection. In: SIGIR 1998, pp. 28–36 (1998)
3. Allan, J., Papka, R., Lavrenko, V.: On-line new event detection and tracking. In: SIGIR 1998, pp. 37–45 (1998)
4. Brants, T., Chen, F., Farahat, A.: A System for new event detection. In: SIGIR 2003, pp. 330–337 (2003)
5. Franz, M., Ward, T., McCarley, J.S., Zhu, W.: Unsupervised and supervised clustering for topic tracking. In: SIGIR 2001, pp. 310–317 (2001)
6. Yang, Y., Zhang, J., Carbonell, J., Jin, C.: Topic-conditioned novelty detection. In: KDD 2002, pp. 688–693 (2002)
7. Liu, S., Merhav, Y., Yee, W.G., Goharian, N., Frieder, O.: A sentence level probabilistic model for evolutionary theme pattern mining from news corpora. In: SAC 2009, pp. 1742–1747 (2009)
8. Nallapati, R., Feng, A., Peng, F., Allan, J.: Event threading within news topics. In: CIKM 2004, pp. 446–453 (2004)
9. Feng, A., Allan, J.: Finding and linking incidents in news. In: CIKM 2007, pp. 821–830 (2007)
10. Mei, Q., Zhai, C.: Discovering evolutionary theme patterns from text: an exploration of temporal text mining. In: SIGKDD 2005, pp. 198–207 (2005)
11. Yang, C., Shi, X.: Discovering event evolution graphs from newswires. In: WWW 2006, pp. 945–946 (2006)
12. Yan, R., Li, Y., Zhang, Y., Li, X.M.: Event Recognition from News Webpages through Latent Ingredients Extraction. In: Cheng, P.-J., Kan, M.-Y., Lam, W., Nakov, P. (eds.) AIRS 2010. LNCS, vol. 6458, pp. 490–501. Springer, Heidelberg (2010)
13. Wei, C.-P., Lee, Y.-H., Chiang, Y.-S., Chen, J.-D., Yang, C.C.C.: Discovering event episodes from news corpora: a temporal-based approach. In: ICEC 2009, pp. 72–80 (2009)
14. Wan, X., Yang, J.: Multi-document summarization using cluster-based link analysis. In: SIGIR 2008, pp. 299–306 (2008)
15. Leskovec, J., Backstrom, L., Kleinberg, J.: Meme-tracking and the dynamics of the news cycle. In: SIGKDD 2009, pp. 497–506 (2009)

Event Detection over Live and Archived Streams

Shanglian Peng¹, Zhanhuai Li¹, Qiang Li², Qun Chen¹,
Wei Pan¹, Hailong Liu¹, and Yanming Nie¹

¹ School of Computer Science and Technology, Northwestern Polytechnical University

² College of Software and Microelectronics, Northwestern Polytechnical University,
Xi'an 710072, China

pengshanglian@mail.nwpu.edu.cn, powerqy@163.com,
{lizhh, chenbenben, panwei1002, liuhailong}@nwpu.edu.cn,
nymcoon@gmail.com

Abstract. It is becoming increasingly crucial to integrate pattern matching functionality over live and archived event streams with hybrid event queries for various complex event processing(CEP) applications. As existing Stream Processing Engine(SPE) and DBMS alone can not accommodate hybrid event queries, we investigate system integration issues and scheduling algorithm optimizations of hybrid event processing based on *discrete pattern* in a single framework. First, hierarchical event stream storage is introduced to synchronize data access over live/archived event streams. Second, live and historical partial pattern matching caching mechanisms are proposed to provide effective partial pattern match search and update. Third, to reduce database access overhead, sub-window based event detect scheduling algorithms are proposed. Empirical performance study in a prototype hybrid event processing engine demonstrates effectiveness of our approaches.

Keywords: complex event processing, NFA, data stream, RFID.

1 Introduction

With the availability of large volumes of data generated by monitoring applications such as RFID-enabled asset tracking, financial trend prediction, network intrusion detection, and credit card billing, CEP over event streams has become increasingly crucial. Currently, CEP is mainly integrated into Stream Processing Engines(SPEs) that can provide efficient real-time information when events stream in. As SPEs focus on high-performance and low latency streaming data processing, they offer limited or no historical events access integration. SPEs such as Borealis [1], TelegraphCQ [2] and HiFi [3] have integrated certain form of historical data and contextual data access while processing new event data.

In fact, there are many applications where historical data access is important and useful for CEP as shown in the following cases.

Case 1 Trend Prediction. Financial Data Analysis: Increase, decrease and stability are three types of events for price of a particular stock. To well understand

behavior of a stock, user may subscribe event query such as “when a *fall pattern*(\setminus) is detected on live stream of a stock, look for *tick patterns*(\surd) on its historical stream” to predict whether the stock will bring profits in the near future [4]. In this case, both live and archived stock streams are accessed to fulfill the event detection task.

Case 2 Object Tracking. RFID-enabled object tracking. One example of RFID-enabled object tracking is driving time estimation in highway monitoring [5]. RFID tags are installed on cars for automatic toll collection and RFID readers are installed among different toll gates and checkpoints. When a car is observed on certain point of the monitoring system, its information is transmitted to the system and joined with its previous historical information to compute travel time between two sites.

Case 3 Anomaly Detection. Credit card fraud detection: Credit card transaction generates event streams which need to be archived. Incoming transaction event is then compared with historical transaction data to detect abnormal consumption pattern(e.g., a series of large purchases).

These cases illustrate the opportunity and necessity for integration of CEP over live and archived stream data, but the following challenges need to be addressed. *First*, application systems can easily generate gigabytes to terabytes of data every day, this sheer volume of data need to be properly maintained in memory and on disks to support effective live and historical event detection. *Second*, as event detection proceeds, partial matches over live and historical event streams become huge, hence effective partial matches caching is very important. *Third*, CEP queries initiate many instances on live stream which could incur frequent access and recomputation of archived event stream, hence effective scheduling strategies of event detection algorithms are needed to reduce archived data load overhead and achieve fairly good system response time. Moirae [6] and Dejavu [4][7] are two SPEs which have comprehensively addressed integration of live and archived stream processing. Moirae focuses on producing the most relevant approximate results. Although precise results may be generated incrementally, Moirae is not suitable for defining and processing hybrid CEP patterns. Dejavu integrates declarative pattern matching over live and archived events streams, but it mainly supports *contiguous patterns* such as tick pattern(\surd) in a stock stream. To our observation, *contiguous patterns* is a special case of *discrete pattern* whose sub-events could disperse in any possible positions in live/archived event streams. *Discrete pattern* is able to express more broader types of patterns and is not well addressed in Dejavu. In this paper, we focus on *discrete pattern match over live and archived event streams* in a single framework. Our works include: (1)To maintain consistency among instances of hybrid query on the stream data, we use a hierarchical storage architecture to synchronize data access over archived event streams. (2)Effective partial match results caching mechanism is proposed to provide fast event match retrieval and update. (3)Instance-based and time-window based scheduling strategies are proposed to facilitate different stream data characteristic and system workload. With proper

selection of scheduling method, DB access and re-computation of overlapping streams can be greatly reduced. (4) A HYbrid Event Processing Engine (HYPE) is implemented and empirical performance study demonstrates effectiveness of our approaches.

The rest of this paper is organized as follows. Section 2 introduces related work. Section 3 introduces preliminary of CEP and formalization of hybrid event detection problem. System architecture is described in Section 4. Section 5 presents experimental analysis in a prototype system of HYPE. Finally, conclusion is summarized in Section 6.

2 Related Work

CEP has been extensively studied in active database [8], SASE [9], Cayuga [10] and ZStream [11] are CEP engines that have their own pattern match languages but do not address live/historical event streams integration and hybrid event query. Latte [12] is a stream-processing system that addresses queries over live/historical streams in intelligent transportation management and information systems. Moirae [6] is a general purpose continuous monitoring engine which integrates event history into continuous monitoring. Moirae supports four types of queries: event queries, standard hybrid queries, context queries and contextual hybrid queries. In Moirae, detected event is complemented with historical information and similar events occurred in the past are retrieved during event detection making use of Information Retrieval methods. Due to large live and archived streams, results of queries are produced incrementally and approximately in Moirae by using techniques from many fields. Dejavu [4] is the most similar system with us in dealing with hybrid event queries. Dejavu extends the MySQL relational database engine with pattern match support proposed in [13]. However, Dejavu mainly focuses on contiguous patterns (such pattern in stock trading), discrete patterns are not well addressed. DataDepot [14] is a stream warehousing system that provides warehousing of streams but does not support hybrid event queries over live/historical streams.

Live and historical streams combination is also addressed by Chandrasekaran et al. [5] and Reiss et al. [15]. Chandrasekaran addresses overload in hybrid query processing and applies a reduction technique on archived data by random sampling or window aggregations. In [15], Bitmap Indices is examined to store and query historical data according to update characteristics and query requirements of streaming applications.

3 Preliminaries and Problem Formalization

In this section, we introduce basic definitions of events, event model and formalization of hybrid pattern match query.

3.1 Event Model

An event is defined as occurrence of an activity in a system. Basically, events can be classified into basic(or atomic) event and complex event.

Definition 1 (Basic Event). *A basic event is denoted as $E=E(EventID, TypeID, time, \langle a_1, \dots, a_n \rangle)$, where $EventID$ is identifier of the event; $TypeID$ is event type identifier of an event instance, $time$ is the time the event occurs and $\langle a_1, \dots, a_n \rangle$ are other attributes of an event. Basic event is indivisible and occurs at a point of time.*

Definition 2 (Complex Event). *Complex event is defined as $E=E(EventID, C=\langle e_1 Opr e_2 Opr \dots Opr e_n \rangle, t_s, t_e, \langle a_1, \dots, a_m \rangle)$ where C is a set of basic or complex events conjuncted with event constructor Opr (such as disjunction(OR), conjunction(AND), sequence(SEQ), negation(NOT), etc.), t_s and t_e are trigger time and detected time of the complex event.*

3.2 Event Query Definition Language

Complex event is defined with event query language. As event query definition language is not focus of this paper, we use SQL-based declarative pattern matching language proposed in [13] with slight extension as Dejavu [4] to support hybrid event queries. A pattern query is defined with MATCH-RECOGNIZE clause using the syntax as follows:

```

SELECT <selected fields list>
FROM <streams or tables> MATCH_RECOGNIZE (
  [PARTITION BY]
  [ORDER BY <field name>]
  [MEASURES <measure list>]
  [ONE/ALL ROW PER MATCH]
  [AFTER MATCH SKIP TO NEXT ROW/ PAST LAST ROW /...]
  PATTERN (pattern description)
  DEFINE <events constraints list>
  [WINDOW <window specification>]
)

```

PARTITION BY and ORDER BY clauses are used to partition and/or order input event stream with given attributes before pattern matching. Patterns are defined with event constructors in PATTERN clause and constraints are defined in DEFINE clause. WINDOW is used to define a sliding window over the event stream.

3.3 Types of Hybrid Patterns

Definition 3 (Hybrid Pattern, HP). *A pattern $E=E_L+E_A$ that is defined over live stream(E_L) and archived stream(E_A) is called hybrid pattern.*

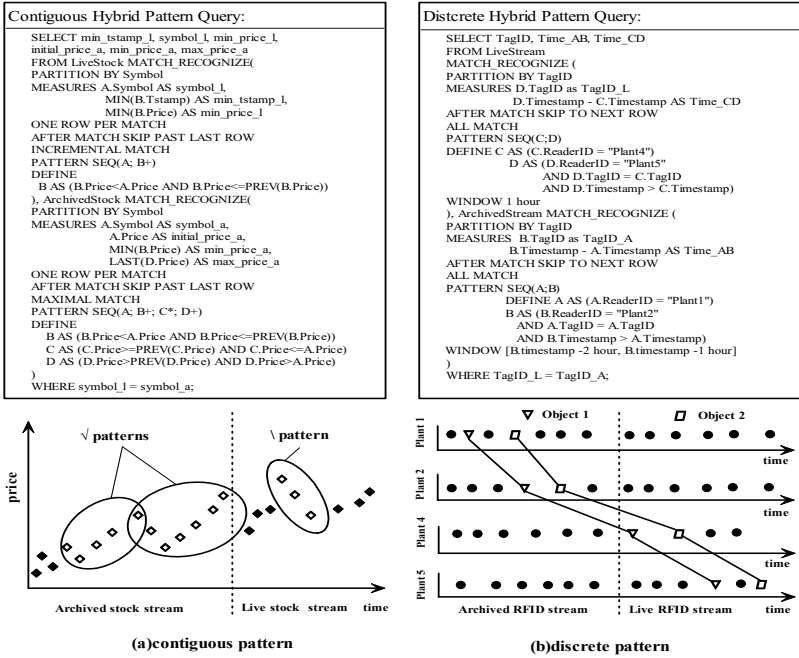


Fig. 1. Illustration of Two Types of Patterns and Queries

Definition 4 (Contiguous Hybrid Pattern, CHP). Suppose event streams are partitioned by event attribute (such as *EventID* or *TypeID*). For an HP E , if events that constitute E are detected in strictly contiguous order (by time) in corresponding partition, the pattern is called *Contiguous Hybrid Pattern*.

Definition 5 (Discrete Hybrid Pattern, DHP). Suppose event streams are partitioned by event attribute. For an HP E , if events that constitute E are detected in discrete order (by time) in corresponding partition, the pattern is called *Discrete Hybrid Pattern*.

Definition 6 (Archived Stream Access Range, Ar). For a pattern $E = E_L + E_A$, range of historical stream accessed by E_A is called *Ar*. *Ar* is defined as $[e.time - r1, e.time - r2]$ where e is a subevent of E_L , $r1$ and $r2$ ($r1 \geq r2 \geq 0$) are start and end points of the accessed historical stream.

In Fig. 1(a) is a contiguous pattern query on stock stream, decrease pattern (\backslash) over live stock stream and tick patterns (\surd) over archived stock stream are contiguous patterns, namely no events that do not satisfy constraints in the DEFINE clause exist in the partition (here by stock symbol) between any of the detected patterns. In Fig. 1(b), discrete pattern is illustrated with an RFID-enabled retail scenario. The event query is to find a sequential pattern $SEQ(C, D)$ over live

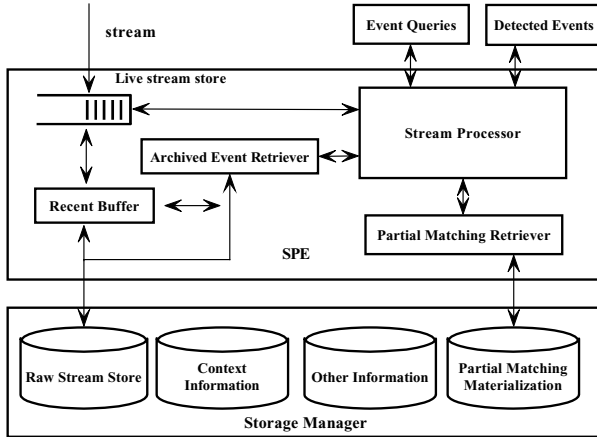


Fig. 2. HYPE System Architecture

streams of RFID readings from plant4 and plant5, then to find another sequential pattern $SEQ(A,B)$ over archived RFID readings from plant1 and plant2, here we can see events of the detected hybrid patterns of object 1 and object 2 disperse in the event streams at different time points.

4 System Design

In this section, we describe system design of HYPE. System architecture of HYPE is shown in Fig. 2. From a high level view, HYPE is a continuous query processor that integrates stream processing with DBMS support in a unified framework.

4.1 Storage Management

In hybrid complex event processing system, live events, historical events, partial match results over live stream and partial match results over historical events need to be monitored in an effective way to support responsive event detection.

Live stream store. In HYPE, raw events periodically generated by streaming application are pushed/pulled from event resources through stream router. Live stream is stored in an internal queue in the shared memory to ensure order of incoming events. As concurrent processes may access the live stream simultaneously, semaphore is used to control concurrent operations.

Recent Buffer. *Recent Buffer* is used to archive live stream into database in bulk and provide hybrid event queries with most recent events which will reduce database access. In HYPE, it is implemented as a queue in main memory with automatically adjustable size according to query requirements. Since events in recent buffer are accessed by stream processor and historical data archiver,

semaphore is used to block concurrent access of the buffer. As we consider discrete pattern, each event may be involved in event detection of E_L and E_A , so raw events are kept in recent buffer in their original order. Note that the size of recent buffer should be addressed carefully. If recent buffer is set too small, when live events are loaded into the buffer, old events need to be archived into DB frequently. Additionally, if instances of E_L require historical data access, small recent buffer means less probability of data hits and more DB access are required which will incur great IO. On the other hand, if recent buffer is set too large, insert event streams into DB with large batch will lead to long time database write operations and memory for event detection engine is limited which leads to ineffective system response.

Raw Stream Store. *Raw Stream Store* is used to archive event streams in append-only manner on disk storage. Live streams are inserted into database in bulk via the recent buffer. Operations such as DELETE, UPDATE, and ALTER are not allowed on raw event tables. *Raw Stream Store* in HYPE is implemented over MySQL, indexes are created over event Timestamp (or other attributes if necessary) of the raw event streams.

4.2 Partial Match Materialization

In event processing, partial matches that are not fallen out of the sliding window or the archived stream access interval should be cached for future matching.

Partial Match Caching of Live Stream. Events that initiate a runtime instance over live stream but do not trigger an archived event detection are called *partial match of live stream*. Partial matches of live stream need to be cached until they fall out of the sliding window or there does not exist a match over corresponding archived stream. In HYPE, partial match of a pattern is denoted as $\langle \text{EventID}, \text{States}, \text{AttriList} \rangle$ where *States* records current states of a partial match and *AttriList* keeps a list of attributes that are involved in the partial match. Partial matches of a pattern are organized into Hash tables (*EventID* as key). When the window slides forward, outdated events and partial matches need to be removed from main memory. A B+tree with *Timestamp* as the key is constructed over hash table of partial events (*Timestamp* of a partial match is the latest updated time) and raw events, the leaf nodes of the B+tree point to corresponding partial matches or raw events in the Hash table. During garbage collection (window sliding forward), HYPE can remove the outdated events by pruning the nodes of the B+tree that fall out of the window in batch manner.

Partial Match Caching of Archived Stream. When a pattern is detected on live stream, pattern match on the archived (historical) stream is initiated. Generally, the history of an event instance intersects with history of another event instance which results in re-accessing and reprocessing some portion of the archived streams. So to decrease archived data access overhead and avoid recomputing of overlapped archived stream, previously found partial matches over archived streams are cached in main memory. For hybrid pattern match, the semantic of accessing archived streams varies with pattern queries. For example, as shown

in Fig. 1(a), contiguous hybrid pattern queries access historical data from the time a new event is detected on live stream to the beginning of historical stream, historical results (such as all tick patterns found over historical stock stream) are cached in main memory, no update is needed for partial matches of contiguous pattern since older events will not participate in later pattern match, outdated historical matches can also be deleted as time passes. However, the case is not true for discrete pattern pattern match. As shown in Fig. 1(b), historical stream access of discrete pattern query is bounded by a window operator and sub-events of a discrete complex pattern are dispersed at different places of the stream. So partial matches of discrete pattern may be updated when new historical data are accessed and processed. To provide fast update access of partial matches, we built an B+tree index on start time of the partial match.

Note that, here we mainly address caching partial matches in the main memory, if the window is very large, , partial matches of E_L and E_A residing in the window increase greatly as event detection proceeds. So to guarantee event detection completeness and avoid system overloading, partial matches should be materialized into DB. Issues on when and how to materialize and update partial matches is left as future work.

4.3 Event Detection Scheduling Algorithms

During continuous event processing, multiple instances of E_L of the same event query or different event queries can be detected successively over the stream. Instance of E_L may trigger event detection of E_A over archived streams. Since detection of E_A involves different length of archived stream loading and computation, response time of event detection will differ under different scheduling strategies. We introduce instance based and sub-window based event processing scheduling algorithms.

Instance based Scheduling, IS. One nature scheduling method is one-instance scheduling as shown in Algorithm 1. Namely for $E=E_L+E_A$, when an instance of E_L reaches full match, then detection of E_A is initiated over corresponding archived stream. One-instance scheduling method does not consider sharing

Algorithm 1. Instance based Scheduling Algorithm

```

1: for each  $e$  in  $W$  do
2:   for ( $i=1; i \leq N_E; i++$ ) do
3:     Detect  $E_{L_i}$  of  $E_i$  over  $e$ ;
4:     for each instance  $I_{L_j}$  of  $E_{L_i}$  that reaches full match do
5:       Compute  $Ar_j$  of  $I_{L_j}$  and load  $Ar_j$  from DB or recent buffer;
6:       Detect instance  $I_{A_j}$  of  $E_{A_i}$  over  $Ar_j$ ;
7:       if  $I_{A_j}$  reaches full match then
8:          $ComplexEvt = I_{L_j} + I_{A_j}$ ;  $Output(ComplexEvt)$ ;
9:       end if
10:    end for
11:  end for
12: end for

```

of E_A computation over archived streams between nearby E_L instances which will incur great historical data loading overhead and re-computation of archived streams. The complexity of IS is $O(|W| * N_E * N_{I_L})$ where $|W|$ is window length, N_E is event queries numbers and N_{I_L} is instance numbers of E_L over W . The cost of detect E_i with IS is given by:

$$Cost_{IS} = \sum_{E_L \text{ of } E_i} \left(\sum_{I_{L_j} \text{ of } E_L} (Cost(I_{L_j}) + Cost(Ar_j) + Cost(I_{A_j})) \right)$$

where $Cost(I_{L_j})$ is cost of finding instance I_{L_j} of E_L of E_i and $Cost(Ar_j)$ is the cost of loading the archived stream of I_{L_j} from DB and recent buffer. Note that $Cost(Ar_j)$ includes time of inserting events into database before I_{L_j} is detected. $Cost(I_{A_j})$ is the cost of processing E_A of E_i over Ar_j .

Algorithm 2. Sub-Window based Scheduling Algorithm

```

1: for each  $W_S$  of  $W$  do
2:   for (i=1; i≤n; i++) do
3:     Detect  $E_{L_i}$  of  $E_i$  over  $W_S$ ;
4:      $IList_L = \{I_{L_i} \text{ that reaches full match}\}$ ;
5:   end for
6:   Compute  $Ar_{W_S}$  of  $IList_L$  and load  $Ar_{W_S}$  from DB or recent buffer;
7:   Detect  $E_A$  over  $Ar_{W_S}$ ;
8:    $IList_A = \{I_{A_j} \text{ that reaches full match}\}$ ;
9:   Output( $IList_L, IList_A, EventID$ );
10: end for
  
```

Sub-Window based Scheduling, SWS. Sub-Window based Scheduling is shown in Algorithm 2. Sub-window based scheduling is to process instances of E_i in batch manner over sub-window of sliding window and archived streams in assumption that some E_L instances that will require overlapped historical event streams exist in the same sub-window. Loading and computing archived streams in batch thus reduce archived stream access requirements (communications between processes) and provide fast response for overall complex event output. The complexity of SWS is $O(\sum_{i=1}^n |W_{si}| * (N_{E_L}^{W_{si}} + N_{E_A}^{Ar_i}))$ where $|W_{si}|$ is the i th sub-window length, $N_{E_L}^{W_{si}}$ is instance numbers of E_L over W_{si} and $N_{E_A}^{Ar_i}$ is instance numbers of E_A over Ar_i . Note the length of the sub-window is crucial for responsive event output in SWS: if the sub-window is too small, SWS is likely to function as IS; if the sub-window is too large, event output of some previously found E_L instances are greatly postponed due to intensive computing over large range of live and historical event streams. The cost of detect E_i with SWS is given by:

$$Cost_{SWS} = \sum_{SW_i} (Cost(I_L) + Cost(Ar_{SW_i})) + \sum_{Ar_{SW_i}} Cost(I_A)$$

where $Cost(I_L)$ is cost of finding instance of E_L over SW_i . $Cost(Ar_{SW_i})$ is cost of loading the archived stream of full match instances in SW_i . $\sum_{Ar_{SW_i}} Cost(I_A)$ is the cost of processing all defined E_A over Ar_{SW_i} .

Table 1. Experimental parameters

Dataset parameters	Notation	Default Value
Data skewness	α	1.0, $N(\mu, \sigma^2)$
Sliding window	W	Landmark
Recent Buffer	R	1000 time unit
Sub-window length	SW	1000 time unit
Archived stream access range	Ar	an interval defined in E_A

5 Experimental Analysis

In this section, we present experimental results of techniques proposed in previous sections in a prototype of HYPE. The prototype system is implemented with C++ in Visual Studio2008 on Windows XP. My SQL is chosen as the archiver of event stream. All experiments are run on Pentium(R) E5200 Dual Core 2.52GHz CPU with 2GB RAM.

In order to evaluate HYPE under different settings, we have performed a set of experiments on sythetic RFID data sets. The sythetic data set simulates an RFID-enabled supply chain scenario in Fig. 1(b). All experiments are tested with sequential patterns(*SEQ*). Live events are loaded into query processing engine in *push-mode*. Parameters of experiments are listed in Table 1.

The proposed scheduling algorithms are evaluated by varying values of R , W , α and SW . The experimental results are shown in Fig. 3. By setting recent buffer length with 500, 1000 and 1500(time units), sub-window size with 100, 1000, 5000, 10000(time units), sliding window size from 10 to 50 thousands, we have tested response time of *IS* and *SWS*. Results from Fig. 3(a)-(c) show that *SWS* with sub window size 1000 and 100 performs much better than *IS*, but as subwindow size increases(5000,10000), *IS* performs better than *SWS*. Results from Fig. 3(a)-(c) also mean that subwindow size is directly related to response time of hybrid event processing, if SW is set too small, hybrid event detection is more likely to proceed in *IS* mode, if SW is set too large(e.g.10000), event processing is postponed due to intensive computation of live stream, large portion of archived stream loading and computation. In hybrid event processing, database operations are another key factor that influences response time. From Fig. 3(d)-(f) we can see that *IS* consumes more DB access than *SWS* in most cases (except for window size 10000, subwindow size 5000)because of little sharing of overlapped historical data access and computation. So different subwindow size and recent buffer size will cause different database access and response time. In this paper, SW and R are set according to data skewness α of the event stream. Here skewness means complex event distribution(here we use normal distribution) over the event stream. Intuitively, if α is large(e.g. $\alpha \geq 0.6$) which means more chance to detect E_L and over E_A over event streams, SW is set with proper value to ensure responsive event output; if α is small($\alpha \leq 0.4$) SW is set with larger value to ensure batch DB access and archived stream processing. Fig. 3(g)-(h)illustrate CPU cost by varying α from 0.2 to 1.0 over different

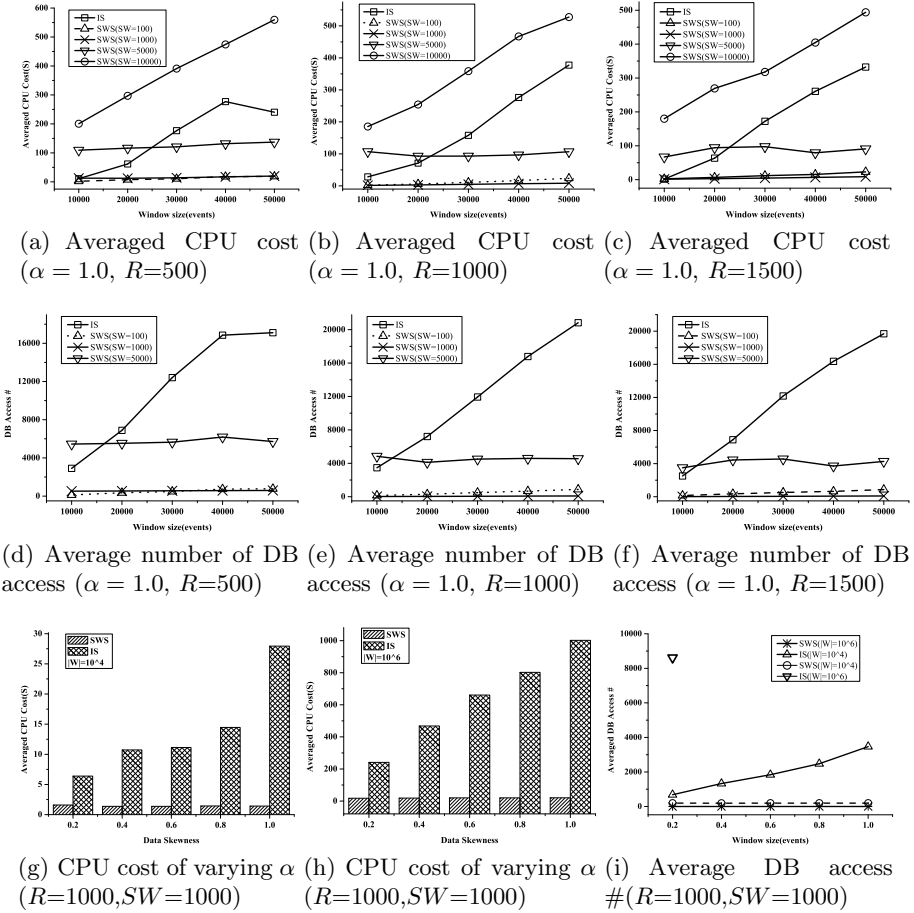


Fig. 3. Experimental results under different parameters

window size. As α increases, IS needs to access DB more times to detect E_A over archived stream. Otherwise, as SWS works in batch manner, database access and computation increases very slow when α increases. Note that in Fig. 3(i), DB access of $IS(W = 10^6)$ is very large, so only DB access at $\alpha=0.2$ is given for ease of illustration.

6 Conclusion

In this paper, we address event detection problems over live and archived streams in a single framework. We present hierarchical stream storage architecture for live and archived streams. Partial matches of live and archived streams are managed in main memory effectively. Two pattern match scheduling methods are proposed

to achieve fairly good event detection response time. At last, a prototype CEP engine is implemented and the proposed methods and algorithms are tested with extensive experiments and the experimental results illustrate that effectiveness of proposed approaches.

Acknowledgments. This work is sponsored partially by National Natural Science Foundation of China (No. 60803043, No. 60873196 and No. 61033007) and the National High-Tech Research and Development Plan of China(863) under Grant No. 2009AA01A404.

References

1. Abadi, D.J., Ahmad, Y., Balazinska, M., Çetintemel, U., Cherniack, M., Hwang, J.H., Lindner, W., Maskey, A., Rasin, A., Ryvkina, E., Tatbul, N., Xing, Y., Zdonik, S.B.: The design of the borealis stream processing engine. In: CIDR, pp. 277–289 (2005)
2. Chandrasekaran, S., Cooper, O., Deshpande, A., Franklin, M.J., Hellerstein, J.M., Hong, W., Krishnamurthy, S., Madden, S., Raman, V., Reiss, F., Shah, M.A.: Telegraphcq: Continuous dataflow processing for an uncertain world. In: CIDR (2003)
3. Franklin, M.J., Jeffery, S.R., Krishnamurthy, S., Reiss, F., Rizvi, S., Wu, E., Cooper, O., Edakkunni, A., Hong, W.: Design considerations for high fan-in systems: The hifi approach. In: CIDR, pp. 290–304 (2005)
4. Dindar, N., Güç, B., Lau, P., Ozal, A., Soner, M., Tatbul, N.: Dejavu: declarative pattern matching over live and archived streams of events. In: SIGMOD Conference, pp. 1023–1026 (2009)
5. Chandrasekaran, S.: Query processing over live and archived data streams. PhD thesis, Berkeley, CA, USA (2005)
6. Balazinska, M., Kwon, Y., Kuchta, N., Lee, D.: Moirae: History-enhanced monitoring. In: CIDR, pp. 375–386 (2007)
7. Soner, M.: Modeling and efficiently processing hybrid pattern matching queries over live and archived streams. Master's thesis, ETH Zurich, Switzerland (2010)
8. Zimmer, D., Unland, R.: On the semantics of complex events in active database management systems. In: ICDE, pp. 392–399 (1999)
9. Wu, E., Diao, Y., Rizvi, S.: High-performance complex event processing over streams. In: SIGMOD Conference, pp. 407–418 (2006)
10. Demers, A.J., Gehrke, J., Panda, B., Riedewald, M., Sharma, V., White, W.M.: Cayuga: A general purpose event monitoring system. In: CIDR, pp. 412–422 (2007)
11. Mei, Y., Madden, S.: Zstream: a cost-based query processor for adaptively detecting composite events. In: SIGMOD Conference, pp. 193–206 (2009)
12. Tufte, K., Li, J., Maier, D., Papadimos, V., Bertini, R.L., Rucker, J.: Travel time estimation using niagarast and latte. In: SIGMOD Conference, pp. 1091–1093 (2007)
13. Zemke, F., Witkowski, A., Cherniak, M., Colby, L.: Pattern matching in sequences of rows. Technical report, ANSI Standard Proposal
14. Golab, L., Johnson, T., Seidel, J.S., Shkapenyuk, V.: Stream warehousing with datadepot. In: SIGMOD Conference, pp. 847–854 (2009)
15. Reiss, F., Stockinger, K., Wu, K., Shoshani, A., Hellerstein, J.M.: Enabling real-time querying of live and historical stream data. In: SSDBM, p. 28 (2007)

Renda-RX: A Benchmark for Evaluating XML-Relational Database System^{*}

Xiao Zhang^{1,2}, Kuicheng Liu^{1,2}, Li Zou^{1,2}, Xiaoyong Du^{1,2}, and Shan Wang^{1,2}

¹ Key Labs of Data Engineering and Knowledge Engineering, Beijing 100872, China

² School of Information, Renmin University of China, Beijing 100872, China
{zhangxiao, lkc, zouli_happy, duyong, swang}@ruc.edu.cn

Abstract. This paper proposes a new benchmark, Renda-RX, for evaluating XML-Relational database management systems which can store and process both relational and XML data. To our knowledge, it is the first time to consider a performance benchmark on mixed data, although there are TPC series for relational data, and X Bench, XMark, TPoX etc. for XML data. We summarize the requirements of a benchmark for database systems, and describe the construction of our benchmark. It includes scenario selection, data storage, transaction design and result report etc. Based on the benchmark design, we also develop a test tool and successfully get some performance results on a commercial XRDBMS, System-X.

Keywords: Benchmark, performance, Renda-RX, XML, transaction processing, XRDBMS.

1 Introduction

As one of the industry standards, XML data is widely used in various areas, such as web, medicare system, multimedia, finance, etc. In particular, the co-existence of XML data and relational data has become popular in the real-world. To take the medicare system as an example, the patients' basic information can be stored in a relation while the other detailed medical records are stored in form of XML. In fact, many XML features have been supported in most of mainstream RDBMSs, such as Oracle [12], DB2 [13], MySQL [14], SQLServer [15], PostgreSQL [16] and so on. For space utilization and efficiency reason, transformation between XML data and relational data must be traded off carefully in an XML-Relational DBMS, *XRDBMS* for short, so as to improve the performance of executing transactions accessing both XML data and relational data simultaneously.

Benchmarks are designed to simulate particular types of workload on a DBMS based on a certain real-world scenario in order to get a performance result that is very useful for the improvement of DBMS technology.

^{*} This work is partly supported by National 863 High Tech. Program(No. 2009AA01Z149), the Important National Science & Technology Specific Projects of China ("HGJ" Projects, No.2010ZX01042-002-002-03), the National Natural Science Foundation of China (No.61070054), the Fundamental Research Funds for the Central Universities (the Research Funds of Renmin University of China, No.10XNI018).

Unfortunately, there is no such a standard benchmark to evaluate those existing XRDBSs. Neither the Transaction Processing Council (TPC) nor the Standard Performance Evaluation Corporation (SPEC) has proposed such a benchmark for transactions on hybrid data.

To our best knowledge, Renda-RX benchmark is the first benchmark dedicatedly focusing on transactions on mixed data, i.e. Relational-data and XML-data together, and further facilitating the improvement of XRDBMS by testing and comparing the performance of their counterparts. This is our most significant contribution.

The rest of the paper is organized as follows. In section 2, we present the related work of the benchmark. The requirements of Renda-RX benchmark are discussed in Section 3. We describe the detail of Renda-RX benchmark and its definition of XML data and DTDs in Section 4. Section 5 presents the workload of Renda-RX benchmark. The architecture of Renda-RX benchmark is shown in the Section 6. In Section 7 we present our test on a commercial system System-X.

2 Related Work

There are a number of well-known benchmarks for RDBMSs, such as TPC-C, TPC-E, TPC-H, which are all proposed by Transaction Processing Performance Council (TPC). TPC-C [1] is for On-Line Transaction Processing (OLTP) system. It simulates a large commodity wholesale marketing company where there are a number of goods warehouse distributed in several different areas. All of its transactions only take into account of relational data. TPC-E [2] is another benchmark for RDBMS based on the model of the New York Stock Exchange and simulates a series of backstage data processing and stock exchange operations of forestage customers, such as querying account, online exchange and survey of market and etc. As for TPC-H [3], it is an ad-hoc, decision support benchmark which is used to solve some real-world questions, e.g. management of supply and demand, analysis of customer satisfaction. The benchmark is also designed to evaluate the performance of RDBMSs.

All these benchmarks are designed to RDBMSs, so during the whole testing process the logical model of data must keep unchanged. XML, however is often used for its flexibility [17], for example, to add or remove a node of XML data, thus the logical model of data will be changed. Obviously, these TPC benchmarks are inappropriate to evaluate the performance of XML-DBMSs. To solve this problem, a number of benchmarks aiming at pure XML-DBMSs have been proposed, including TPoX, XMark, XBench, XOO7, XMach-1 and others. TPoX [5] is proposed by IBM and simulates a financial multi-user workload with pure XML data conforming to the FIXML standard [11]. It not only evaluates the query processor but also all parts of the database system under test, including storage, indexing, logging, transaction processing, and concurrency control. XMark [8] is a single-user benchmark, and the data is based on an Internet auction site. It provides a much comprehensive set of queries, but there is not schema or DTD involved. XBench [6] is a family of XML benchmarks, designed to capture the variety of database schema and workload. The database applications are divided into data-centric applications and text-centric applications, and in terms of data characteristics, the documents can be identified: single document and multiple document. XOO7 [7] is a benchmark for XMLMS

(management system), which is an XML version of the OO7 benchmark enriched with relational document and navigational queries that are specific and critical for XML database system. The data model of it is mapped from OO7. XMach-1 [9] is a scalable multi-user benchmark based on a Web application to evaluate the performance of XML-DBMS and different kinds of XML data, such as test documents, schema-less data and structured data. It defines many different XML queries and updates which cover a lot of function tests.

Compared with the benchmarks for RDBMSs, these benchmarks are at the other end of the spectrum. They just take pure XML data into consideration which means they can only focus on pure XML transactions. Therefore, they are also inappropriate when transactions contain operations on relational-data and XML-data at the same time.

3 Requirements of Database Benchmark

Based on the analysis of the existing benchmarks and the real-world transactions on mixed data, we outline the requirements of Renda-RX benchmark below.

Scenario. As an acknowledge OLTP benchmark aiming at RDBMS, TPC-C simulates a large commodity wholesale marketing company where there are a number of goods warehouse distributed in several different areas. Another representative benchmark for pure XML-DBMS is TPoX, which is based on an on-line financial trading scenario and uses FIXML to model some of its data. Different from TPC-C and TPoX, Renda-RX simulates a scenario of health-care procedures.

DTD variability. The schema of relational data should keep unchanged during the test, so all benchmarks designed for RDBMS requires the static data model, for example, in TPC-C [1], the rule of table schema is defined clearly. However, one important reason for the success of XML is the flexibility of schema or DTD, i.e. to describe data that is difficult to be specified in a uniform model. In some XML benchmarks, the requirement has been addressed, such as XMach-1, TPoX. There are many different DTDs in XMach-1 benchmark, and TPoX uses FIXML as its XML schema. Based on the real-world situation, we define three DTDs in Renda-RX benchmark.

Pressure test. To evaluate the database performance, pressure test is an important metric. In TPC-C, the pressure test is embodied in large-scale data and multi-user. This test is also adopted in TPoX benchmark. Actually, most real-world database applications serve multi-user and the scale of data increase with time going. Therefore, Renda-RX benchmark also supports pressure test.

Aiming at transactions. In real-world applications, a transaction usually contains a serial of queries rather than only one query. As a pattern, TPC-C has five transactions and each transaction contains a serial of queries. The “query” here means query, insert, update or delete. In Renda-RX benchmark, we define seven transactions based on the scenario and each transaction involving several different operations.

Reports. One significant motivation of a benchmark is to present the performance of various systems under test. According to TPC-C specification, several reports are

generated, such as the frequency distribution of response times of all transactions, a graph of the throughput of the New-Order transaction etc. Renda-RX benchmark defines its test reports, (i) the number of each transaction executed during the measurement, (ii) the rate of each transaction executed in all the transactions, (iii) a graph of frequency distribution of response time of all transactions, (iv) a graph describing the throughput of registering transaction.

All the requirements above will be described in Renda-RX benchmark next section in detail.

4 Renda-RX Benchmark

4.1 Scenario

As an application-oriented benchmark, the scenario is one of important parts and the basis of Renda-RX design. Its scenario is the health-care procedures and some relevant transactions. Each hospital is responsible for 6000 patients and there are 20 different departments. Meanwhile, a department has 20 doctors presumably. The main procedure is that every patient registers for a doctor, for example, according to his/her resume, i.e. experience and reputation, and the doctor writes out a prescription, then the patient pays for the prescription and gets medicines.

4.2 The Definition of Tables

After analyzing and surveying of the real-world applications and the measurement requirements of XML data and relational data, we designed nine tables in the benchmark, Hospital, Department, Doctor, Patient, Prescription, Registration, New_Registration, Payment and Medicine.

Figure 1 gives the relationship of the nine tables. The numbers in the diagram illustrate the database population requirements; the numbers in the entity blocks represent the cardinality of the tables, and the numbers are factored by “H”, the number of Hospitals, to illustrate the database scaling; “+” refers to no less than; “0-1” means 0 or 1.

These tables fall into two categories: pure relational tables containing relational data alone, such as Hospital, Department, Medicine, Registration, New_Registration and Payment tables and several hybrid tables comprising of relational fields and XML fields together such as Doctor, Patient, Prescription tables. Table 1 shows the Hospital table, a pure relational table, and acts as the basis for the expansion of data scale in the benchmark. Table 2 describes the definition of the Doctor table which is a mixed table containing relational data and XML data. The reasons why we definite mixed tables can be found in followings.

4.3 XML Data

In this section, we describe the definition of XML data in the hybrid tables, and the reasons why we adopt XML type in the fields. In our test model, Doctor, Patient and Prescription are the most important entities containing the XML field.

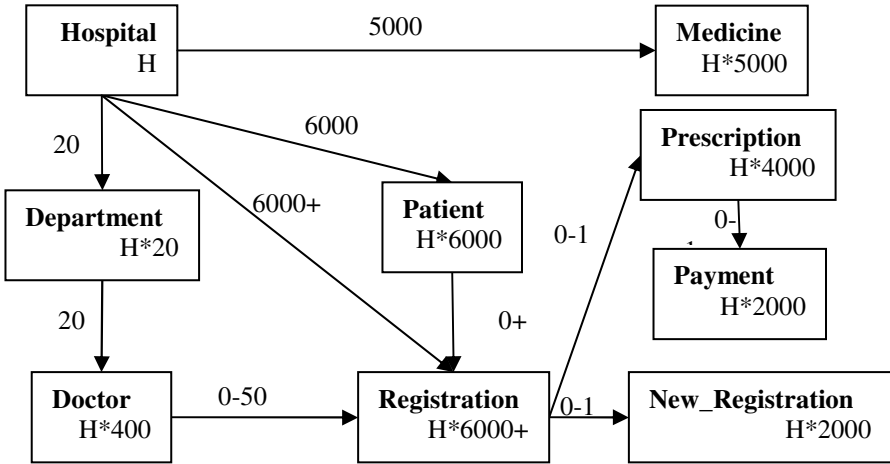


Fig. 1. The Relationship of Tables

Table 1. Hospital Table

Field Name	Field Definition	Comments
hp_id	2*H unique ids	the id of the hospital
hp_name	variable txt, size 10	the name of the hospital
hp_street1	variable txt, size 20	the address of the hospital
hp_street2	variable txt, size 20	the address of the hospital
hp_city	variable txt, size 20	the city of the hospital
hp_director	variable text, size 10	the leader of the hospital
hp_zip	fixed text, size 6	the zip of the hospital
hp_contact	fixed text, size 11	the contact of the hospital
hp_next_r_id	100,000 unique ids	the next available id for registration
Primary key: (hp_id)		

Table 2. Doctor Table

Field Name	Field Definition	Comments
dr_id	40 unique ids	the id of the doctor
dr_hp_id	2*H unique ids	the id of the hospital
dr_dtp_id	d unique ids	the id of the department
dr_name	variable text, size10	the name of the doctor
dr_gender	M or F	“male” or “female”
dr_age	numeric(2)	the age of the doctor
dr_r_cnt	numeric(2)	the number of registration of the doctor
dr_r_now_cnt	numeric(2)	the number waiting for the doctor
dr_xml	xml	the resume of the doctor
Primary key: (dr_hp_id, dr_dtp_id, dr_id)		
(dr_hp_id, dr_dtp_id) foreign key, references department(dpt_hp_id, dpt_id)		

Doctor Entity. Supposing we use relational model to describe the Doctor entity, its attributes usually are hospital, department and the personal information of the doctor, and the schema is illustrated in Table 3.

Table 3. A Relational Table of Doctor

dr_id	hp_id	dpt_id	dr_name	dr_gender	dr_age	dr_brief
-------	-------	--------	---------	-----------	--------	----------

In Table 3, dr_brief is a variable text type used to describe the doctor's personal information, such as, experience, studied fields, rewards, etc. If the field, for example, only used to store the information, or only to select the whole text from doctor table, the relational model is available. However, in the real-world applications, some problems may emerge:

1. Because the dr_brief field is no semantic, when we need to get the experience or the speciality of a doctor, we have to do one of these (i) parse the text and use string searching and matching, or (ii) modify the schema by adding a new table to record the personal information of a doctor
2. When assess doctors, we need to calculate the comments of doctors and update the corresponding reward to doctors. Then we should do that (i) parse the text, use string searching and matching, and then update the whole text, or (ii) modify the schema of the table by adding a new attribute to describe the comment of a doctor

So, if we use the relational model to describe a Doctor entity and meet the actual requirements, we have to apply string searching and matching which will cause inconvenience and the cost of time or modify the schema and add a new table, which will lead to the information redundancy. Here, the superiorities of XML type are apparent (i) the text is becoming sensible because of the XML hierarchical structure, (ii) it is very convenient to add or remove a node of XML data, not to worry about the vary of schemas, (iii) querying a string by path query is much more convenient and efficient than string parsing and matching.

Patient Entity. Similar to the Doctor entity, in real applications, the case history of a patient keeps changing. Now the advantages of XML type are almost same as that in Doctor entity.

Prescription Entity. In our daily life, each doctor has his own style to write a prescription and the model of a prescription is usually different from another one, therefore, it is impossible to define a unified fixed model to meet the actual requirements. Therefore, the XML type is more appropriate here.

4.4 DTD

The DTDs [10] are defined based on our survey and analysis of the real-world data, for example, the resume of doctors and the medical records of patients. There are also some simplifications for evaluating, for example, the DTD in prescription table is the same as that in patient because it is convenient to insert information of prescription

into prescription table in case of reconstruction or parse of prescription, which can reduce the cost of time, and redundancy of information.

The Rules of DTD. All the DTDs should be used as-is and in full even if the benchmark data does not use all parts of a DTD [5]. Figure 2 is an example of DTD.

```

<!DOCTYPE doctor>
<!ELEMENT doctor(brief, expert, careers, honors, comment)>

<!ELEMENT brief(profile, loc, contact)>
<!ELEMENT profile(idCard, name, gender)>
<!ELEMENT idCard(#PCDATA)>
<!ELEMENT name(#PCDATA)>
<!ELEMENT gender(#PCDATA)>
<!ELEMENT loc(city, street)>
<!ELEMENT city(#PCDATA)>
<!ELEMENT street(#PCDATA)>
<!ELEMENT contact(emails, tels)>
<!ELEMENT emails(email*)>
<!ELEMENT email(#PCDATA)>
<!ELEMENT tels(tel*)>
<!ELEMENT tel(#PCDATA)>

<!ELEMENT expert(sickness)>
<!ELEMENT sickness(#PCDATA)>

<!ELEMENT careers(career*)>
<!ELEMENT career(period, comp)>
<!ELEMENT period(from, to)>
<!ELEMENT from(#PCDATA)>
<!ELEMENT to(#PCDATA)>
<!ELEMENT comp(dpt)>
<!ELEMENT dpt(title)>
<!ELEMENT title(#PCDATA)>

<!ELEMENT honors(honor*)>
<!ELEMENT honor(date, detail)>
<!ELEMENT date(#PCDATA)>
<!ELEMENT detail(field, case)>
<!ELEMENT field(#PCDATA)>
<!ELEMENT case(org, partners, name)>
<!ELEMENT org(#PCDATA)>
<!ELEMENT partners(idCard*)>
<!ELEMENT id(#PCDATA)>

<!ELEMENT comment(A, B, C)>
<!ELEMENT A(#PCDATA)>
<!ELEMENT B(#PCDATA)>
<!ELEMENT C(#PCDATA)>

```

Fig. 2. The DTD of Doctor Table

5 Workload Design

In this section, the details about transactions and the scalability of data volume are presented. The Renda-RX benchmark is executed in three steps. Firstly, populate the database, and the rule of this operation should follow the relationship shown in Figure 1, especially the rates among those tables. And the time of accomplishing this step is not included in the whole measurement time. Secondly, perform a multi-user workload on the populated database. Finally, get the test reports by analyzing the log files generated during the testing. And this is a similar approach to that in the TPC-C benchmark.

5.1 Transactions

The benchmark workload consists of a set of transactions executing queries, insertions, updates and deletions. All operations can be applied on either XML data or relational data or both. To meet the requirements of performance measurement of database system in the real-world scene, in Renda-RX benchmark we define the transactions listed in Table 4.

Table 4. The Description of Transactions

No	Transaction Type	Measure Type	Frequency	Operations
T1	registering transaction	pure relational	high	query update insert
T2	diagnosing transaction	mixed	high	query delete update insert
T3	pricing transaction	mixed	high	query delete update
T4	newMedicine transaction	pure relational	low	insert
T5	doctorProfile transaction	mixed or pure XML	low	query
T6	patientProfile transaction	mixed or pure XML	low	query
T7	assessment transaction	pure XML	low	update delete

The Ratio of Transactions

As the execution of transactions in real-world and the specification of the existing benchmarks [1, 5], we define the ratio of each transaction in Table 5.

Table 5 shows the percentage of each transaction execution. In the row of transaction T1, “r” means the number of registering transaction execution and “a” represents the total frequencies of all transactions executed. The ratios of T2 to T7 transactions are required to be around the shown number and the error range should be under 0.2%, and the ratio of registering transaction is a dynamic number. The reason is that we make the throughput of registering transaction as the final performance objective of the database system under testing. If the ratio is not set, there may be a high performance result of registering transaction at the cost of abortion of other transactions, which certainly cannot be considered as a good result. And usually each registering transaction approximately corresponds to one diagnosing transaction and one pricing transaction, so the ratios of all the transactions are set in this way.

Table 5. The Ratio of Transactions

No	Transaction Type	Ratio
T1	registering transaction	r/a
T2	diagnosing transaction	28%
T3	pricing transaction	28%
T4	newMedicine transaction	2%
T5	doctorProfile transaction	6%
T6	patientProfile transaction	6%
T7	assessment transaction	2%

5.2 Renda-RX Scalability

Renda-RX benchmark is suitable for pressure test, and the scalability makes it feasible. The throughput of the Renda-RX benchmark is related to the activity of the terminals connected to each hospital. To increase the throughput, more hospitals must be configured. In our proposed benchmark, we take the number of hospitals as the baseline factor of scalability, and the cardinality of all the other tables is a function of the number of configured hospitals.

Rules of Scalability. When extending the scale of data, the rate among these tables should be kept unchanged, which means all the tables should be extended proportionally. The proposed ratios among the tables can be found in Figure 1.

6 The Architecture of Renda-RX Benchmark

In this section, we introduce the framework of Renda-RX benchmark. The Renda-RX benchmark adopts the C/S architecture similar to the definition in TPCC-UVa [4]. In the architecture, there are some Clients and only one Server. Figure 3 shows the architecture of Renda-RX benchmark. And the Client model and Server model are following.

RTE (Remote Terminal Emulator) plays the role of Client. There is one RTE process corresponding to each active terminal during the execution of the benchmark and each RTE runs as an individual process. RTEs form the pressure system in the Renda-RX benchmark. During the whole course of testing, the RTEs generate new transaction requests according to the seven transactions listed above continuously, and send the transaction to Server.

TM (Transaction Monitor) works as Server in the benchmark, which is connected to the under test database system, passing the requests from all the RTEs and feeding back the results. Once receiving a transaction request, TM sends the transaction to the underlying database system, and the final result is reflected in the database. All the transactions are executed in their reaching order.

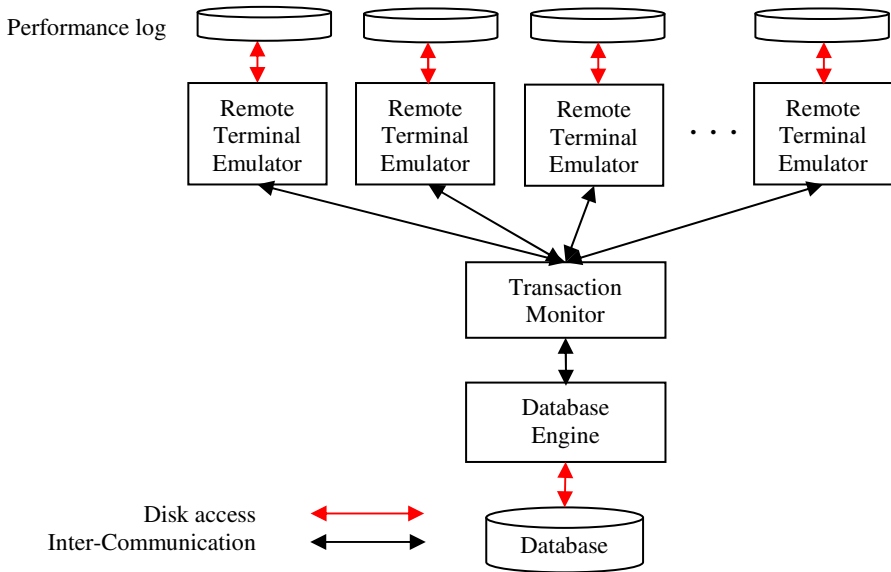


Fig. 3. The Architecture of Renda-RX

7 One Sample Test on System-X

In this section, we discuss a sample test of Renda-RX benchmark on System-X which has provided pure XML technology. To efficiently process XML data, System-X supports SQL/XML and XQuery through a mixed compiler and processing engine. The usage of schema or DTD in System-X is optional, and we just use DTDs to generate XML data and insert the data into tables as a string or BLOB.

The experiment is conducted on a machine with 2.66GHZ Intel Core2 Quad CPU, 2G main memory and 500G disk running Windows 7. And all the experiments are conducted on this computer.

In our test, the data is generated with the nine tables in Figure 1 and all the rules comply with our definition. The number of hospitals is set to 10. After populating the database, we execute a multi-user test with transactions in Table 4 where the ratio of each transaction shown in Table 5.

The total number of the transactions is:36067
 The number of registering transaction is:10090 ; the rate is: 27.98%
 The number of diagnosing transaction is:10133; the rate is: 28.09%
 The number of pricing transaction is:10091; the rate is: 27.98%
 The number of newMedicine transaction is:712; the rate is:1.97%
 The number of doctorProfile transaction is: 2154 ; the rate is: 5.97%
 The number of patientProfile transaction is: 2177; the rate is: 6.04%
 The number of assessment transaction is: 710; the rate is: 1.97%

Fig. 4. Results Summary of a Test on System-X

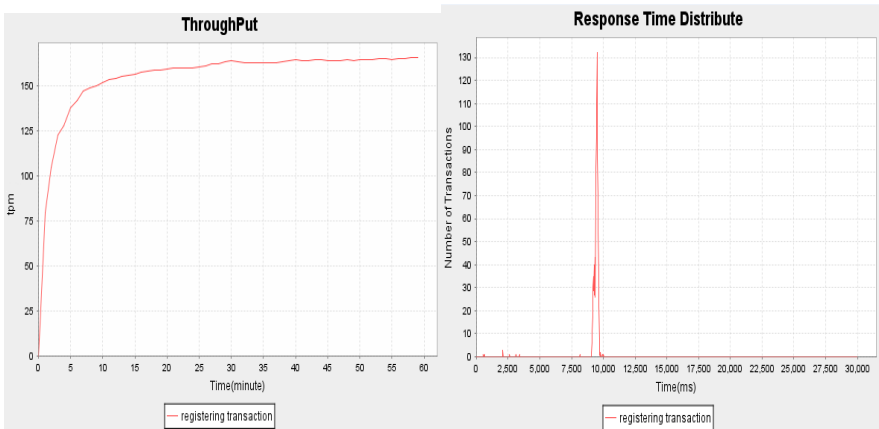


Fig. 5. Throughput and Response Time Distribute of Registering Transaction

Following are the reports of a 60 minutes test with 10 hospital and 100 RTEs on System-X. Figure 4 is the total test results of all the transactions and Figure 5 shows the tpm (Transaction Per Minute) and Response Time Distribute of Registering transaction.

8 Summary and Conclusion

In this paper we main on evaluating XRDBMS. It is an application-oriented benchmark based on a real-ly describe the design of the Renda-RX benchmark focusing world scenario of health-care procedures. The DTDs, data and transactions of Renda-RX are all designed based on our analysis and comparison of the current benchmarks and the real-world transactions on mixed data, i.e. either relational data or XML data, or both.

In conclusion, the contributions of this paper are as follows: a). Renda-RX tries to bridge the gap between relational benchmark and pure XML benchmark. To the best of our knowledge, Renda-RX benchmark is the first benchmark aiming at evaluating the performance of XRDBMSs. b). As an application-oriented benchmark, we describe the definition of the scenario in Renda-RX in detail, including tables, XML data, DTDs, transactions and so on. The scenario is from medicare system and applicable. c). A sample test tool of Renda-RX benchmark on a commercial system System-X has been implemented and the result is according to our expectation.

In fact, the current version of Renda-RX benchmark needs more refinement, such as more practical ratio for percentage of XML data, e-commerce mode and so on. We will keep working out these problems and welcome other peers to improve it. In addition, we will try to make it public so as to evolve it faster. We believe that the following versions of Renda-RX can evaluate the performance of XRDBMS in a more feasible and fair way.

References

1. TPC benchmark C version 5.11.0 Transaction Processing Performance Council
2. TPC benchmark E version 1.12.0 Transaction Processing Performance Council
3. TPC benchmark H version 2.14.0 Transaction Processing Performance Council
4. Llanos, D.R.: TPCC-UVa: An Open-Source TPC-C Implementation for Global Performance Measurement of Computer Systems. SIGMOD Record (2006)
5. Nicola, M., Gonzalez, A., Xie, K., Schiefer, B.: Transaction Processing over XML (TPoX), version 2.0
6. Yao, B.B., Tamer Ozsu, M., Khandelwal, N.: XBench Benchmark and Performance Testing of XML DBMSs. In: ICDE 2004 (2004)
7. Bressan, S., Lee, M.L., Li, Y.G., Lacroix, Z., Nambiar, U.: The XOO7 XML Management System Benchmark (2004)
8. Schmidt, A., Waas, F., Kersten, M., Carey, M.J., Manolescu, I., Busse, R.: XMark: A Benchmark for XML Data Management. In: VLDB 2002 (2002)
9. Böhme, T., Rahm, E.: XMach-1: A Benchmark for XML Data Management. In: BTW 2001 (2001)
10. XML DTD - An Introduction to XML Document Type Definitions, <http://www.xmlfiles.com/dtd/>
11. Chicago Mercantile Exchange: The Business Case for FIXML (2004), <http://www.cme.com/files/BusinessCaseFIXML.ppt>
12. Oracle 11g, <http://www.oracle.com/pls/db111/homepage>
13. DB2 9 pureXML Guide (January 2007), <http://www.ibm.com/redbooks>
14. MySQL 5.1, <http://dev.mysql.com/doc/>
15. XML Support in Microsoft SQL Server 2005 (2005), [http://msdn.microsoft.com/en-us/library/ms345117\(v=sql.90\).aspx](http://msdn.microsoft.com/en-us/library/ms345117(v=sql.90).aspx)
16. PostgreSQL 9.0.3 XML Functions, <http://www.postgresql.org/docs/9.0>
17. Extensible Markup Language (XML), <http://www.w3.org/XML/>

Energy-Conserving Fragment Methods for Skewed XML Data Access in Push-Based Broadcast

Jingjing Wu, Peng Liu, Lu Gan, Yongrui Qin, and Weiwei Sun

School of Computer Science, Fudan University, No.825, Zhangheng Rd.,
Pudong New District, Shanghai, China

{wj, liupeng, ganlu, yrqin, wwsun}@fudan.edu.cn

Abstract. Broadcasting XML data via wireless channel is an efficient way for disseminating semi-structured information and attracts more interests of researchers. As different parts of a piece of XML information have different access probability, fragmenting the original XML data intelligently can improve the broadcast efficiency. Existing works focus on splitting XML documents according to the queries in on-demand mode, but the split results have redundancy. In this paper, we propose a novel scheme of fragmenting XML data in push-based broadcast. First, a linear algorithm, whose idea is from an efficient document splitting algorithm in on-demand mode is provided. Then, an optimized algorithm with a little more time complexity is proposed. Existing air indexing and scheduling techniques are proved to work well under this scheme. Finally, experimental results show that the fragment methods improve the broadcast efficiency a lot by bringing a little auxiliary information.

Keywords: XML, push-based broadcast, skewed data, organization.

1 Introduction

In recent years, there is an increasing interest of using mobile devices (e.g., PDAs, palmtops, cellular phones, etc.) to gain information in wireless environment. Data broadcast is an efficient way for information delivery, as it allows an arbitrary number of mobile clients to access data simultaneously. The types of broadcast data are boundless, including live news, stock quotes, and traffic information.

As mobile clients are usually powered by batteries with limited capacity, access efficiency and energy conservation are two crucial issues in wireless broadcast system. Accordingly, the access time and tuning time are employed as the major performance metrics [1]:

1. Access Time (shorted as *AT*): The time elapsed from the moment a query is issued to the moment it is answered.
2. Tuning Time (shorted as *TT*): The time a mobile client stays in active mode to receive the requested data items and index information.

In broadcast, data are sent to all users residing in the broadcast area, and it is up to users to retrieve the data they are interested in. Air indexing technique is used to improve TT by including a little auxiliary information about the arrival times of data items. On the other hand, scheduling technique, which determines the content and order of broadcast data, is adopted to improve AT under different broadcast modes. There are two typical broadcast modes [1]:

1. *push-based* broadcast: The server disseminates data periodically and mobile users access the data selectively.
2. *on-demand* broadcast: data are sent based on the queries submitted by mobile users.

Traditional broadcast data are formatted as (key, value) like records in relational database, and mobile users retrieve the data item by its key value. However, more complex format of data and queries are also required, such as semi-structured data. Over the past years, XML has gained a great of popularity as the de facto standard for data integration and an effective format for data exchange. Therefore, broadcasting XML data via wireless channel is a novel research issue, which attracts interest of many researchers.

Existing researches on XML data broadcast pay more attention to indexing [2][3][4][5] and scheduling [6][7] techniques under different XML data organization schemes. Two typical schemes are concluded from related works:

1. *Node-level*: an XML document can be represented by a tree of nodes as proposed in the XML data model. The types of nodes are element nodes, text nodes and attribute nodes. Each node is the unit of data for retrieving.
2. *Document-level*: broadcast data are XML documents set and an XML document is the unit data for retrieving. Each XML document is treated as plain text.

The *node-level* scheme improves TT as the mobile users download the precise information they needed. While the auxiliary information which maintains the structure of XML data results in a larger AT . The *document-level* scheme is well designed for scheduling and air indexing technique. However, more redundant information of XML data may be retrieved by mobile users which enlarge the TT . We will give a detailed description of these two schemes in Section 2.

In this paper, we focus on research of the fragmenting methods for skewed XML data access in *push-based* broadcast. Scheduling data item by access probability is a meaningful issue in traditional data broadcast [8][9][10]. To the best of our knowledge, few work pay attention to the skewed XML data access in data broadcast environment. We proposed a *subtree-level* scheme to improve the TT compared with the *document-level* scheme, and increased little additional information. The main contributions of this paper are:

1. A novel organization of XML data named *subtree-level* scheme is proposed. Each XML document is fragmented to pieces according to the access probability of each node.

2. We put forward two efficient fragment algorithms: horizontal fragment algorithm and threshold fragment algorithm.
3. We prove that the two-tier index can be adopted in *subtree-level* scheme by a little modification.

The remainder of the paper is organized as follows. Section 2 presents the related works. The details of *subtree-level* scheme and the two fragment algorithms are presented in Section 3. Section 4 shows the experiments and analyzes the results. Finally, Section 5 concludes our work.

2 Related Works

In *node-level* scheme, [2] separates the structure information and the text values of XML document. The structure information is used as index, and redundant information is reduced by adopting the path summary technique. However, the XPath queries result is ambiguous as the different text values of the element nodes with the same tag name may put in a package, and the users cannot restore the original XML document. [3] separates each XML document into two tree structure. One is index tree, the other is data tree. Thus, the redundant data caused by replicated path (which is reduced by path summary) may adversely affects the broadcast efficiency.

In short, the index scheme proposed in *node-level* scheme is either imprecise or incurs a larger size. Besides, they schedule the broadcast data in flat broadcast scheme and hardly response to the skewed data access, which is our concern in this paper. This is because the scheduling of nodes in this scheme is always in a DFS order to maintain the tree structure. Thus, scheduling skewed data may destroy the order and cannot support XPath query over it. For these reasons, our work is provided based on the techniques under *document-level* scheme, such as indexing technique and so on.

[4][5] treat the XML document as the unit of broadcast data. [6] first proposed the idea of splitting XML documents. A more efficient document-split algorithm is provided in [7]. The objective of these works is similar to that of our current work. But the split results of these algorithms still have redundancy of label paths. This is because the label path (also with the corresponding attributes and text values) from root to split node are duplicated, and the deeper the depth of split node, the more redundancy. Besides, [6][7] are used in on-demand environment, so queries must be known in advance. And the split results are difficult to response to the changes of query pattern. While our work focus on fragment document in *push-based* broadcast mode for skewed data access.

3 Document Fragment

3.1 Subtree-Level Scheme

In practice, users may be interested in parts of one XML document. A part of action data is shown in Figure 1. *auction_info* and *item_info* are two hot spots

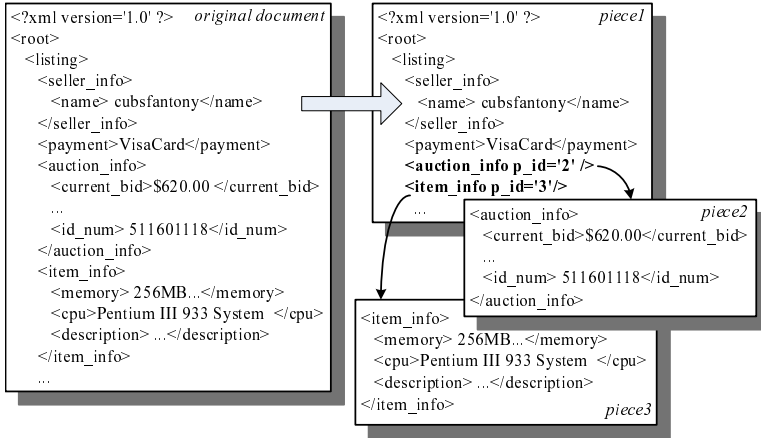


Fig. 1. An example of the subtree-level scheme for XML data organization

which attracted more attention. In *document-level* scheme, as one document is a unit data, people must retrieve all of the information in it. However, if the users care about *auction_info*, the other information (e.g. *item_info*, *seller_info*) is redundant for them. Downloading these useless information wastes the energy of mobile device.

Definition 1. *virtual node*: A virtual node is located in a piece and it is a substitute of the corresponding fragment node in original document. It contains the tag name and the piece id of fragment node.

Definition 2. *fragment node*: A fragment node is an element node of the original document and the content of its sub-tree compose a new piece.

Motivated by the inefficiency of previous works, we propose the *subtree-level* scheme. The idea of *subtree-level* scheme is easy to understand, as it is considered to be a tradeoff between *node-level* scheme and *document-level* scheme. The purpose of this scheme is to fragment the hot parts which accessed by more users from the original document, and the users are able to download the part of document they are interested in rather than the whole one.

Unlike the document-split algorithms in [6] [7], the label paths from root to fragment node are not replicated in each piece. The right part of Figure 1 shows the fragment results of *subtree-level* scheme. We use the *virtual nodes* to connect pieces. For example, the element node *auction_info* is fragment node in original document and *virtual node* in *piece1*. The process of retrieving data under *subtree-level* scheme contains these steps: Firstly, mobile users use the index to get the matched pieces'ids and the arrival time of them. Secondly, these pieces are downloaded and some of them reconstruct the original XML documents under the help of *virtual nodes*.

It is needed to be pointed out that the *virtual nodes* do not contain the addresses of the pieces they connected to. This is because the connected pieces

Table 1. Notations

Notation	Description
d_i	document (or piece) i
n_{ij}	node j of document d_i
N_i	The element node set of d_i
$ANC(n_{ij})$	the set of ancestors of node n_{ij}
$DEC(n_{ij})$	the set of descendants of node n_{ij}
$P(n_{ij})$	The access probability of n_{ij}
$P_{up}(n_{ij})$	Equals $\sum P(n_{ik}), n_{ik} \in ANC(n_{ij})$
$P_{down}(n_{ij})$	Equals $P(n_{ij}) + \sum P(n_{ik}), n_{ik} \in DEC(n_{ij})$
$Size(n_{ij})$	The size of sub-tree whose root is n_{ij}
$Cost(n_{ij})$	Presents an approximate cost for accessing n_{ij}
$Cost(d_i)$	Equals $\sum Cost(n_{ij}), n_{ij} \in N_i$
$Weight(n_{ij})$	The weight of n_{ij}
$add(n_{ij})$	The approximate size of the virtual node of n_{ij}

may have been broadcast before the current piece, as scheduling technique is adopted.

Table 1 shows the notations in the following of this paper. We use the cost of all the element nodes as a metric of TT . The cost of each node is the total size of original document multiplies the probability before fragmenting, while the cost after fragmenting is the size of pieces which contains part of its sub-tree multiplies its probability. We use the $add(n_{ij})$ to represent the size of *virtual node* of n_{ij} . And it approximately equals to the tag name and the attribute which mark the connected piece. In our fragment algorithms, the fragment nodes rather than the attribute nodes or the text nodes are selected from the element nodes in a XML document. And the access probability of an element node also contains those of the attribute nodes and the text nodes connected to it. In the following of this paper, a node also means the element node.

3.2 Horizontal Fragment Algorithm

Inspired by the query-grouping algorithm in [7], we adopt the idea in our horizontal fragment algorithm. The query-grouping algorithm has two phases. The first phase is to categorize the queries into different groups by the prefixes of them, and the second phase is to schedule XML data according to the results of the first phase. The prefixes are varied by the depth of them (specified as Dg). For example, in Figure 1, the query set is $\{\text{/root/listing/item_info}, \text{/root/listing}, \text{/root/listing/auction_info}\}$, and $Dg = 3$. Then, the grouping results are $\{\text{/root/listing/item_info}\}$, $\{\text{/root/listing/auction_info}\}$ and $\{\text{/root/listings}\}$. In the scheduling phase, each XML document is split by the groups, and the split XML documents contain the branches which match the corresponding queries in one group.

As our *subtree-level* scheme work in *push-based* broadcast mode, the queries are not known in advance. However, each element node has access probability, which means there may be some queries it matches. Without loss of generality, the depth of prefixes can be treated as the level of element node. The element nodes with level of Dg are fragment nodes. Dg in query-grouping algorithm is a variable parameter. However, in horizontal fragment algorithm, the optimal level for each document can be computed according access probability of each node. Let d'_i represents the result pieces of fragmenting document d_i . The optimal level is specified as *opt_level*, which fragmenting nodes located in and can get the minimum $Cost(d'_i)$.

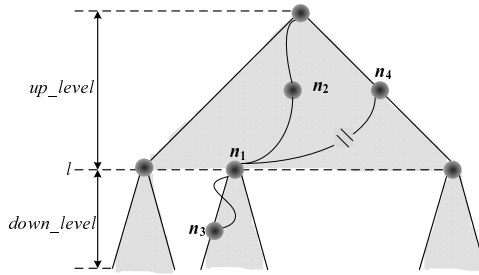


Fig. 2. Horizontal fragment algorithm

The horizontal fragment algorithm has two phases: the first phase is to select the *opt_level* for each document. The second phase is to fragment the nodes located in the optimal level. Selecting the *opt_level* is the core of horizontal fragment algorithm. In the following of this section, we propose a linear time algorithm for calculating *opt_level*.

The document tree is traversed in a breadth-first-search (BFS) way, and the $Cost(d'_i)$ is calculated when finishing the traversal of the element nodes of that level. We separate the element nodes into four sets for each level:

1. *fragment nodes set*: as node n_1 in Figure 2. The cost is changed to be $Cost(n_{ij}) = P(n_{ij}) \cdot Size(n_{ij})$, $n_{ij} \in \text{fragment nodes set}$.
2. *descendants set*: contains nodes which are the descendants of any node in *fragment nodes set*. For example, node n_3 is a descendant of n_1 . And the cost of the nodes in this set is $Cost(n_{ik}) = P(n_{ik}) \cdot Size(n_{ij})$, $n_{ij} \in \text{fragment nodes set}$.
3. *ancestors set*: contains nodes which are the ancestors of any node in *fragment nodes set*. As node n_2 in Figure 2 and the cost is changed. $Cost(n_{ik}) = P(n_{ik}) \cdot (Size\ of\ upper_level + \sum(Size(n_{ij}) + add(n_{ij})))$, $n_{ij} \in \text{fragment nodes set}$ and $n_{ij} \in DEC(n_{ik})$.
4. *no-relations set*: contains nodes which are neither not the ancestors nor the descendants of any node in *fragment nodes set*. For example, node n_4 in Figure 2. $Cost(n_{ik}) = P(n_{ik}) \cdot (Size\ of\ upper_level + \sum add(n_{ij}))$.

The size of *up_level* is $Size(n_{i0}) - \sum Size(n_{ij})$, n_{i0} is the root node in d_i , and $n_{ij} \in \textit{fragment nodes set}$. Thus the $Cost(d'_i)$ is the sum of cost of nodes in the four sets mentioned above, and we can conclude it as Formula 1 using the notations in Table 1:

$$\begin{aligned} Cost(d'_i) &= (P_{down}(n_{i0}) - \sum P_{down}(n_{ij})) \\ &\cdot (Size(n_{i0}) - \sum (Size(n_{ij}) - add(n_{ij}))) \\ &+ \sum ((P_{up}(n_{ij}) + P_{down}(n_{ij})) \cdot Size(n_{ij})) \\ &\quad n_{ij} \in \textit{fragment nodes set} \end{aligned} \quad (1)$$

We traverse the document tree in a BFS way by using a queue and then, put the nodes in the current level into *fragment nodes set*, and calculate the corresponding $Cost(d'_i)$ according to Formula 1. Finally, an *opt_level* with largest $Cost(d'_i)$ will be returned. As each node is visited once in the algorithm, the time complexity is $O(DN)$, where D is the number of XML documents and N is the average number of element nodes in one document. Before calculating the *opt_level*, the $P_{up}(n_{ij})$ and $P_{down}(n_{ij})$ for each node should have been computed. It is easy to calculate them by pre-order and post-order traversal of the document tree.

3.3 Threshold Fragment Algorithm

The horizontal fragment algorithm is a simple way for fragmenting document, as it chooses an optimal level, and nodes of the level construct the fragment nodes set of that document. In this section, we propose an optimized algorithm for choosing the fragment nodes, named threshold fragment algorithm. Firstly, we assign a weight value to each element node. The weight value of a node is considered as the $\Delta Cost = Cost(d_i) - Cost(d'_i)$ after fragmenting the node. Secondly, the node with largest weight value is recursively selected to generate a new piece, and the weight values of nodes in the same original piece as the fragment node may be updated at the end of each recursion.

There are two points should be noticed. The first one is that the node with largest weight value is select from the total element nodes from document set rather than on document. The second is that a threshold T should be given to controlling the number of recursions. The derivation of weight is shown as follows. We separate the element nodes into three sets for each node. For example, the node n_1 in Figure 2:

1. *descendants set*: contains nodes which are the descendants of the given node n_{ij} . For example, node n_3 is a descendant of n_1 . And the cost of the nodes in this set is changed. The reduction of the cost is to be $\Delta Cost(n_{ik}) = P(n_{ik}) \cdot (Size(n_{i0}) - Size(n_{ij}))$, $n_{ik} \in DEC(n_{ij})$.
2. *ancestors set*: contains nodes which are the ancestors of the given node n_{ij} . As node n_2 in Figure 2 and the cost is $\Delta Cost(n_{ik}) = -add(n_{ij})$.

3. *no-relations set*: contains nodes which are neither not the ancestors nor the descendants of the given node n_{ij} . For example, node n_4 in Figure 2. The reduction of the cost of this node is changed to be $\Delta Cost(n_{ik}) = P(n_{ik}) \cdot Size(n_{ij})$.

The weight value of one node is the reduction of cost after fragmenting the node. We can compute the weight value as Formula 2 according to the above analysis. using the notations in Table 1.

$$\begin{aligned} Weight(n_{ij}) = & (P_{down}(n_{i0}) - P_{down}(n_{ij}) - P_{up}(n_{ij})) \cdot Size(n_{ij}) \\ & + (Size(n_{i0}) - Size(n_{ij})) \cdot P_{down}(n_{ij}) \\ & - (P_{down}(n_{i0}) - P_{down}(n_{ij})) \cdot add(n_{ij}) \end{aligned} \quad (2)$$

As the threshold algorithm recursively select the node with the largest weight value from the document set, the data structure for the total data set is crucial. We assign a max-heap for maintaining the nodes with maximum weight values of each document. In this structure, the time complexity for one recursion is $O(N + \log D)$, D is the number of XML documents and N is the average number of element nodes in a XML document. $O(N)$ is for finding the node with maximum weight from the document and $O(\log D)$ is for updating the max-heap. The threshold fragment algorithm is shown as follows.

Algorithm 1. Threshold fragment algorithm

Require: XML document set D , the threshold T

Ensure: fragment result set R

- 1: initialize max-Heap H , insert the nodes with largest weight value of each document into H
 - 2: initialize $R \leftarrow \emptyset$, put each n_{i0} of document i into R
 - 3: **for** $t = 0$ to T **do**
 - 4: put the head node n_{ij} of H in R
 - 5: **for** each node n_{ik} in document i **do**
 - 6: update the weight value of n_{ik}
 - 7: **end for**
 - 8: add the node with largest weight value in document i to H
 - 9: adjust the H
 - 10: **end for**
-

3.4 Modification of the Two-Tier Index Scheme

The two-tier index [4] uses the DataGuides to extract the structure information of each XML document and the RoXSum to combine these DataGuides for eliminating the redundancy. The document id located in a node is referred as existing a label path from root to the node in the document. In this paper, the fragmented pieces' ids should be put in the index. A label path may be fragmented into different pieces by the fragment algorithms. For example, in

Figure 1, the label path `/root/listing/auction_info/current_bid` is fragmented into *piece1* and *piece2*. *Piece2'id* is put in node `current_bid` and *piece1'id* is put in node `listing`, which is the parent node of the virtual node. When the *auction_info* is needed, users retrieve the *piece2*. When the listing information is needed, users download the *piece1* and *piece2*. Finally, the restoration of part of XML can be accomplished by replacing the *virtual node* with the root node of the piece whose *id* is the attribute value of it.

4 Experiments

In this section, we study the performance of our two fragment algorithms compared with the original XML documents without fragmenting. As described in section2, the scheduling technique in *node-level* scheme is flat broadcast, which is not suitable for skewed data access. For *document-level* scheme, the broadcast modes are different. These works are proposed in *on-demand* broadcast, while our work focuses on *push-based* broadcast. The document splitting algorithms split XML documents according to the XPath queries pending on the server, while our fragment algorithms work based on the access probability of each node in XML document.

4.1 Experimental Setup

In XML data broadcast, it is not suitable to adopt the definition of access probability in traditional data broadcast. This is because the XML data are semi-structured while the data is flat in traditional data broadcast. We vary the types (by varying the variable *prob* and D_q) of XPath queries to change the distribution of access probability. In our simulation, the access probability of each node is statistical numbers from a large set of queries. The average *TT* of each query is the total size of pieces which contain the sub-trees of matched nodes. Though it is uncertain that the piece satisfies the XPath query completely, there must be an XPath query satisfied by it. Without loss of generality, these XPath queries are used as they retrieved the sub-tree of the matched nodes.

In our simulation, we value the *AT* and *TT* by number of bytes broadcast, the unit of them is KB in the results. The threshold *T* of threshold fragment algorithm is specified as the number of fragments of horizontal fragment algorithm. Similar to the existing work [4][5], the XML is generated by the IBM's Generator tool [11] and the XPath queries are generated by YFilter [12].

Table 2. Experimental setup

Variable	Description	Default value
D_q	minimum depth of queries	4
<i>prob</i>	The probability of wildcard * and double slash // in queries	0.1
N_d	the number of XML documents	1000

We use the QEM algorithm [13] to schedule the XML documents or XML pieces. Table 2 summarizes the system parameters.

4.2 Results and Discussion

Figure 3 shows the effects of D_q . TT and AT decrease with the increasing of D_q . This is because the bigger the minimum depth of queries, the smaller number of matched pieces. The TT of data processed by horizontal fragment algorithm (shorted as HF) and threshold fragment algorithm (shorted as TF) is less affected than that of original XML document (shorted as $no - F$). This results from the small TT . The TT of HF is about 15% of $no - F$ and that of TF is about 10% of $no - F$. The AT of both are smaller than that of $no - F$. There are two reasons. The first one is the AT is affected by the selectivity which is the degree of TT over the size of broadcast data [13]. Thus, the smaller selectivity, the smaller AT . The second one is the pieces whose access probabilities are equals to zero are not put in the broadcast channel. Thus, the final broadcast size after fragmenting is reduced about 3% of the size of original data.

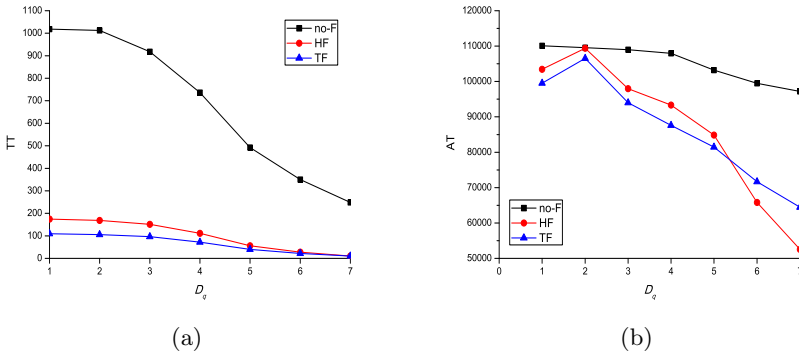


Fig. 3. Performance of fragment methods by varying D_q

Figure 4 depicts the results by varying $prob$. As $prob$ increases, TT and AT increase because more XML pieces are matched a XPath query with higher probability of $*$ and $//$. TT of HF and TF increase less than $no - F$ with the increasing $prob$, the reason is the same as in Fig. 3. And the AT of HF and TF are smaller than $no - F$. This is because of the large reduction of TT . When $prob = 0.0$, the TT of HF and TF are both smaller than 5% of $no - F$ and the AT of them are about 50% and 20% of $no - F$ respectively.

The performances of HF and TF by varying the number of original XML documents are shown in Fig 5. The TT and AT increase with the increasing of N_d . It is because the size of result data becomes larger when increasing the N_d . The TT and AT of HF and TF are both reduced a lot compared with $no - F$. It shows that our fragment algorithms are of good scalability.

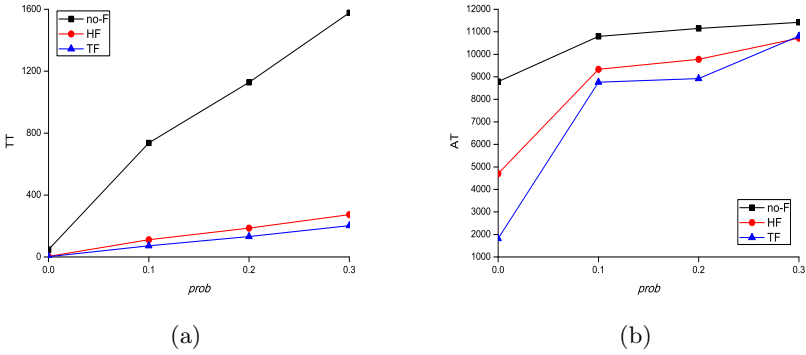


Fig. 4. Performance of fragment methods by varying $prob$

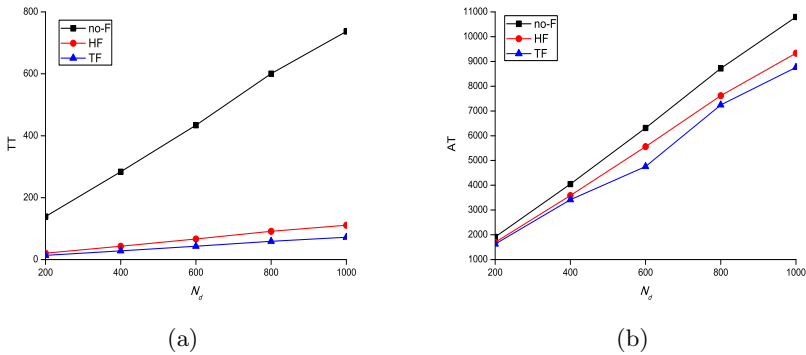


Fig. 5. Performance of fragment methods by varying N_d

5 Conclusions

In this paper, we proposed a *sub-tree* level scheme for skewed XML data access in *push-based* data broadcast and two efficient fragment algorithms working under this scheme. The threshold fragment algorithm adopted the idea of the query-grouping scheduling algorithm in *document-level* scheme of *on-demand* broadcast, and the performance of threshold fragment algorithm was better than the former one with a little higher time complexity. The experiments showed that our fragment algorithms improved the broadcast efficiency a lot.

Acknowledgment. This research is supported in part by the National Natural Science Foundation of China (NSFC) under grant 61073001. Weiwei Sun is the corresponding author.

References

1. Xu, J., Lee, D.L., Hu, Q., Lee, W.-C.: Data Broadcast. In: Handbook of Wireless Networks and Mobile Computing, John Wiley & Sons, Chichester (2002)
2. Park, S., Choi, J., Lee, S.: An effective, efficient XML data broadcasting method in a mobile wireless network. In: Bressan, S., Küng, J., Wagner, R. (eds.) DEXA 2006. LNCS, vol. 4080, pp. 358–367. Springer, Heidelberg (2006)
3. Chung, Y., Lee, J.: An indexing method for wireless broadcast XML data. *Information Sciences* 177(9), 1931–1953 (2007)
4. Sun, W., Yu, P., Qin, Y., Zhang, Z., Zheng, B.: Two-Tier Air Indexing for On-Demand XML Data Broadcast. In: ICDCS (2009)
5. Qin, Y., Sun, W., Zhang, Z., Yu, P., He, Z., Chen, W.: A Novel Air Index Scheme for Twig Queries in On-Demand XML Data Broadcast. In: Bhowmick, S.S., Küng, J., Wagner, R. (eds.) DEXA 2009. LNCS, vol. 5690, pp. 412–426. Springer, Heidelberg (2009)
6. Qin, Y., Sun, W., Zhang, Z., Yu, P.: An Efficient Document-Split Algorithm for On-Demand XML Data Broadcast Scheduling. In: CCWMSN (2007)
7. Qin, Y., Sun, W., Zhang, Z., Yu, P., He, Z.: Query-Grouping Based Scheduling Algorithm for On-Demand XML Data Broadcast. In: WiCOM (2008)
8. Wong, J.W.: Broadcast Delivery. *Proceedings of the IEEE* 76(12), 1566–1577 (1988)
9. Hameed, S., Vaidya, N.H.: Efficient Algorithms for Scheduling Data Broadcast. *ACM/Baltzer Journal of Wireless Networks* 5(3), 183–193 (1999)
10. Acharya, S., Alonso, R., Franklin, M., Zdonik, S.: Broadcast Disks: Data Management for Asymmetric Communications Environments. In: SIGMOD (1995)
11. Diaz, A., Lovell, D.: XML Generator, <http://www.alphaworks.ibm.com/tech/xmlgenerator>
12. Diao, Y., Altinel, M., Franklin, M., Zhang, H., Fischer, P.: Path sharing and predicate evaluation for high-performance XML filtering. *ACM Transactions on Database Systems* 28(4), 467–516 (2003)
13. Chung, Y.D., Kim, M.H.: QEM: A Scheduling Method for Wireless Broadcast. In: DASFAA (1999)

Evaluating Probabilistic Spatial-Range Closest Pairs Queries over Uncertain Objects

Mo Chen, Zixi Jia, Yu Gu, Ge Yu, and Chuanwen Li

Northeastern University, China

{chenmo, jiazixi, guyu, yuge, lichuanwen}@ise.neu.edu.cn

Abstract. In emerging applications such as location-based service (LBS), the values of spatial database items are naturally uncertain. This paper focuses on the problem of finding probabilistic closest pairs between two uncertain spatial datasets in a given range, namely, *probabilistic spatial-range closest pair* (PSRCP) query. In particular, given two uncertain spatial datasets which contain uncertain objects modeled by a set of sampled instances, a PSRCP query retrieves the pairs that satisfy the query with probabilities higher than a given threshold. Several pruning strategies are proposed to filter the objects that cannot constitute an answer, which can significantly improve the query performance. Extensive experiments are performed to examine the effectiveness of our methods.

1 Introduction

Uncertain data have common appearance in spatial databases since a number of indirect location collection methodologies have been proposed [1][2]. For example, consider a collection of moving objects, whose locations are tracked by Global-Positioning System (GPS). Due to GPS errors and transmission delays, the exact locations of these objects are difficult to be collected. As a result, the set of possible locations of an object is usually described by its appearance probabilities, which are derived from discrete instances or a *probability density function* (PDF).

Closest pair (CP) query is a combination of join and nearest neighbor query, which is first described in [3]. By definition, given two object sets A and B , their CP join is a one to one assignment of objects from the two sets, such that the distance of the result pair is the smallest amongst all possible object pairs. CP queries are widely used in emerging applications where spatial objects exists with uncertainty, thus we introduce *top- K probabilistic closest pairs* (Top K -PCP) query in [4], where efficient processing methods are proposed to retrieve the object pairs with top- K maximal probabilities of being the closest pair.

In this paper, we study *probabilistic spatial-range closest pairs* (PSRCP) query for spatial databases with uncertain data. In other words, PSRCP query deals with range CP problems over uncertain objects, which is considered as an extension to Top K -PCP query. This type of query is important since users prefer to search in a portion of the space in many realistic cases. For instances, query “*find a pair of taxi and pedestrian that has the smallest distance in Madison*

street” is more useful than “find a pair of taxi and pedestrian that has the smallest distance in U.S.”. Figure 1 exemplifies PSRCP query over uncertain objects that are modeled by sampled instances, in which the dark region denotes query range R . Specifically, each uncertain object (e.g., object A) contains a set of uncertain instances. Range R intersects with object A , B and C . Therefore, the answer sets of PSRCP does not contain object D definitely. Therefore, we need to calculate the range CP probabilities of pairs (A, B) , (A, C) and (B, C) .

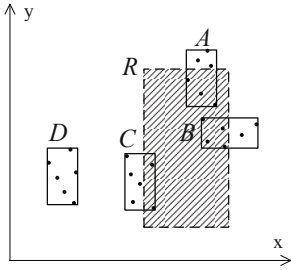


Fig. 1. A PSRCP query on uncertain objects

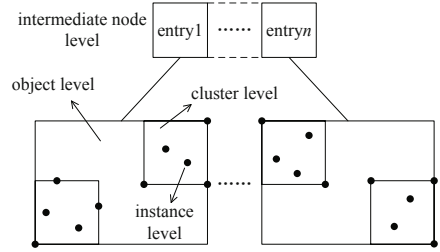


Fig. 2. Index structure

To the best of our knowledge, none of the existing work has addressed the probabilistic range CP problem. In this paper, we give detailed analysis of appearance probabilities based on uncertain instances, which can be straightforwardly extended to PDF. Our contributions also include:

- We formally define PSRCP query over uncertain objects.
- Several prune strategies are proposed to improve searching performance.
- An efficient algorithm is developed to support the evaluation of PSRCP query.
- An extensive experimental study is conducted to show the effectiveness of our method.

The rest of the paper is organized as follows. Section 2 briefly overviews related works. Section 3 describes problem definitions. The details of the query processing are presented in Section 4. Section 5 shows the experimental results of the proposed method. The paper is concluded with a discussion about future work in Section 6.

2 Related Work

Section 2.1 surveys the studies on query processing in uncertain databases. Section 2.2 presents previous methods of answering closest pair queries on spatial data.

2.1 Probabilistic Queries over Uncertain Data

Probabilistic queries have gained much attention due to the emerging applications that involve uncertainty. A survey of the research area concerning uncertainty and incomplete information in databases is given in [5]. Uncertain tuple and uncertain attribute are considered as two types of uncertain data [6]. Specifically, uncertain tuples are usually derived from the probabilistic relational databases [7][8]. On the other hand, the attribute uncertainty is introduced as each uncertain object is modeled by a distribution within an uncertain region. The appearance probabilities can be described as either discrete case (i.e., uncertain instances) [9][10] or continuous case (i.e., PDF) [11][12]. Many techniques have been designed for uncertain databases, with queries on attribute uncertainty such as *nearest neighbor query* [9], *range query* [11], *reverse nearest query* [13], *top-k query* [14], *closest pairs query* [4] and so on. A tree index, U-tree, is constructed for probabilistic range queries [15], which is one of the most popular index structure for uncertain data with arbitrary PDFs. However, to the best of our knowledge, so far, no existing work has studied spatial-range closest pair query in the context of the uncertain databases, which assumes that spatial objects have uncertain attribute modeled by discrete representations.

2.2 Closest Pair Queries over Spatial Objects

Closest pairs queries have been extended to spatial databases in these few years. Distance-join algorithms, which are based on a priority queue, is first proposed in [16]. But the algorithm takes up a lot of main memory since the pair items store both node pairs and object pairs in priority queue. Therefore, Corral et al. [3] proposed several non-incremental algorithms that significantly reduce the size of the queue. Afterwards, Corral et al. extended and enhanced the work with respect to the design of branch-and-bound algorithms in a non-incremental way [17]. But the above methods cannot work well in cases that the two data sets 'overlap', thus, Yang and Lin [18] proposed a new index structure by pre-computing and storing the nearest neighbor information in the node entry of R-tree. Leong et al. [19] identified the ECP (*exclusive closest pairs*) problem, which is a spatial assignment problem. We first studied probabilistic closest pairs queries over spatial objects in [4], in which each spatial object is modeled by a set of sample points. An efficient method is proposed, which retrieves the pairs with top-K maximal probabilities of being the closest pair. In this paper, we extend the work by considering the problem of searching for probable closest pairs within a spatial region.

3 Problem Definition

Let $U = \{U_1, U_2, \dots, U_n\}$ and $V = \{V_1, V_2, \dots, V_m\}$ be two sets of uncertain objects. Following the previous work [4][9][18][20], we describe the uncertain objects by discrete PDF model. In particular, each uncertain object $O_i (O_i \in U \cup V)$ is associated with z sampled instances, i.e., $O_i = \{o_{i,1}, o_{i,2}, \dots, o_{i,z}\}$, O_i

may appear at any of its instances with a probability. For simplicity, we assume that the probability is identical, i.e., the PDF-value of each instance $P(O_{i,j}) = 1/z$ ($j = 1, 2, , z$). We apply the integration method proposed in [21] to reduce the computation cost, which is implemented by clustering the instances of each object into several clusters individually. Therefore, the uncertain object in this paper is approximated by the clusters of sampled instances instead of distributed instances. The index organization of the uncertain objects is shown in Figure 2, where each cluster is bounded by a *minimum bounding rectangle*(MBR).

In certain databases containing precise spatial objects, range closest pair query (R-CP) [22] discovers the closest pair of spatial objects inside a spatial range R . In uncertain databases, the attribute data of each object are "probabilistic", we then define the probabilistic versions of R-CP query as follows.

Definition 1. A Probabilistic Spatial-Range Closest Pair (PSRCP) Query takes as input a spatial region R and returns all $((U_i, V_j), P_{CP_ij})$ pairs, such that $U_i \in U, V_j \in V$ and $P_{CP_ij} > 0$, where P_{CP_ij} is the probability that (U_i, V_j) is the closest pair inside R .

We use $F_{SR}(u_{i,l}, v_{j,f})$ to denote the probability of instance pair $(u_{i,l}, v_{j,f})$ to be the closest pair. Then the value of P_{CP_ij} is computed by Equation 1, which can be easily verified.

$$P_{CP_ij} = \frac{\sum_{u_{i,l} \in U_i, u_{i,l} \mapsto R} \sum_{v_{j,f} \in V_j, v_{j,f} \mapsto R} F_{SR}(u_{i,l}, v_{j,f})}{z^2} \tag{1}$$

where $u_{i,l} \mapsto R$ denotes uncertain instance $u_{i,l}$ is inside R . Similar to [4], $F_{SR}(u_{i,l}, v_{j,f})$ is given as follows:

$$F_{SR}(u_{i,l}, v_{j,f}) = \prod_{\substack{\forall (U_{i'}, V_{j'}) \in U \times V \\ \wedge (U_i, V_j) \neq (U_{i'}, V_{j'})}} 1 - \frac{|\{(u_{i'}, v_{j'}) | D(u_{i'}, v_{j'}) < D(u_{i,l}, v_{j,f})\}|}{z^2} \tag{2}$$

where $D(u_{i,l}, v_{j,f})$ denotes the Euclidean distance between instances $u_{i,l}$ and $v_{j,f}$.

However, since the query qualification (i.e., $P_{CP_ij} > 0$) is relatively loose, the result of PSRCP query may be arbitrarily large if there are numerous uncertain objects intersecting with R . Therefore, we define a practical version of PSRCP query with controlled output by thresholding the results of low probability to occur:

Definition 2. Let $((U_i, V_j), P_{CP_ij} > 0)$ be an output item of PSRCP query. The thresholding version of PSRCP takes threshold P_t as an additional input, $0 < P_t \leq 1$, and returns the results for which $P_t \leq P_{CP_ij}$.

In the rest of this paper, we focus on the evaluation of the thresholding version of PSRCP, which is more preferred by many real applications. We refer to this version as PSRCP query for simplicity.

4 Evaluation of PSRCP Query

In this section, we firstly present some pruning strategies which filter out the objects that are definitely not PSRCPs. Then the PSRCP query processing algorithm is illustrated based on the pruning strategies.

4.1 Pruning Strategies

Given two sets of uncertain objects $U = \{U_1, U_2, \dots, U_n\}$ and $V = \{V_1, V_2, \dots, V_m\}$, let $P_{RA}(U_i)(U_i \in U)$ and $P_{RA}(V_j)(V_j \in V)$ denote the probabilities that U_i and V_j are inside R , respectively. The rationales behind the pruning strategies are stated in the following lemmas.

Lemma 1. Given PSRCP query threshold P_t , $0 < P_t \leq 1$, if $P_{RA}(U_i) < P_t$, $P_{RA}(V_j) < P_t$, then (U_i, V_j) will not appear in the answer data set.

Proof. If $P_{RA}(U_i) < P_t$ and $P_{RA}(V_j) < P_t$ hold, the probability that (U_i, V_j) is the closest pair within R is smaller than P_t according to $0 < P_t \leq 1$ and

$$P_{CP_ij} = P_{RA}(U_i)P_{RA}(V_j)P'_{CP}(U_i, V_j) \quad (3)$$

i.e., we have $P_{CP_ij} < P_t$ even in the special case that $P'_{CP}(U_i, V_j) = 1$, where $P'_{CP}(U_i, V_j)$ denotes the probability that (U_i, V_j) is the closest pair in the whole space. Therefore, no answer set contains the pair (U_i, V_j) .

Lemma 2. Given PSRCP query threshold P_t , $0 < P_t \leq 1$, if $P_{RA}(U_i)P_{RA}(V_j) < P_t$, then (U_i, V_j) will not appear in the answer data set.

Proof. Similar to lemma 1, according to Equation 3, we have $P_{CP_ij} < P_t$ since $P_{RA}(U_i)P_{RA}(V_j) < P_t$, which implies that (U_i, V_j) will not appear in the answer data set.

U-tree^[15] is a popular index structure for range queries over uncertain data, which is employed in our pruning procedure. According to the structure of U-tree, the above lemmas not only provide rationales for the pruning of uncertain objects, but also support the pruning of intermediate nodes. Since U-tree is designed for uncertain data with arbitrary PDFs, the location of uncertain object in our paper can be considered uniformed distributed according to the above uncertain data model. In particular, some pre-computed information (e.g., probabilistically constrained regions, PCRs) is maintained at all levels of the tree, which is utilized to conduct pruning in the index level. Given two indexing U-trees of set U and V , T_U and T_V , the pruning strategies are presented as follows:

Pruning Strategy 1. For any two nodes E_U and E_V in T_U and T_V , as long as they simultaneously hold that $P_{RA}(E_U) < P_t$ and $P_{RA}(E_V) < P_t$, the subtrees of E_U and E_V can be safely pruned, where $P_{RA}(E_U)$ and $P_{RA}(E_V)$ are the respective appearance probabilities that the MBRs of E_U and E_V lie in R .

Pruning Strategy 2. For any two nodes E_U and E_V in T_U and T_V , as long as they hold that $P_{RA}(U_i)P_{RA}(V_j) < P_t$, the subtrees of E_U and E_V can be safely pruned.

P.S.1 (short for Pruning Strategy 1) can be directly conducted by PCRs. Note that P.S.2 (short for Pruning Strategy 2) requires extra probability computations, it is applied only for nodes that pass P.S.1.

Lemma 3. Let k be the number of clusters indexed by an uncertain object. We assume that $(c_{U_i,a}, c_{V_j,b})$ is the closest pair within the whole space, where $c_{U_i,a}$ and $c_{V_j,b}$ are the a th and b th cluster of U_i and V_j respectively. If $P_{RA}(U_i)P_{RA}(V_j) < \frac{P_t}{k} \text{Pr}_c(c_{U_i,a}, c_{V_j,b})$, then (U_i, V_j) will not appear in the answer data set, where $\text{Pr}_c(c_{U_i,a}, c_{V_j,b})$ is the likelihood that $(c_{U_i,a}, c_{V_j,b})$ contributes to the result probability.

Proof. Since $(c_{U_i,a}, c_{V_j,b})$ is the closest pair within R , we have $\text{Pr}_c(c_{U_i,a'}, c_{V_j,b'}) < \text{Pr}_c(c_{U_i,a}, c_{V_j,b})$ for $\forall (c_{U_i,a'}, c_{V_j,b'}) \neq (c_{U_i,a}, c_{V_j,b})$. Therefore, it holds that

$$\sum_{\forall c_{U_i,a'} \subset U_i, c_{V_j,b'} \subset V_j} \text{Pr}_c(c_{U_i,a'}, c_{V_j,b'}) < k \text{Pr}_c(c_{U_i,a}, c_{V_j,b}) \quad (4)$$

As Pr_c denotes the contribution probability, we have $P'_{CP}(U_i, V_j) < k \text{Pr}_c(c_{U_i,a}, c_{V_j,b})$, which can be substituted into Equation 3, resulting in

$$P_{CP_{ij}} < k P_{RA}(U_i) P_{RA}(V_j) \text{Pr}_c(c_{U_i,a}, c_{V_j,b}) \quad (5)$$

Since inequality $P_{RA}(U_i)P_{RA}(V_j) < \frac{P_t}{k} \text{Pr}_c(c_{U_i,a}, c_{V_j,b})$ holds as a precondition, we obtain $P_{CP_{ij}} < P_t$, which indicates that (U_i, V_j) is not contained in the answer data set.

Lemma 4. Let z be the number of samples of each uncertain object. We assume that $(s_{U_i,a}, s_{V_j,b})$ is the closest pair within the whole space, where $s_{U_i,a}$ and $s_{V_j,b}$ are the a th and the b th sample of U_i and V_j respectively. If $P_{RA}(U_i)P_{RA}(V_j) < \frac{P_t}{z} \text{Pr}_s(s_{U_i,a}, s_{V_j,b})$, then (U_i, V_j) will not appear in the answer data set, where $\text{Pr}_s(s_{U_i,a}, s_{V_j,b})$ is the likelihood that $(s_{U_i,a}, s_{V_j,b})$ contributes to the result probability.

Proof. Similar to the proof of lemma 3, which is omitted here due to space limitation.

Based on the above lemmas, we employ the following pruning strategies when we access a cluster pair or a sampled instance pair, which are called P.S.3 and P.S.4 for short respectively.

Pruning Strategy 3. In case a cluster pair $(c_{U_i,a}, c_{V_j,b})$ has the smallest distance within R , and $P_{RA}(U_i)P_{RA}(V_j) < \frac{P_t}{k} \text{Pr}_c(c_{U_i,a}, c_{V_j,b})$, $c_{U_i,a} \subset U_i$, $c_{V_j,b} \subset V_j$, then the object pair (U_i, V_j) can be pruned.

Pruning Strategy 4. In case a sample pair $(s_{U_i,a}, s_{V_j,b})$ has the smallest distance within R , and $P_{RA}(U_i)P_{RA}(V_j) < \frac{P_t}{z} \text{Pr}_s(s_{U_i,a}, s_{V_j,b})$, $s_{U_i,a} \in U_i$, $s_{V_j,b} \in V_j$, then the object pair (U_i, V_j) can be pruned.

Table 1. Meanings of notations

Symbols	Descriptions
$E_{U,i}, E_{V,j}$	the i th/ j th node in T_U/T_V
$E_{U,i.chi}, E_{V,j.chi}$	child node of $E_{U,i}/E_{V,j}$
num_{gh}	total number of accessed instance pairs of (U_g, V_h)
$Min_MinD(.)$	minimal minimum MBR distance of two nodes
$Max_MaxD(.)$	maximal maximum MBR distance of two nodes
$Min_MaxD(.)$	minimal maximum MBR distance of two nodes
$Num(E_{U,i}), Num(E_{V,j})$	the number of the instances contained by $E_{U,i}/E_{V,j}$

4.2 PSRCP Query Algorithm

The process of PSRCP querying is detailed in Algorithm 1 and Table 1 summarizes the notations used in the algorithm. The algorithm, which is considered as an extended version of TopK-PCP query algorithm [4], takes the pruning strategies presented above into consideration. Similarly, a priority queue \mathcal{H} is maintained, which contains the entry of form $((E_{U,i}, E_{V,j}), Min_MinD(E_{U,i}, E_{V,j}))$ (line 1). A list \mathcal{L} is used to maintain the information of the probabilistic closest pairs found up to now (line 2). We employ a set Res to store the query results (line 3). The computation of probabilistic closest pairs is processed in a similar manner with TopK-PCP. However, it is important to consider "range" requirements, based on which we extend the algorithm. In particular, each time the first entry Cur is de-heaped from \mathcal{H} (line 7), we check the type of the two elements in Cur as follows.

If the type of the elements is intermediate node, we use P.S.1 and P.S.2 to prune non-qualifying nodes (line 9). We update \mathcal{W} according to $\min(Min_MaxD(E_{U,i.chi}, E_{V,j.chi}))$ of the remaining nodes (lines 10 and 11). P.S.1 and P.S.2 are applied again to filter the child nodes $(E_{U,i.chi}, E_{V,j.chi})$, which are inserted into \mathcal{H} according to $Min_MinD(E_{U,i.chi}, E_{V,j.chi})$ (lines 12-15).

If we reach the object level, P.S.1 and P.S.2 are used to prune non-qualifying objects (line 17). Then the next entry is de-heaped from \mathcal{H} , and we apply P.S.1 and P.S.2 to the entry to ensure its qualification (lines 19 and 20). If the entry is non-qualifying, the next entry is popped until \mathcal{H} is empty (lines 21 and 22). Then, we check whether the current object pair is the closest pair within R (lines 23-29). Specifically, if the current pair is the closest, we need to calculate the likelihood that it is the closest pair and update \mathcal{L} by setting $P_{CP_gh} = Pr_o(E_{U,i}, E_{V,j})$ and $num_{gh} = z^2$, whose details are presented in [4]. Then $((E_{U,i.chi}, E_{V,j.chi}), Min_MinD(E_{U,i.chi}, E_{V,j.chi}))$ is inserted into Res if $P_{CP_gh} \geq P_t$. Otherwise, the object pair is refined by repeating the steps in lines 12 to 15. If \mathcal{H} is empty after the popping of current pair, we check whether it is the result pair (lines 30 and 31).

If the cluster level is reached, we prune non-qualifying cluster pair by P.S.1 and P.S.2 (lines 32 and 33). If \mathcal{H} is empty after de-heaping Cur , the qualification of Cur is checked (line 40). Else, the next entry is checked in a similar manner (lines 34-38). After both of the two popped entries are validated, the likelihood

Algorithm 1. PSRCP Query Processing**Input:** two uncertain object sets U and V **Output:** Result set Res

```

1: initialize  $\mathcal{H}$  accepting entries  $((E_{U,i}, E_{V,j}), Min\_MinD(E_{U,i}, E_{V,j}))$ 
2: initialize  $\mathcal{L}$  accepting entries in the form  $(U_i, V_j, num_{ij})$ 
3:  $Res \leftarrow \emptyset$ 
4: start from the roots of  $T_U$  and  $T_V$ ,  $\mathcal{W} \leftarrow \infty$ 
5: insert  $((T_U.root, T_V.root), Min\_MinD(T_U.root, T_V.root))$  into  $\mathcal{H}$ 
6: while  $\mathcal{H}$  is not empty do
7:   entry  $Cur \leftarrow$  de-heap  $\mathcal{H}$ 
8:   if  $Cur.(E_{U,i}, E_{V,j})$  is an intermediate node pair then
9:     perform P.S.1 and P.S.2
10:    if  $\min_{\substack{\forall E_{U,i,chi} \\ \forall E_{V,j,chi}}} (Min\_MaxD(E_{U,i,chi}, E_{V,j,chi})) < \mathcal{W}$  then
11:      update  $\mathcal{W}$ 
12:      for each child pair  $(E_{U,i,chi}, E_{V,j,chi})$  do
13:        perform P.S.1 and P.S.2
14:        if  $Min\_MinD(E_{U,i,chi}, E_{V,j,chi}) \leq \mathcal{W}$  then
15:          insert  $((E_{U,i,chi}, E_{V,j,chi}), Min\_MinD(E_{U,i,chi}, E_{V,j,chi}))$  into  $\mathcal{H}$ 
16:    if  $Cur.(E_{U,i}, E_{V,j})$  is an object pair then
17:      perform P.S.1 and P.S.2
18:      if  $\mathcal{H}$  is not empty then
19:        the next entry  $Nex.((E_{U,i'}, E_{V,j'}), Min\_MinD(E_{U,i'}, E_{V,j'})) \leftarrow$  de-heap  $\mathcal{H}$ 
20:        perform P.S.1 and P.S.2
21:        if  $Nex$  is pruned then
22:          goto line 18
23:        if  $Min\_MinD(E_{U,i'}, E_{V,j'}) \geq Max\_MaxD(E_{U,i}, E_{V,j})$  then
24:          calculate  $Pr_{\mathcal{O}}(E_{U,i}, E_{V,j})$ 
25:          update  $\mathcal{L}$ 
26:          if  $\exists E_{U,i} \in U_g \wedge E_{V,j} \in V_h$  with  $P_{CP\_gh} \geq P_t$  then
27:            insert  $((U_g, V_h), P_{CP\_gh})$  into  $Res$ 
28:        else
29:          repeat lines 10-15
30:      else
31:        goto line 24
32:    if  $Cur.(E_{U,i}, E_{V,j})$  is a cluster pair then
33:      perform P.S.1 and P.S.2
34:      if  $\mathcal{H}$  is not empty then
35:        the next entry  $Nex.((E_{U,i'}, E_{V,j'}), Min\_MinD(E_{U,i'}, E_{V,j'})) \leftarrow$  de-heap  $\mathcal{H}$ 
36:        perform P.S.1 and P.S.2
37:        if  $Nex$  is pruned then
38:          goto line 32
39:      else
40:        goto line 42
41:      if  $Min\_MinD(E_{U,i'}, E_{V,j'}) \geq Max\_MaxD(E_{U,i}, E_{V,j})$  then
42:        calculate  $Pr_{\mathcal{C}}(E_{U,i}, E_{V,j})$  and perform P.S.3
43:        update  $\mathcal{L}$ 
44:        if  $\exists E_{U,i} \in U_g \wedge E_{V,j} \in V_h \wedge num_{gh} = z^2$  with  $P_{CP\_gh} \geq P_t$  then
45:          insert  $((U_g, V_h), P_{CP\_gh})$  into  $Res$ 
46:      else
47:        repeat lines 12-15
48:    if  $Cur.(E_{U,i}, E_{V,j})$  is a sample pair then
49:      if  $E_{U,i} \mapsto R$  and  $E_{V,j} \mapsto R$  then
50:        calculate  $Pr_{\mathcal{S}}(E_{U,i}, E_{V,j})$  and perform P.S.4
51:        update  $\mathcal{L}$ 
52:        if  $\exists E_{U,i} \in U_g \wedge E_{V,j} \in V_h \wedge num_{gh} = z^2$  with  $P_{CP\_gh} \geq P_t$  then
53:          insert  $((U_g, V_h), P_{CP\_gh})$  into  $Res$ 
54: return  $Res$ 

```

$Pr_{\mathcal{L}}$ is calculated and we perform P.S.3. Besides, \mathcal{L} is updated, i.e., set $P_{CP_gh} = Pr_{\mathcal{L}}(E_{U,i}, E_{V,j}) + P_{CP_gh}$ and $num_{gh} = num_{gh} + Num(E_{U,i})Num(E_{V,j})$ (lines 41-43). If all the instances of the two objects have been accessed, we add the result to Res according to $P_{CP_gh} \geq P_t$ (lines 44 and 45). Otherwise, the repeating procedure is implemented (line 47).

If the instance level is reached, we prune the sample pairs which are outside R (lines 48 and 49). For the validated sample pair, $Pr_{\mathcal{S}}$ is calculated and P.S.4 is performed (line 50). The updating is implemented by setting $P_{CP_gh} = Pr_{\mathcal{S}}(E_{U,i}, E_{V,j}) + P_{CP_gh}$ and $num_{gh} = num_{gh} + 1$ (line 51). After the updating, we need to make a decision on adding pair $((U_g, V_h), P_{CP_gh})$ to Res or not according to num_{gh} and P_{CP_gh} (lines 52 and 53).

The algorithm terminates if \mathcal{H} is empty. The returned pairs in Res are the result pairs with the probabilities not smaller than P_t (line 54).

5 Experimental Evaluation

All our experiments are run on a 1.86 GHz Intel Core 2 6300 CPU and 8 GB RAM. We use two realistic data sets of geographical objects¹, namely GU (utilities), GR (roads), GL (railroad lines) and GH (hypsography data) with respective cardinalities 17K, 30K, 36K and 77K. Given a value z , these datasets are transformed into uncertain datasets. For example, we transform GU into uncertain databases GU_u as follows: each MBR of point p in GU is regarded as an uncertain region where z instances of p appear in. The instances are sampled randomly in the region. The experiments are conducted under four different combinations, namely UR, RL, UH and RH, representing $(U, V) = (GU_u, GR_u)$, (GR_u, GL_u) , (GU_u, GH_u) and (GR_u, GH_u) . Both of the two dimensions in these uncertain datasets are normalized to domain $[0,1]$.

In the following experiments, we perform a comparative study on our algorithm PSRCP and a naive method, NSRCP, which is a modified version of PSRCP based on abandoning the pruning strategies in section 4. Since the generation of the instances is randomly, each point in the following graphs is an average of the results for 100 queries.

Effect of sample rate. In this set of experiments, we examine the query performance by varying the number of sampled instances per object, which is considered as sample rate. Our algorithm is proceeded as a function of sample rate varies from 5 to 20. Figure 3 illustrates the results of the experiment on the four datasets. We use 0.5 and 10% as a default value for P_t and query range, respectively. As expected, PSRCP significantly outperforms NSRCP for all datasets, which verifies that the pruning strategies in PSRCP leads an effective reduction of computation cost. Besides, computation cost increases as the sample size gets larger, which can be easily understood.

Effect of threshold. In Figure 4, we present how the threshold P_t affects the performance of the two algorithms. In particularly, both of the two algorithms

¹ <http://www.rtreeportal.org/>

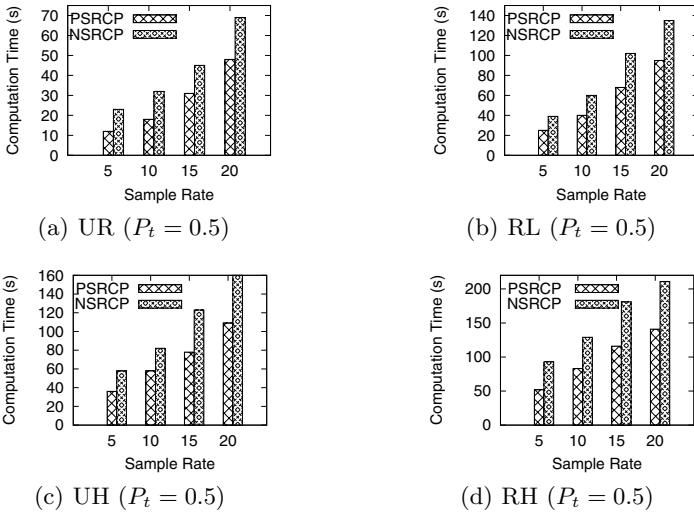


Fig. 3. Performance vs. Sample Rate

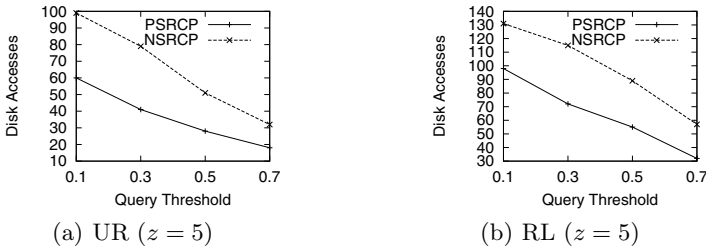


Fig. 4. Performance vs. Threshold

require less disk accesses when P_t increases, which is due to the number of candidate pairs decreases sharply, e.g., from 41 for $P_t = 0.3$ to 18 for $P_t = 0.7$ (UR-PSRCP). Therefore, a well-chosen threshold can provide better performance.

Effect of query range. We compare the algorithms when the queries are issued with different range from 10% to 40% of the space. As shown in Figure 5, the computation cost increases for PSRCP and NSRCP when the query range increases. This is because more objects will intersect the query range as the range ratio increases. We also observe that PSRCP algorithm has better performance.

Effect of buffer size. In the following, we investigate I/O activities of the two algorithms with different buffer sizes in Figure 6. We use the buffer sizes of 16, 64, 256, 1024 pages in this set of experiments. It is observed that the performance is improved as long as the buffer size grows. However, both of the two algorithms are not so sensitive to buffer size. For instance, as the buffer size increases from

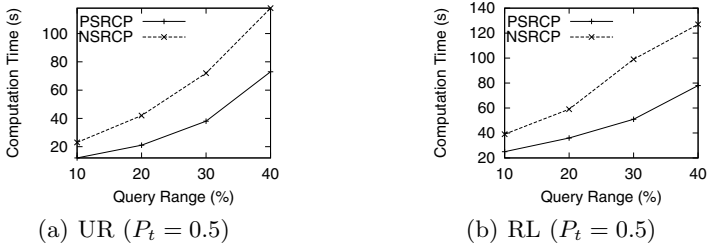


Fig. 5. Performance vs. Range

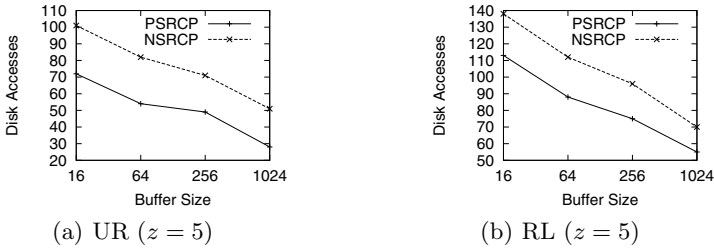


Fig. 6. Performance vs. Buffer Size

64 to 256, only 27.8% improvement is obtained (UR-PSRCP). This is because that each of the algorithms processes the query by following a best-first traversal pattern using a priority queue.

6 Conclusion

Closest pairs queries over spatial objects have attracted a lot of research attention. In this paper, we proposed an efficient approach to compute spatial-range closest pairs query over uncertain objects, which, to the best of our knowledge, no other work has studied before. We introduced several pruning strategies with considerations of range constraints. As shown by our experimental results, many unqualified objects can be pruned with the pruning techniques. In the future, we will study how these techniques can be extended to support other queries, e.g., ECP query.

Acknowledgment. This research was supported by NSFC No.60933001 and No.61003058, Research and Innovation Fund for Young Teachers N090304003.

References

1. de Almeida, V.T., Güting, R.H.: Supporting uncertainty in moving objects in network databases. In: Proceedings of the 13th Annual ACM International Workshop on Geographic Information Systems, pp. 31–40 (2005)
2. Pfoser, D., Jensen, C.: Capturing the uncertainty of moving-object representations. In: Güting, R.H., Papadias, D., Lochovsky, F.H. (eds.) SSD 1999. LNCS, vol. 1651, pp. 111–131. Springer, Heidelberg (1999)
3. Corral, A., Manolopoulos, Y., Theodoridis, Y., Vassilakopoulos, M.: Closest pair queries in spatial databases. In: SIGMOD, pp. 189–200 (2000)

4. Chen, M., Jia, Z., Gu, Y., Yu, G.: Top-k probabilistic closest pairs query in uncertain spatial databases. In: Du, X., Fan, W., Wang, J., Peng, Z., Sharaf, M.A. (eds.) APWeb 2011. LNCS, vol. 6612, pp. 53–64. Springer, Heidelberg (2011)
5. Aggarwal, C.C., Yu, P.S.: A survey of uncertain data algorithms and applications. *IEEE Trans. Knowl. Data Eng. (TKDE)* 21(5), 609–623 (2009)
6. Singh, S., Mayfield, C., Shah, R., Prabhakar, S., Hambrusch, S.E., Neville, J., Cheng, R.: Database support for probabilistic attributes and tuples. In: ICDE, pp. 1053–1061 (2008)
7. Bosc, P., Pivert, O.: Modeling and querying uncertain relational databases: a survey of approaches based on the possible worlds semantics. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems (IJUFKS)* 18(5), 565–603 (2010)
8. Chang, L., Yu, J.X., Qin, L.: Query ranking in probabilistic xml data. In: EDBT, pp. 156–167 (2009)
9. Kriegel, H.-P., Kunath, P., Renz, M.: Probabilistic nearest-neighbor query on uncertain objects. In: Kotagiri, R., Radha Krishna, P., Mohania, M., Nantajeewarawat, E. (eds.) DASFAA 2007. LNCS, vol. 4443, pp. 337–348. Springer, Heidelberg (2007)
10. Yang, S., Zhang, W., Zhang, Y., Lin, X.: Probabilistic threshold range aggregate query processing over uncertain data. In: Li, Q., Feng, L., Pei, J., Wang, S.X., Zhou, X., Zhu, Q.-M. (eds.) APWeb/WAIM 2009. LNCS, vol. 5446, pp. 51–62. Springer, Heidelberg (2009)
11. Cheng, R., Kalashnikov, D.V., Prabhakar, S.: Evaluating probabilistic queries over imprecise data. In: SIGMOD, pp. 551–562 (2003)
12. Xu, C., Wang, Y., Lin, S., Gu, Y., Qiao, J.: Efficient fuzzy top- k query processing over uncertain objects. In: Bringas, P.G., Hameurlain, A., Quirchmayr, G. (eds.) DEXA 2010. LNCS, vol. 6261, pp. 167–182. Springer, Heidelberg (2010)
13. Lian, X., Chen, L.: Efficient processing of probabilistic reverse nearest neighbor queries over uncertain data. *VLDB J.* 18(3), 787–808 (2009)
14. Jin, C., Yi, K., Chen, L., Yu, J.X., Lin, X.: Sliding-window top-k queries on uncertain streams. *PVLDB* 1(1), 301–312 (2008)
15. Tao, Y., Cheng, R., Xiao, X., Ngai, W., Kao, B., Prabhakar, S.: Indexing multi-dimensional uncertain data with arbitrary probability density functions. In: PVLDB, pp. 922–933 (2005)
16. Hjaltason, G.R., Samet, H.: Incremental distance join algorithms for spatial databases. In: SIGMOD, pp. 237–248 (1998)
17. Corral, A., Manolopoulos, Y., Theodoridis, Y., Vassilakopoulos, M.: Algorithms for processing k-closest-pair queries in spatial databases. *Data and Knowledge Engineering* 49, 67–104 (2004)
18. Yang, C., Lin, K.I.: An index structure for improving nearest closest pairs and related join queries in spatial databases. In: IDEAS, pp. 140–149 (2002)
19. Hou, U.L., Mamoulis, N., Yiu, M.L.: Computation and monitoring of exclusive closest pairs. *IEEE Trans. Knowl. Data Eng. (TKDE)* 20(12), 1641–1654 (2008)
20. Yuen, S.M., Tao, Y., Xiao, X., Pei, J., Zhang, D.: Superseding nearest neighbor search on uncertain spatial databases. *IEEE Trans. Knowl. Data Eng. (TKDE)* 22(7), 1041–1055 (2010)
21. Kriegel, H.-P., Kunath, P., Pfeifle, M., Renz, M.: Probabilistic similarity join on uncertain data. In: Li Lee, M., Tan, K.-L., Wuwongse, V. (eds.) DASFAA 2006. LNCS, vol. 3882, pp. 295–309. Springer, Heidelberg (2006)
22. Shan, J., Zhang, D., Salzberg, B.: On spatial-range closest-pair query. In: Hadzilacos, T., Manolopoulos, Y., Roddick, J., Theodoridis, Y. (eds.) SSTD 2003. LNCS, vol. 2750, pp. 252–269. Springer, Heidelberg (2003)

Synthesizing Routes for Low Sampling Trajectories with Absorbing Markov Chains

Chengxuan Liao, Jiaheng Lu*, and Hong Chen

DEKE, MOE and School of Information, Renmin University of China, Beijing, China
liaochengxuan@gmail.com {jiahenglu, chong}@ruc.edu.cn, jiahenglu@gmail.com

Abstract. The trajectory research has been an attractive and challenging topic which blooms various interesting location based services. How to synthesize routes by utilizing the previous users' GPS trajectories is a critical problem. Unfortunately, most existing approaches focus on only spatial factors and deal with high sampling GPS data, but low-sampling trajectories are very common in real application scenarios. This paper studies a new solution to synthesize routes between locations by utilizing the knowledge of previous users' low-sampling trajectories to fulfill their spatial queries' needs. We provide a thorough treatment on this problem from complexity to algorithms. (1) We propose a shared-nearest-neighbor (SNN) density based algorithm to retrieve a transfer network, which simplifies the problem and shows all possible movements of users. (2) We introduce three algorithms to synthesize route: an inverted-list baseline algorithm, a turning-edge maximum probability product algorithm and a hub node transferring algorithm using an Absorbing Markov Chain model. (3) By using real-life data, we experimentally verify the effectiveness and the efficiency of our three algorithms.

1 Introduction

With the pervasive use of GPS-enabled mobile devices, people are able to achieve their location histories in a form of trajectories. The GPS trajectories reflect genuine travel recommendations implying users' life interests and preferences. The research problem in this paper is to answer users' route plan queries using these trajectories. The route planning service is useful especially for users who are traveling in unfamiliar areas or when they are hiking, mountain climbing or cycling outdoors without road network.

The standard approaches to synthesize routes are the shortest or fastest path algorithms [8, 14]. They regard spatial distance as the most important factor in optimizations that guiding their route selection. However, we observe that one finds out suitable route for places of interest (POIs) which does not necessarily to be the shortest path. There are multiple reasons to determine route selection. For example, the width or tonnage limit of road prohibits vehicle to pass. The drivers are reluctant to trap in a traffic jam tending not to choose congested

* Corresponding author.

road segments in their driving route. Also, a park traveler is probably to follow a route that covers most of the attractions other than travel in a shortest path from the entrance to exit gate which miss most of the POIs. Works [20, 13, 10] analyze users' travel behaviors by mining sequential patterns of trajectories to predicate or planning routes. The trajectories pattern mining helps us to make a turn at road cross in some cases. However, it is not feasible for all trajectories following patterns at every turning point. For some low-frequent or low-sampling trajectories, it is even hardly to establish patterns. In [23], Tao et al. discuss object moving predictions. It is also not suitable for our problem not only because it mainly focuses on a general prediction framework, but also it does not utilize the features of trajectories.

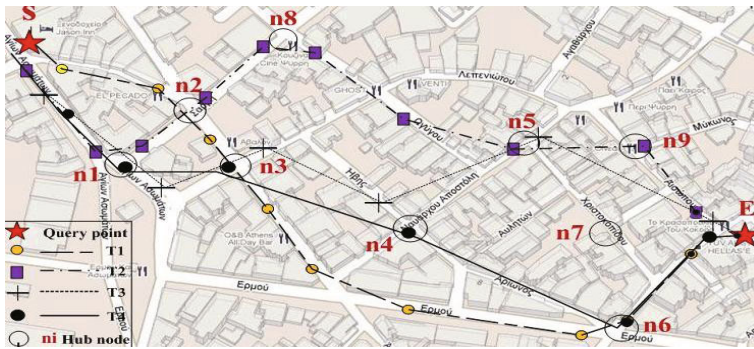


Fig. 1. Synthesizing Routes from Low-Sampling Raw Trajectories

Example 1. We use this example to illustrate the main challenge of our problem and the shortcomings of existing works. In Figure 1, users want to find a route from the start location S to destination E . There are 4 existing trajectories $T1$ to $T4$ and 9 intermediate hub nodes in the Figure 1. Historical trajectories are usually not accurate due to measurement errors and in a low-sampling rate. Thus, two consecutive points may be far away from each other. For example, we cannot decide if $n5$ in trajectory $T3$ ($S-n1-n3-n4-n5-E$) moves to the point near destination E by hub node $n7$ or $n9$. Thus, we cannot directly choose one trajectory from $T1$ to $T4$ to depict a specific route. By considering both the length and previous users' trajectories as travel recommendation, we think our hub node transfer algorithm is a better solution to route synthesizing problem.

To overcome the difficulties of uncertainty and complexity brought by low-sampling data and possibilities of individual road segments' combination to synthesize routes, we present a framework, which effectively transforms the trajectories to points on discrete network and apply the Absorbing Markov Chain for route planning. It first constructs hub nodes and transfer edges based on the local, adjacent points in trajectory histories. Given the transfer network, our method derives the transfer probabilities between the hub nodes and automatically designs some routes based on the support function with the Absorbing Markov Chain model.

To process the synthesizing routes problem efficiently, we propose three algorithms by learning the knowledge from historical raw trajectories, which take into account the length and support rate of trajectories. The contributions and the organization of this paper are summarized as follows.

1. We introduce a new synthesizing routes solution. To our best knowledge, this is the first work to plan route with comprehensive consideration of historical user's experiences (Section 2, 3).
2. We provide a SNN density based algorithm to extract the transfer network from the raw trajectories. And we propose a hub node transfer algorithm and an edge maximum probabilities product algorithm to synthesize routes over a transfer network (Section 4).
3. We use a breadth first prune method and expected turning steps calculating by Fundamental Matrix to optimize our algorithms (Section 4).
4. We conduct extensive experiments among inverted list baseline algorithm, the shortest-path algorithm, the hub node transfer algorithm and the edge maximum probabilities product algorithm (Section 5).

2 Related Work

Recently, different constraints and approaches have been proposed with conventional spatial queries to trajectories data. Trajectories pattern mining [26,19,9] help to some extent in synthesizing routes. Zheng et al. [26] adopt to explore frequent path segments or sequences of POIs. Mamoulis et al. [19] analyze regional periodic movements to find patterns. While in [9], Leticia et al. introduce constraints evaluation in categorical sequential pattern mining. All these proposals, however, ignore that query positions may not be on any frequent patterns, and they are difficult to handle cases for low-sampling trajectories.

Petko et al. [2] use symbolic representations to solve trajectories join. Yun et al. [4] also design and evaluate trajectory join algorithms. Although joining trajectories, it assembles segments of trajectories into routes. It cannot apply this solution to our problem. Li et al. [17] select hot routes based on a density algorithm FlowScan which finds high density connected trajectories in a cluster way. In [22], it also searches popular routes by checking whether a trajectory is passing through a certain number of objects. Works in [22,17] find hot routes or popular routes with regard to numbers of trajectories more than a threshold. Our work distinguishes itself from [22,17,5] where we focus on synthesizing routes on a low-sampling rate. Thus, we can handle the low-threshold conditions which are difficult for [22,17,5] to process.

Other works include density based cluster method DBSCAN in [7], CLIQR in [1] and shared nearest neighbor (SNN) cluster method in [12]. Due to the noise and biased GPS trajectories data, DBSCAN does not fit this problem since trajectories data usually contains clusters of diverse densities. According to the density based definition of core points in DBSCAN, it cannot identify the core points of varying density clusters correctly. Thus, we extend the cluster definition in SNN other than DBSCAN by introducing direction changes to help

us to retrieve the transfer network. Moreover, we also research searching similar trajectories [24, 3, 15] by different similarities functions, clustering trajectories in [16, 25] and the shortest path or fastest path algorithms in [6, 8, 14]. Their solutions do not apply to our problem of using users’ historical raw trajectories that implying their travel experiences to synthesize routes.

3 Preliminaries

In this section, we will give the preliminaries and formally define the problem of synthesizing routes for low-sampling-rate GPS trajectories.

GPS Trajectory: A GPS trajectory T is a sequence of GPS points with the time interval between any consecutive GPS points not exceeding a certain threshold ΔT , i.e. $Traj: p_1 \rightarrow p_2 \rightarrow \dots \rightarrow p_n$, and $0 < p_{i+1}.t - p_i.t < \Delta T (1 \leq i < n)$. Here, we focus on low-sampling rate GPS trajectories with $\Delta T \geq 1$ minute.

Hub Node: A hub node n is a point where trajectories change directions. We extract by the SNN cluster method from raw trajectories or end locations of a trajectory. Travelers make a turn at hub nodes. The moving direction of a GPS point p_i represents by $\overrightarrow{p_i p_{i+1}}$.

Route: A route R is a sequence of GPS points which the starting (ending) point is an existing GPS point or the nearest hub node to user’s query point in the transfer network, i.e. $R: n_1 \rightarrow n_2 \rightarrow \dots \rightarrow n_n$, which n_i is a hub node and (n_j, n_{j+1}) , $(1 < j \leq n)$, is an existed transfer edge.

Transfer Network: A transfer network $G (N, E)$ is a directional graph indicating the users’ movements between locations. N represents a set of hub nodes. E is a collection of transfer edges connecting hub nodes. If there exists at least a trajectory between adjacent hub node A and B , we consider the part between A and B as a transfer edge. If users start from an end point of a trajectory, no routes can reach the destination. One solution is to enhance the transfer network to a bidirectional one with minor additional changes.

Support Function: $S(R) = e^{\alpha P(R) - \beta L(R)}$ ($\alpha + \beta = 1, 0 \leq \alpha$ and $\beta \leq 1$). $P(R)$ represents a route’s transfer probability. $L(R)$ states a route’s length. We use a transfer probability $P(n_i \rightarrow n_j)$ to represent hub node n_i moves to n_j .

Now the problem of route synthesizing is defined as:

Given a trajectory database D and a query $Q (p_s, p_e)$, where p_s and p_e denote the start and destination points, synthesize a route that has maximum $S(R)$ from p_s to p_e with real trajectories points.

4 Algorithms for Synthesizing Routes

We analyze the users’ traveling behaviors and introduce three algorithms to synthesize routes in this section. First, we extract a transfer network and introduce

a baseline algorithm. Then, we provide a turning edge maximum probability product algorithm based on the transfer network. And we create a hub node transferring algorithm using an Absorbing Markov Chain model.

4.1 Retrieve Transfer Network

In order to synthesize route by learning users' previous knowledge from raw trajectories data, we build a transfer network to better process the trajectories without road network situation. If there is a road map, we could build this transfer network by using map-matching [18]. To address the challenge, we provide three key observations to motivate our approaches.

Observation 1: True paths tend to be not roundabout.

Observation 2: The density of trajectory points is higher in intersection areas compared with adjacent edges.

Observation 3: People will not change their moving direction until they make a turn. People make turns at the road intersecting areas. The angle of trajectory point changes more closer to $\pi/2$, the higher that the area has an intersection.

Thus, we could find hub node where people make a direction change by density cluster method. The point density varies widely in different regions of intersection. Because some intersections are hot consisted by hundreds of points, others are unpopular which people travel less time in our trajectories data set. We choose to extend SNN rather than other density cluster methods.

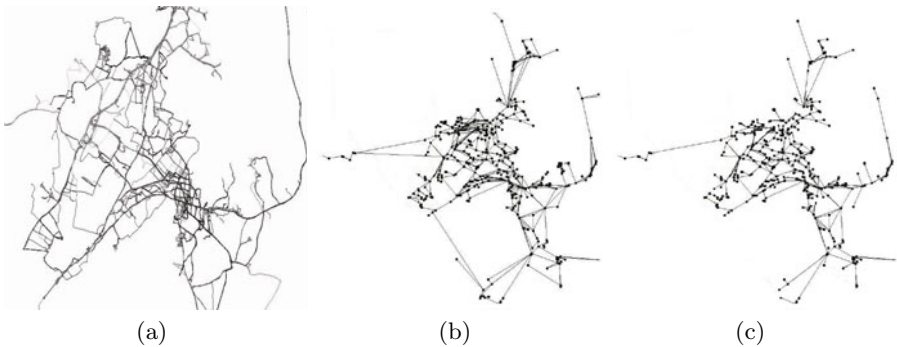


Fig. 2. (a). Truck trajectories distribution. (b). A certain part of raw trajectories after using the SNN method when $k=5$, $\beta = 2$. (c). Based on the result in (b), we prune the less supportive connected edges and remove edge that are not orthogonal.

In [12], a SNN graph is constructed from a proximity matrix as follows. A link is built between a pair of point i and j if and only if i and j are in each other's closest k nearest neighbor lists. In the SNN graph, the shared number of two points' nearest neighbors can be used as the weights of links between two nodes. Or we can use the function below to compute by considering the order

of the near neighbors. For example, if i and j are two points. The weight of the link between i and j is defined as:

$$Strength(i, j) = \sum (k + 1 - m) * (k + 1 - n), \text{ where } i_m = j_n. \quad (1)$$

In equation (1), k is the size of nearest neighbor list, m and n are the positions in point i and j 's lists. Given a threshold, we remove the edge if its weight is less than it. And all the connected components in the resulting graph are our clusters. We also expand the algorithm with an angle filter that $\text{angle}(p, q) = |\sin \theta|^\beta$. (θ is the angle difference between p and q ranging from 0 to π . β is a turning parameter.) After discovering all the clusters, we calculate each of them as a hub node whose location is approximated by the average coordinate. In Figure 2, we make 292,364 trajectory points into cluster making a transfer network with 651 nodes. It greatly reduces the node size that we would compute when synthesizing routes.

4.2 Baseline Algorithm

If we build inverted list for every hub node, a loop algorithm can be used to solve the route synthesizing problem for a given query. Its pseudo code is shown in Algorithm 1. For each hub node, we build an inverted list by every trajectory passing it. We check every trajectory on hub node that whether it reaches the destination. Then we expand in a breadth-first way in a priority queue. Combining the traces when we reach the destination to form a route, we compare all these possible routes in length to provide the final result. The time complexity is $O(m^n)$ (m is the number of points, n is the number of trajectories), which is too high. In order to obtain a better performance, we propose the following prune strategy.

Algorithm 1. Min-Hop Algorithm

Input: a query (p_s, p_e)

Output: a route with minimum length in priority queue

```

1: Build an inverted list  $P_i(T_1, T_2, \dots, T_k)$ 
2: Build a hash set  $H \leftarrow \emptyset$ 
3: ENQUEUE( $H, p_s$ ),  $p_s$ 's hash value = 1
4:  $\pi[s] \leftarrow \text{NIL}$ 
5: for ( $i = 1, i \leq \text{length of hash set}, i++$ ) do
6:   for ( $j = 1, j \leq \text{length of } T_j \text{ of hash value } i, j++$ ) do
7:     for (first point  $p_k$  after  $p_s$  in  $T_j$ ,
8:        $k \leq \text{length of maximize number of points in } T_k, k++$ ) do
9:        $\pi[s] \leftarrow p_{next}$ 
10:      if  $p_{next} \neq p_e$  then
11:        ENQUEUE( $H, p_k$ )
12:      else
13:        BREAK ALL
14: return Compute length in  $\pi[s]$ 

```

Prune Strategies: The work [21] states the Incremental Euclidean Restriction (IER) algorithm which enlightens us to prune the synthesizing areas. Assuming that only the shortest route is maintained, we first retrieve a route R1 from p_s to p_e for query $Q(p_s, p_e)$. Then, the network distance $L(R1)$ of R1 is computed. We use p_s as the center and $L(R1)$ as the radius to draw a circle. Due to the Euclidean lower bound property, routes containing hub nodes outside of the area should not be the results. In other words, we do not have to consider hub nodes outside the bound since the Euclidean distance of p_s and hub nodes outside the bound is more than $L(R1)$. Moreover, we can update the bound by other candidate routes whose length less than $L(R1)$.

4.3 Turning Edge Maximum Probability Product Method

After prune the synthesizing area, the time complexity is still too high to use in real scenarios. The hub node transfer probability is proposed to represent the likelihood of node leading users to their destination. Adjacent matrix is built as a one step transient matrix to observe travelers' historical behaviors. $P(z)$ represents the number of trajectories on (n_i, n_j) . $P(m)$ represents the number of trajectories on all outgoing edges. Thus, the hub node turning probability of moving from n_i to n_j on adjacent edge $e(n_i, n_j)$ will be:

$$Pr(n_i \rightarrow n_j) = \frac{P(z)}{P(m)} \quad Pr^t(n_i \rightarrow d) = \sum_{j=1}^t p_{n_i, d}^j \quad (2)$$

If we take the travel on transfer network as Random Walk behavior on a directed graph, the transition probability from n_i to n_j equals to $Pr(n_i \rightarrow n_j)$. If we conduct such a random walk on a transfer network following the turning probability, we will reach the destination when we select the edge that has a higher probability along our route. A higher hub node transfer probability implies more historical trajectories head for the destination through this. In addition, if the length of a route is ten times of the shortest path's length, people tend not to choose this route. If a route is too small even smaller than the length of the shortest path, it fails the original objectives. Thus, we need to choose a proper length or numbers of hip of the route. If setting t be the maximum steps of our algorithm, we consider all possible connecting edges within t steps after leaving from start position p_s . Assuming that we arrive at destination after t times by N_t hub nodes. The p_{n_i, n_j}^t represents the first time moving from n_i to n_j probability. The $Pr^t(n_i \rightarrow d)$ is the sum of probability that we first arrive at d in 1, 2, \dots t step, which indicates the probability of moving from any n_i to destination d within t steps

After we discuss hub node transfer probability above, we can also use edge as the transfer indicator instead of hub node. The transfer probability of a turning edge (n_i, n_j) is computed as $Pr(n_i \rightarrow n_j)$. Synthesizing routes problem is computed by selecting the maximum product of the turning probabilities of all edges on it, considering the length and the number of edge as thresholds.

4.4 Hub Node Transfer Method

We introduce Absorbing Markov Chain model [11] to utilize probability in Equation (2) to synthesize routes. A Markov chain is absorbing if it has at least one absorbing state, and if from every state it is possible to go to an absorbing state.

$$P(i, j) = \begin{cases} 1 & \text{if } n_i \text{ is an absorbing state and } i = j \\ Pr_d(n_i \rightarrow n_j) & \text{if } n_i \text{ is a transient state and } i \neq j \\ 0 & \text{otherwise} \end{cases}$$

In transfer network, the destination node and the end point of trajectories without any outgoing edges are treated as an absorbing state. All other hub nodes are considered as transient states. We have the following canonical form for the transition matrix when there are r absorbing states and t transient states. I is an r-by-r matrix, 0 is an r-by-t zero matrix, R is a nonzero t-by-r matrix, and Q is a t-by-t matrix. The first t states are transient, and the last r states are absorbing. The entry p_{ij}^t of the matrix P^t is the probability when it is started from s_i after t steps in s_j .

$$P = \begin{array}{c|cc} & \text{TR.} & \text{ABS.} \\ \hline \text{TR.} & Q & S \\ \text{ABS.} & 0 & I \end{array} \quad P^t = \begin{array}{c|cc} & \text{TR.} & \text{ABS.} \\ \hline \text{TR.} & Q^t & * \\ \text{ABS.} & 0 & I \end{array} \quad (3)$$

The 1st to (t - 1)th states of a route are transient. A route transfers between transient states for t-1 times and finally jumps from a transient state to the destination at the tth step. For the first t-1 step, they are acquired from Q^{t-1} . And the probability of moving from a transient state to the destination is given in S. Consequently, the $p_{n_i,d}^t$ for a given n_i can be computed in Equation (4).

$$P_{n_i,d}^t = \sum_{n_k \in TR} (P^{t-1}(i, k) \cdot P(k, d)) \quad (4)$$

$$Pr^t(n_i \rightarrow d) = \sum_{j=1}^t P_{n_i,d}^j = \sum_{j=1}^t \sum_{n_k \in TR} (P^{t-1}(i, k) \cdot P(k, d)) \quad (5)$$

Thus, the transfer probability of a node n_i in t step to destination is determined by Equation (5).

$$V = [Pr^t(n_1 \rightarrow d), Pr^t(n_2 \rightarrow d), \dots, Pr^t(n_l \rightarrow d)]^{-1} \quad (6)$$

$$V = D + Q \cdot D + Q^2 \cdot D + \dots + Q^{t-1} \cdot D \quad (7)$$

Given a destination d and parameter t, we have Q and S, and d in ABS. We use D as the corresponding column vector of node d in the sub-matrix S. The result V is calculated by Equation (7). We compute the support function S(R) in Section 3 after acquiring the directional transfer network G(N,E) and transfer probabilities.

Algorithm 2. Hub Node Transfer Probability

Input: A transfer network $G(N,E)$

Output: A set of vectors V

- 1: $V \rightarrow \emptyset$
 - 2: **for** each hub node $n_i \in N$ **do**
 - 3: set v for n_i
 - 4: set n_i as the destination
 - 5: construct the transition matrix P
 - 6: compute v by P // See Equation (6)
 - 7: re-organize P in a canonical form
 - 8: acquire Q, S from P
 - 9: push v into V // See Equation (7)
 - 10: **return** V
-

The time complexity to compute the matrix multiplication is $O(t * m^3)$. The space complexity is $O(m^2)$ storing the pre-computed vectors. The complexity of Algorithm is $O(E+N\log N)$, where E is the number of edges, and N is the number of nodes. We use following lemmas to prove the correctness of our Algorithm.

Lemma 1. *Given an absorbing Markov chain, the fundamental matrix is $N = (I - Q)^{-1}$. The entry n_{ij} of N gives the expected number of times that it moves from transient state s_i to s_j .*

Proof: Let $(I - Q)x = 0$; $x = Qx$. Iterating this that $x = Q^n x$. Since $Q^n \rightarrow 0$, $Q^n x \rightarrow 0$, so $x = 0$. Thus $(I - Q)^{-1} = N$ exists. Note $(I - Q)(I + Q + Q^2 + \dots + Q^n) = I - Q^{n+1}$. Thus both sides multiply N : $I + Q + Q^2 + \dots + Q^n = N(I - Q^{n+1})$. $n \rightarrow \infty$, $N = I + Q + Q^2 + \dots + Q^n$. Let s_i and s_j be two transient states. If the chain is in state s_j after k steps $P(X^k = 1) = q_{ij}^k$, otherwise $P(X^k = 0) = 1 - q_{ij}^k$, where q_{ij}^k is the ij th entry of Q^k . These equations hold for $k = 0$ since $Q^0 = I$. Therefore, $E(X^k) = q_{ij}^k$. $E(X^0 + X^1 + \dots + X^n) = q_{ij}^0 + q_{ij}^1 + \dots + q_{ij}^n$. $n \rightarrow \infty$, $E(X^0 + X^1 + \dots) = q_{ij}^0 + q_{ij}^1 + \dots = n_{ij}$.

Lemma 2. *Let t_i be the expected number of steps before the chain is absorbed, and let t be the column vector whose i th entry is t_i . Then $t = Nc$, where c is a column vector all of whose entries are 1.*

Proof: If we add all the entries in the i th row of N , we will have the expected number of times in any of the transient states for a given starting state s_i , that is, the expected time required before being absorbed. Thus, t_i is the sum of the entries in the i th row of N .

Lemma 3. *Let b_{ij} be the probability that an absorbing chain will be absorbed in the absorbing state s_j if it starts in the transient state s_i . Let B be the matrix with entries b_{ij} . Then B is an t -by- r matrix, and $B = NR$, where N is the fundamental matrix and R is as in the canonical form.*

Proof(sketch): We have

$$B_{ij} = \sum_n \sum_k q_{ik}^n r_{kj} = \sum_k \sum_n q_{ik}^n r_{kj} = \sum_k n_{ik} r_{kj} = (NR)_{ij}$$

5 Experiments

In this section, we use a real low-sampling trajectories data set¹ ($\Delta T \geq 1$ minute) to conduct our experiments. The truck data set consists of 276 trajectories containing 50 trucks to deliver concretes to various construction places around Athens metropolitan area in Greece for 33 distinct days. The synthesizing routes algorithms are implemented in JAVA and examined on a Windows platform with Intel Core 2 CPU (2.54GHz) and 2.0GB Memory. The process of extracting the transfer network is executed off-line.

We usually cannot obtain a clean data set, due to the GPS sampling flaws or noise factors. Thus, (1) we eliminate outlier points of the raw trajectories by considering physical limits such as vehicle speed limit. (2) We smooth the direction to alleviate the effect of GPS position fluctuation.

The pre-computation process consumes about 230 seconds and involves around $2.3 * 10^7$ node accesses for the complete data set. After clustering, we take all the intersections and the end points of trajectories as transfer nodes. Compared with an online road map web site OpenStreetMap, 651 out of 694 transfer nodes are correctly clustered by our algorithm, which produces an accuracy rate = 0.938. In Figure 2(C), the retrieved transfer network keeps the shapes of the raw trajectories quite well. We show the overhead of the extracting transfer network algorithm in Figure 3 (a) (b). Both of the clustering time and R-tree node access increase linearly with the number of trajectory points.

We first calculate the node transfer probability to build the transition matrix. Then we compute the vector V in Equation (6) for each node. Here, we compare the Min-Hop algorithm, turning Edge Maximum Probability Product (Edge-MPP) algorithm, Hub Node Transfer (Node-AMC) algorithm with the shortest path (SP) method in Figure (3), where the performance is measured by query time, length of result route, the number of hub node in result route and the number of visited transfer nodes. The shortest path solution is implemented by A* algorithm [23].

In Figure 3(c), we can see that the query time of the hub node transfer algorithm is about half of the shortest-path algorithm. The distance of query becomes longer, the better time performance of Node-AMC algorithm. Synthesizing routes by Node-AMC need less time than the shortest-path algorithm. On the other hand, the length of routes synthesized by Node-AMC is normally larger than the corresponding shortest-path routes. The route synthesized by Node-AMC is about 1/4 longer on average than the corresponding shortest-path routes in Figure C(f). It also illustrates that the shortest path may not always be the best one. Drivers are willing to take a slightly longer route in order to enjoy a higher quality road or to avoid traffic jams.

We can also see the difference between our algorithm and the shortest-path algorithm in the number of hub nodes contained by the resulting route. For the shortest-paths which containing more than 6 transfer nodes, the corresponding routes synthesized by Node-AMC have fewer hub nodes in Figure 3(d). It may

¹ The Rtree-Portal, <http://www.rtreportal.org>

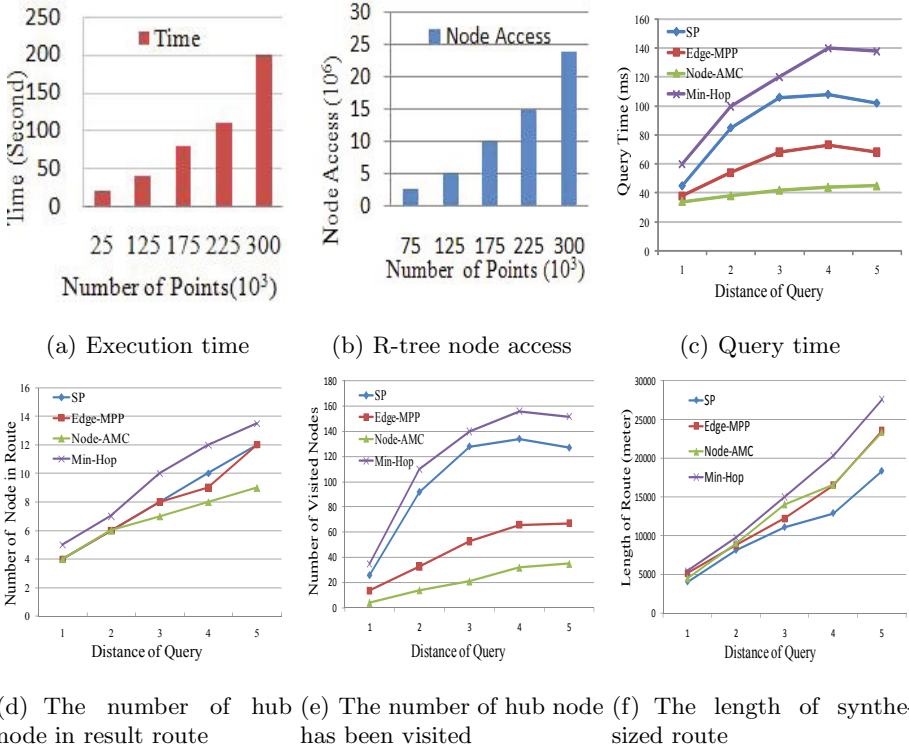


Fig. 3. Performance of the Algorithm

imply that the driver is reluctant to make turns for Long-distance transport of goods. The synthesizing route by Node-AMC contains 9 nodes on average comparing 12 nodes of corresponding shortest-path routes.

As shown in Figure 3(e), the number of hub node has been visited before finding the resulting route is different between our algorithm and the corresponding the shortest-path algorithm. The hub node transfer algorithm usually travel fewer transfer nodes before it finally finds the destination. It also shows that the hub node transfer algorithm has a better time performance than corresponding shortest-path algorithm. Therefore, the search region of the Node-AMC algorithm is much smaller. For the Edge-MPP algorithm, it has a performance between the Node-AMC and the shortest algorithms. There is no ground truth for us to use as a reference to evaluate the goodness of a search result. Also, the min-hop algorithm and the turning edge maximum probability product method may fail to find a global best route in some cases as it makes an immediate move after checking the turning probability in adjacent edges. Overall, the algorithms have a good performance.

6 Conclusions

In this paper, we have proposed a new solution synthesizing routes between any two given locations by learning knowledge from previous travelers' low sampling trajectories. We have extended the shared nearest neighbor cluster method for retrieving a transfer network from raw trajectories. Based on hub node transfer probabilities, we have utilized Absorbing Markov Chain as well as edge maximum probabilities product to synthesize routes. Finally, experimental results have been verified the effectiveness and efficiency of our methods in real application scenarios.

Acknowledgement. This paper is supported by the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University of China (No: 11XNJ003).

References

1. Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic subspace clustering of high dimensional data for data mining applications. In: SIGMOD, pp. 94–105 (1998)
2. Bakalov, P., Hadjieleftheriou, M., Keogh, E.J., Tsotras, V.J.: Efficient trajectory joins using symbolic representations. In: Mobile Data Management, pp. 86–93 (2005)
3. Chen, L., Özsu, M., Oria, V.: Robust and fast similarity search for moving object trajectories. In: SIGMOD, pp. 491–502. ACM, New York (2005)
4. Chen, Y., Patel, J.M.: Design and evaluation of trajectory join algorithms. In: GIS, pp. 266–275 (2009)
5. Chen, Z., Shen, H., Zhou, X.: Discovering popular routes from trajectories. In: ICDE, pp. 900–911 (2011)
6. Dijkstra, E.: A note on two problems in connection with graphs. *Numerische Mathematik* 1(269-270), 269–271 (1959)
7. 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)
8. Goldberg, A.V., Harrelson, C.: Computing the shortest path: A* search meets graph theory. In: SODA, pp. 156–165 (2005)
9. Gómez, L.I., Vaisman, A.A.: Efficient constraint evaluation in categorical sequential pattern mining for trajectory databases. In: EDBT, pp. 541–552 (2009)
10. Gonzalez, H., Han, J., Li, X., Myslinska, M., Sondag, J.P.: Adaptive fastest path computation on a road network: A traffic mining approach. In: VLDB, pp. 794–805 (2007)
11. Grinstead, C., Snell, J.: Introduction to probability. Amer. Mathematical Society, Providence (1997)
12. Jarvis, R., Patrick, E.: Clustering using a similarity measure based on shared near neighbors. *IEEE Transactions on Computers*, 1025–1034 (1973)
13. Jeung, H., Liu, Q., Shen, H.T., Zhou, X.: A hybrid prediction model for moving objects. In: ICDE, pp. 70–79 (2008)
14. Kanoulas, E., Du, Y., Xia, T., Zhang, D.: Finding fastest paths on a road network with speed patterns. In: ICDE, p. 10 (2006)

15. Keogh, E.J., Pazzani, M.J.: A simple dimensionality reduction technique for fast similarity search in large time series databases. In: Terano, T., Chen, A.L.P. (eds.) PAKDD 2000. LNCS, vol. 1805, pp. 122–133. Springer, Heidelberg (2000)
16. Lee, J.-G., Han, J., Whang, K.-Y.: Trajectory clustering: a partition-and-group framework. In: SIGMOD, pp. 593–604 (2007)
17. Li, X., Han, J., Lee, J.-G., Gonzalez, H.: Traffic density-based discovery of hot routes in road networks. In: Papadias, D., Zhang, D., Kollios, G. (eds.) SSTD 2007. LNCS, vol. 4605, pp. 441–459. Springer, Heidelberg (2007)
18. Lou, Y., Zhang, C., Zheng, Y., Xie, X., Wang, W., Huang, Y.: Map-matching for low-sampling-rate gps trajectories. In: GIS, pp. 352–361 (2009)
19. Mamoulis, N., Cao, H., Kollios, G., Hadjieleftheriou, M., Tao, Y., Cheung, D.W.: Mining, indexing, and querying historical spatiotemporal data. In: KDD, pp. 236–245 (2004)
20. Monreale, A., Pinelli, F., Trasarti, R., Giannotti, F.: Wherenext: a location predictor on trajectory pattern mining. In: KDD, pp. 637–646 (2009)
21. Papadias, D., Zhang, J., Mamoulis, N., Tao, Y.: Query processing in spatial network databases. In: VLDB, pp. 802–813 (2003)
22. Sacharidis, D., Patroumpas, K., Terrovitis, M., Kantere, V., Potamias, M., Mouratidis, K., Sellis, T.K.: On-line discovery of hot motion paths. In: EDBT, pp. 392–403 (2008)
23. Tao, Y., Faloutsos, C., Papadias, D., Liu, B.: Prediction and indexing of moving objects with unknown motion patterns. In: SIGMOD, pp. 611–622 (2004)
24. Vlachos, M., Gunopulos, D., Kollios, G.: Discovering similar multidimensional trajectories. In: ICDE, pp. 673–684 (2002)
25. Won, J.-I., Kim, S.-W., Baek, J.-H., Lee, J.: Trajectory clustering in road network environment. In: CIDM, pp. 299–305 (2009)
26. Zheng, Y., Zhang, L., Xie, X., Ma, W.-Y.: Mining interesting locations and travel sequences from gps trajectories. In: WWW, pp. 791–800 (2009)

Approximate Continuous K -Nearest Neighbor Queries for Uncertain Objects in Road Networks

Guohui Li¹, Ping Fan^{1,2,*}, and Ling Yuan¹

¹ School of Computer Science and Technology,

Hua Zhong University of Science and Technology, Wuhan, China

² School of Computer Science, Xianning University, Hubei, China

{Guohuilihw, fanping1028, cherry yuanling}@gmail.com

Abstract. Continuous K nearest neighbor queries (CKNN) on moving objects retrieves among all moving objects the K -Nearest Neighbors (KNNs) of a moving query point within a given time interval. Since the frequent updates of object locations make it complicated to process CKNN, the cost for retrieving the exact CKNN data set is expensive, particularly in highly dynamic spatio-temporal applications. In some applications (e.g. finding my nearest taxis while I am moving within the next 5 minutes), it is not necessary to obtain the accurate result set. For these applications, we introduce a novel technique, Moving state based Approximate CKNN (MACCKNN), to approximate the CKNN query results with certain accuracy to make the query process more efficient by using Moving State of Uncertain Object (MSUO) Model and guarantee certain accuracy. We evaluate the MACCKNN technique with simulations and compare it with a traditional approach. Experimental results are presented to demonstrate the utility of our new approach.

Keywords: Continuous K -Nearest Neighbor query, Road network, Uncertainty, Moving state.

1 Introduction

The problem of K nearest neighbor (KNN) queries in spatial databases has been studied by many researchers. This type of query is frequently used in Geographical Information Systems and is defined as: given a set of spatial objects and a query point, find the K nearest objects to the query. In recent years, with the rapid development of wireless communication and positioning technology, mobile users can enjoy sorts of convenient services such as location information query [1], nearest neighbor query [2-6], range query, traffic conditions query etc. K Nearest Neighbor (KNN) query [7-12] is one important type of these services, which is to find the K Nearest Neighbors (KNN) of a moving user among all moving objects.

In order to improve the efficiency of CKNN query process for the moving objects, some previous work in this field assumed that the velocity of each moving object is

* Corresponding author: Ping Fan, School of Computer Science, Huazhong University of Science and Technology, Wuhan, China, EMAIL: fanping1028@gmail.com

fixed, since the motion of each object can be precisely determined under this assumption [13-16]. However, in road networks of real world, the objects move arbitrarily. The release of the fixed velocity makes it difficult to determine precisely the distance between moving objects and query object, which leads to a more complicated CKNN query process. Existing methods in CKNN query mostly aim at Euclidean spaces [11, 15, 17]. For the moving objects in road networks [18,19], the distance computation between data object and query point is quite different from that of Euclidean spaces. The distance in road network is defined as the length of the shortest path connecting data object and query point. Huang [20] proposed a method based on periodical snapshot recalculation to process CKNN query where both data objects and query points move continuously in a road network. The main problem of this method lies in how to set the snapshot recalculation period appropriately. Long period setting will affect the CKNN quality and short period can bring too much communication cost.

In some applications, such as finding my nearest taxis while I am moving within the next 5 minutes, it is not necessary to obtain the accurate result set. If we just focus on retrieving the approximate CKNN query result, the process cost could be greatly reduced. In this paper, a novel approach is presented to process approximate CKNN queries efficiently based on the moving state of objects. A Moving State of Uncertain Object (MSUO) model is proposed to determine the moving state of object which is motivated by the *moving_state* value determination process of objects with fixed velocity in our paper [16]. The Moving state based Approximate Continuous K nearest neighbor query (MACKNN) algorithm is composed of two phases: One is the pruning phase, which can scale down the object candidates for the CKNN query within a given time interval; Another is the refining phase, which can determine the time sub-intervals where the approximate CKNN query results are obtained.

The rest of this paper is organized as follows. Section 2 reviews related work in continuous and approximate KNN query. Moving State of Uncertain Object (MSUO) model is presented in Section 3. Section 4 presents the proposed MACKNN algorithm involving pruning phase and refining phase to obtain the approximate CKNN query results efficiently. Section 5 evaluates the performance of our proposed methods with a set of simulation experiments in a real road network. Section 6 concludes the paper.

2 Related Work

Processing KNN queries over moving objects in road networks is a hot research topic in recent years. In this section, we first review the existing continuous K nearest neighbor query methods, and then discuss the related work that deals with the approximate continuous K nearest neighbor queries.

For continuous nearest neighbor monitoring in a road network, Mouratidis et al. [19] proposed an Incremental Monitoring Algorithm (IMA), which can process K nearest neighbor monitoring over moving objects and query points and re-calculate query results whenever an update occurs. At each re-evaluation time, by processing the object updates, the query updates and edge updates, the query results may be incrementally obtained from the result at the previous timestamp. Thus the overhead incurred by processing repeated queries can be reduced. However due to the nature of discrete location updates, the KNNs of the query object within two successive updates

are unknown. Thus, IMA would return invalid results between two successive update timestamps. Y.-K.Huang et al. [20] proposed a continuous monitoring method over moving objects in a road network. The procedure of this monitoring method is divided into two phases, the pruning phase and the refinement phase. The main aim of pruning phase is to calculate the pruning distance. With the given pruning distance, unqualified objects are efficiently pruned. Then in the refinement phase, candidates are verified whether they belong to the K NNs of query q or not. The inadequacy of this method is that the pruning distance is too large. Thus there are too many objects monitored and the cost for computing the distances between the query point and objects could be extremely high. Moreover, in the refinement phase, each candidate object should be examined whether it can replace the K^{th} NN or not. Thus the K NN result may be invalid due to the heavy overhead.

For the approximate continuous K nearest neighbor queries, Arya et al. [21] proposed a sparse neighborhood graph RNG, which sets the error variance constant ϵ and the angular diameter δ to tune the size of divided neighboring areas around one data point. This algorithm reduces the time and space requirements for processing data objects significantly. Ferhatosmanoglu et al. [22] utilized VA+-file to solve approximate nearest neighbor queries. This algorithm uses bit vectors to represent a division of the data space. In its first phase it calculates and compares distance of these spatial cells, then it computes and checks real distance between data points of nearby cells in its second phase. Berrani et al. [23] proposed another algorithm to solve approximate K nearest neighbor queries. Firstly it clusters data points in space and represents these clusters as spatial spheres. Then it sets the error variance constant ϵ to tune approximate spheres for original clusters. Data points located in the approximate clustering sphere are candidate results for the approximate K NN queries. However, all these algorithms are aimed at high dimensional and fixed data points. Yu-Ling Hsueh et al. [24] proposed an AC- K NN algorithm by defining split points on the query trajectories, moving objects can only update their locations on a segment basis regardless of the change of their velocities. To our best knowledge, the problem of approximate continuous K nearest neighbor queries for moving objects with uncertain velocity in road networks has not been studied before.

3 Moving State of Uncertain Object (MSUO) Model

In our paper [16], a MSO model has been proposed to determine the *moving_state* value of object with fixed velocity in road networks. When the fixed velocity of the object is released, the determination of *moving_state* value of object becomes more complicated. A Moving State of Uncertain Object (MUSO) model is proposed here to resolve such problem. The symbols \downarrow and \uparrow represent an object is getting closer to and moving away from another object respectively. The velocity of moving object is within the interval $[v^-, v^+]$. When the object o_i and the query point q are moving in the same direction on their shortest path with the velocity interval $[o_i.v^-, o_i.v^+]$ and $[q.v^-, q.v^+]$, the moving state of o_i can be determined by the help of calculating the moving degree of o_i to q based on their velocity intervals. The larger the moving degree value is, the faster the object is getting closer to or moving away from the query point. According to the method in literature [25], the definition of computation for the moving degree of a moving object is presented as follows:

Definition 3.1. For a moving object o_i with velocity interval $[o_i.v^-, o_i.v^+]$, and a query point q with velocity interval $[q.v^-, q.v^+]$, the moving degree of o_i to the query point q can be calculated as Equ.3.1.

$$dv(o_i, q) = 0.5 - \frac{(q.v^+ + q.v^-) - (o_i.v^+ + o_i.v^-)}{2 \times ((o_i.v^+ - o_i.v^-) + (q.v^+ - q.v^-))} \tag{Equ.3.1}$$

where $o_i.v^+ - o_i.v^- + q.v^+ - q.v^- \neq 0$, $dv(o_i, q)$ represents the moving degree of o_i to the query point q . If $dv(o_i, q) > 0.5$, the relative velocity of o_i to q is larger than the relative velocity of q to o_i . If $dv(o_i, q) \leq 0.5$, the relative velocity of o_i to q is smaller than the relative velocity of q to o_i .

There are four cases about different moving directions of the object o_i and query point q on their shortest path, as shown in Fig.1. The moving state of o_i is represented as $\downarrow o_i$ or $\uparrow o_i$, representing getting closer to or moving away from q respectively.

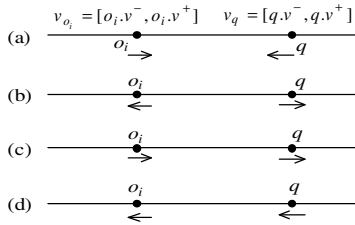


Fig. 1. Moving state of objects with uncertain velocity

1. The moving directions of object o_i and query point q are described in Fig.1 (a). In this case, o_i is getting closer to query point q , then the moving state of o_i is represented as $\downarrow o_i$. As the moving velocities of o_i and q are both within a interval, the network distance between the object o_i and q is also within a distance interval $[dq.o_i(t), Dq.o_i(t)]$, where $dq.o_i(t)$ indicates the minimal network distance and $Dq.o_i(t)$ indicates the maximal distance, calculated respectively as follows:

$$dq.o_i(t) = dq.o_i(t_0) - (|o_i.v^+| + |q.v^+|) \times (t - t_0)$$

$$Dq.o_i(t) = Dq.o_i(t_0) - (|o_i.v^-| + |q.v^-|) \times (t - t_0)$$

2. The moving directions of object o_i and query point q are described in Fig.1 (b). In this case, o_i is moving away from q , and then the moving state of o_i is represented as $\uparrow o_i$. The minimal and maximal network distance between the two objects at time t can be calculated as follows:

$$dq.o_i(t) = dq.o_i(t_0) + (|o_i.v^-| + |q.v^-|) \times (t - t_0)$$

$$Dq.o_i(t) = Dq.o_i(t_0) + (|o_i.v^+| + |q.v^+|) \times (t - t_0)$$

3. The moving directions of object o_i and query point q are described in Fig.1 (c). In this case, the moving state of object o_i to the query point q needs to be determined by the definition 3.1. If $dv(o_i, q) > 0.5$, the moving state of o_i is getting closer to q ,

represented as $\downarrow o_i$, otherwise, the moving state of o_i is moving away from q , represented as $\uparrow o_i$.

If $\downarrow o_i$, the minimal and maximal network distance between o_i and q at time t can be calculated as follows:

$$\begin{aligned} dq_{o_i}(t) &= dq_{o_i}(t_0) - (\|o_i.v^+ - |q.v^+\|) \times (t - t_0) \\ Dq_{o_i}(t) &= Dq_{o_i}(t_0) - (\|o_i.v^- - |q.v^+\|) \times (t - t_0) \end{aligned}$$

If $\uparrow o_i$, the minimal and maximal network distance between o_i and q at time t can be calculated as follows:

$$\begin{aligned} dq_{o_i}(t) &= dq_{o_i}(t_0) + (\|q.v^- - |o_i.v^+\|) \times (t - t_0) \\ Dq_{o_i}(t) &= Dq_{o_i}(t_0) + (\|q.v^+ - |o_i.v^-\|) \times (t - t_0) \end{aligned}$$

4. The moving directions of object o_i and query point q are described in Fig.1 (d).

In this case, the moving state of object o_i to the query point q also needs to be determined by the definition 3.1. If $dv(o_i, q) > 0.5$, the moving state of o_i is moving away from q , represented as $\uparrow o_i$, otherwise, the moving state of o_i is getting closer to q , represented as $\downarrow o_i$.

If $\uparrow o_i$, the minimal and maximal network distance between o_i and q at time t can be calculated as follows:

$$\begin{aligned} dq_{o_i}(t) &= dq_{o_i}(t_0) + (\|o_i.v^- - |q.v^+\|) \times (t - t_0) \\ Dq_{o_i}(t) &= Dq_{o_i}(t_0) + (\|o_i.v^+ - |q.v^-\|) \times (t - t_0) \end{aligned}$$

If $\downarrow o_i$, the minimal and maximal network distance between o_i and q at time t can be calculated as follows:

$$\begin{aligned} dq_{o_i}(t) &= dq_{o_i}(t_0) - (\|o_i.v^- - |q.v^+\|) \times (t - t_0) \\ Dq_{o_i}(t) &= Dq_{o_i}(t_0) - (\|o_i.v^+ - |q.v^-\|) \times (t - t_0) \end{aligned}$$

With the proposed MSUO model, we can determine the moving state of moving object o_i to the query point q as $\uparrow o_i$ or $\downarrow o_i$, and calculate the minimal and maximal distances between o_i and q as a distance interval $[do_i, q(t), Do_i, q(t)]$.

4 Moving State Based ACKNN (MACKNN) Algorithm

In order to process the CKNN queries over moving objects in road network more efficient, we propose a Moving State based Approximate CKNN (MACKNN) algorithm involving pruning phase and refining phase. The pruning phase is used to scale down the CKNN query region by cutting a pruning distance. In the refining phase, since when the sequence of an orderly candidate set changes, the sequence of two subsequent nearest neighbors would change firstly, the time interval is divided into several subintervals by finding the sequence changing time points of two subsequent nearest neighbors with the help of MSUO model in Section 3. In each division time

subinterval, the KNN query result could be certain. In the following, the pruning phase and refining phase are described in detail.

4.1 Pruning Phase

The main idea of pruning phase in the CKNN queries is to scale down the CKNN query region by using a pruning distance, denoted as $d_{pruning}$. For any moving object, if the minimal distance between this object and query point is less than or equal to $d_{pruning}$ within a given time interval, this moving object can be considered in the process of CKNN queries, otherwise this object will be pruned. The calculation of the pruning distance is defined as $d_{pruning} = D_{o,q}^{max} + D_{add}^{max}$. The explanations about $D_{o,q}^{max}$ and D_{add}^{max} are presented in the following.

$D_{o,q}^{max}$ indicates the maximal distance among all the maximal distances between moving objects and query point within a given time interval (denoted as $[t_0, t_n]$). In order to continuously monitor the KNN result within the time interval $[t_0, t_n]$, we divide the time interval into several time subintervals within which the directions of moving objects are certain. At the start time t_0 for K moving objects nearest to the query point, we calculate the time instants for each object and query point reaching an intersection of the road network with their maximal moving velocity. The fastest one of these time instants, denoted as t_1 is chosen to divide the time interval $[t_0, t_n]$ into two subintervals: $[t_0, t_1]$ and $[t_1, t_n]$. Next, the subinterval $[t_1, t_n]$ is divided into $[t_1, t_2]$ and $[t_2, t_n]$ with the similar process. This process is repeated until the chosen time instant is larger than t_n . Thus, the time interval $[t_0, t_n]$ is divided into $[t_0, t_1], [t_1, t_2], \dots, [t_{j-1}, t_j], \dots, [t_{n-1}, t_n]$. For each time subinterval, we just need to focus on the distance between moving object and query point at the time instant of boundary value. These chosen time instants are $t_0, t_1, t_2, \dots, t_j, \dots, t_{n-1}$, and t_n . $D_{q,o_i}(t_j)$ is calculated to obtain the maximal distance between the i^{th} moving object and query point at time t_j by using the method presented in the section 3.2. After calculating all of the maximal distances between K objects nearest to the query point at t_0 and query point at all chosen time instants, the maximal one is chosen to be the $D_{q,o}^{max}$, shown in Equ.4.1.1.

$$D_{q,o}^{max} = \max\{D_{q,o_i}(t_j)\} \quad (1 \leq i \leq K, 0 \leq j \leq n) \tag{Equ.4.1.1}$$

The D_{add}^{max} indicates an additional distance to the distance $D_{q,o}^{max}$ within the given time interval $[t_0, t_n]$. Taking the query point as a start point, if this point moves the distance $D_{q,o}^{max}$ via all the possible edges in the road network, it will terminate on a point of any of these possible edges. We take these terminated points on the possible edges as boundary points (denoted as P_{bound}). For an edge with P_{bound} and an object on this edge, if the minimal distance between this object and query point at t_0 is larger than $D_{q,o}^{max}$, and the object is moving with the maximal speed limit of this edge toward the P_{bound} within the time interval $[t_0, t_n]$, if this object can reach or pass through the P_{bound} , we need to consider this object in the CKNN query. Therefore, we need to add a distance to the $D_{q,o}^{max}$ to be the pruning distance, in case missing any possible object to be the KNN. If the number of P_{bound} is m , then the number of edges with boundary point is m , assume that the maximal speed limit of an edge with P_{bound} is denoted as SL_x ($1 \leq x \leq m$), we can calculate the additional distance for each edge as $SL_x \times (t_n - t_0)$. The maximal one of these calculated additional distances is chosen to be D_{add}^{max} , shown in Equ.4.1.2.

$$D_{add}^{max} = \max\{SL_x \times (t_n - t_0)\} \quad (1 \leq x \leq m) \tag{Equ.4.1.2}$$

The pruning distance $d_{pruning}$ is composed of $D_{q,o}^{max}$ and D_{add}^{max} . Furthermore, we can determine whether an object is considered in the CKNN query process or not by calculating the minimal distance between this object and query point within the time interval $[t_0, t_n]$, denoted as $d_{q,o}^{min}$. With the chosen time instants of $[t_0, t_n]$, we can calculate the minimal distance between the object and query point at a chosen time instant t_j , denoted as $d_{q,o}(t_j)$, by using the similar method presented in section 3.1. By calculating the minimal distances at all chosen time instants, the minimal one is chosen to be the $d_{q,o}^{min}$, as shown in Equ.4.1.3.

$$d_{q,o}^{min} = \min\{d_{q,o}(t_j)\} \quad (0 \leq j \leq n) \tag{Equ.4.1.3}$$

Firstly, for the K nearest objects to the query point at the start time t_0 , the distance $D_{q,o}^{max}$ is calculated with the chosen time instants from the given time interval $[t_0, t_n]$. Secondly, an additional distance D_{add}^{max} is calculated based on the boundary point P_{bound} which is determined by the distance $D_{q,o}^{max}$. Then, the pruning distance $d_{pruning}$ is calculated as $d_{pruning} = D_{q,o}^{max} + D_{add}^{max}$. Finally, whether an object o needs to be pruned or not is determined by comparing its minimal distance $d_{q,o}^{min}$ within the time interval $[t_0, t_n]$ with the pruning distance $d_{pruning}$. If $d_{q,o}^{min} \leq d_{pruning}$, the object can be a candidate for the CKNN query, otherwise, the object will be pruned.

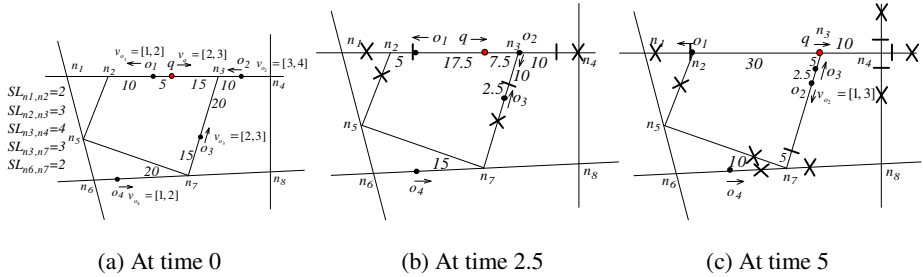


Fig. 2. Graph of a road network at different time points in MACKNN algorithm

We take an example to illustrate the pruning phase. As illustrated in Fig.2(a), there are four moving objects near query point q with uncertain velocity within the time interval $[0, 5]$ in a road network, denoted as o_1, o_2, o_3 , and o_4 , where the corresponding length of edges, the speed and direction of all objects and q are marked. Our task is to process continuously 2NN query over q within $[0, 5]$. At the start time 0, o_1 and o_2 are 2 nearest objects to q , and their network distance to q is 5 and 25 respectively. In the pruning phase, we need to scale down the object candidates that will be considered in the 2NN query process. Firstly, since moving object q, o_1, o_2 take 5s, 10s, 2.5s to reach intersection n_3 and n_2 of the road network with their maximal moving speed 3, 2, 4 respectively, and the smallest arrival time is 2.5 s, the time interval $[0, 5]$ is divided into two time subintervals: $[0, 2.5]$ and $[2.5, 5]$ by using our proposed time division method. Secondly, we can calculate the maximal distances between o_1, o_2 and q at time subinterval $[0, 2.5]$, as for time subinterval $[2.5, 5]$ we can handle in similar way.

These maximal distances are represented as $D_{q,o_1}(t)$, $D_{q,o_2}(t)$ and $D_{q,o_1}(2.5)=17.5$, $D_{q,o_2}(2.5)=7.5$. Thirdly, we can calculate the additional distance for each edge with boundary points. As shown in Fig. 4(b), there are three boundary points on the edges. Each boundary point is marked as a short line on the edge, such as the short line on the edge $[n_2, n_3]$. For each edge with the boundary point, we calculate the moving distance with the velocity of maximal speed limit of this edge within $[0,2.5]$. As shown in Fig.2 (b), the boundary points P_{bound} , marked with short vertical bars, are on edges $e(n_2,n_3)$, $e(n_3,n_4)$, and $e(n_3,n_7)$. Then $AD=\max\{3\times 2.5, 4\times 2.5, 3\times 2.5\}=10$, so $D_{pruning}=17.5+10=27.5$. The pruning distance ranges of query q are marked with black thick cross lines. Since there are only three moving objects o_1, o_2 and o_3 within the pruning range, the object candidate set $O_{candidate}$ is $\{o_1, o_2, o_3\}$. The other objects outside the pruning range are excluded, as they cannot be 2NN query results within the time interval $[0, 2.5]$.

4.2 Refining Phase

With the pruning phase, we can prune the objects which cannot be the KNN query results, and obtain a object candidate set $O_{candidate}=\{o_1, o_2\dots o_i\dots o_m\}$ ($m\geq K$) within the given time interval $[t_0, t_n]$. From the above candidate set, we generate a sorted set Os of m nearest neighbors. According to MSUO model, we specify whether each NN is moving away from or getting closer to query q using \uparrow or \downarrow symbols. if the moving degree of o_i to q need to further calculated, the moving state of object o_i with degree is represented as $\overset{dv(o_i,q)}{\uparrow}o_i$ or $\overset{dv(o_i,q)}{\downarrow}o_i$. For the above given example, the sorted candidate list Os is $\{\uparrow o_1, \downarrow o_2, \downarrow o_3\}$.

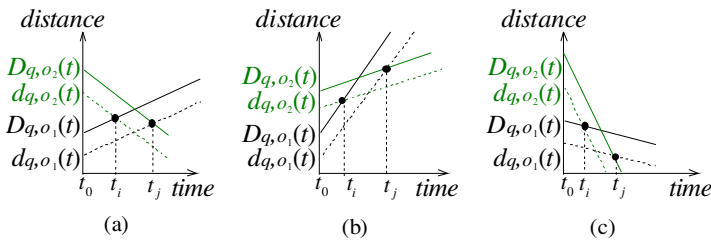


Fig. 3. Three cases about the sequence of two subsequent NNs changing

As the objects are moving continuously, the KNN query results may change during the given time interval. Thus we need to find the time point when the sequence of two subsequent nearest neighbors changes. Since it is difficult to obtain the exact changing time point, we just focus on getting the approximate changing time point. As shown in Fig.3, two subsequent nearest neighbors o_1 and o_2 , where o_1 is nearer to q than o_2 , could change their relative sequence in the sorted set Os in three different cases according to their different moving state, e.g. $\uparrow o_1$ and $\downarrow o_2$ (Fig.3 (a)), $\uparrow o_1$ and $\uparrow o_2$ (Fig.3 (b)), and $\downarrow o_1$ and $\downarrow o_2$ (Fig.3 (c)). We can observe that the sequence changing time points of o_1 and o_2 could be within $[t_i, t_j]$, where $D_{q,o_1}(t_i) = d_{q,o_2}(t_i)$ or $D_{q,o_2}(t_j) = d_{q,o_1}(t_j)$.

We still use the same example in section 4.1 to illustrate how to find the changing time points. From the pruning phase, we can get the sorted candidate list O_s ($\uparrow o_1$, $\downarrow o_2$, $\downarrow o_3$). According to the MSUO model, the sequence of o_1 and o_2 may change, while the sequence of o_2 and o_3 cannot change since the moving degree of o_2 is larger than o_3 , calculated as $\{\uparrow o_1, \downarrow o_2, \downarrow o_3\}$. Then we just need to determine the changing time point of o_1 and o_2 by calculating $D_{q,o_1}(t_i) = d_{q,o_2}(t_i)$ and $D_{q,o_2}(t_j) = d_{q,o_1}(t_j)$. By solving the equations, $5+(2+3)\times t_i=25-(3+4)\times t_i$ and $5+(1+2)\times t_j=25-(2+3)\times t_j$, we can get that $t_i=1.67$, and $t_j=2.5$. Therefore, the KNN result is $\langle [0,1.67], (o_1, o_2) \rangle$ and $\langle [1.67,2.5], \{o_1, o_2 \leftrightarrow o_1\} \rangle$. For the query time interval $[2.5, 5]$, the KNN query can be processed in the similar way. Assumed that o_2 reach the node n_3 and continue to move toward node n_7 and its velocity interval is updated to be $[1,3]$, With the pruning phase, the pruning distance is $30+10=40$, and the candidate list with moving state is $\{\downarrow o_2, \uparrow o_1, \downarrow o_3\}$, clearly, the sequence of o_1 and o_3 may change, then we get that $t_i=2.72$, and $t_j=4.3$ by calculation, so the KNN result is $\langle [2.5,2.72], (o_2, o_1) \rangle$ and $\langle [2.72,4.3], \{o_2, o_3 \leftrightarrow o_1\} \rangle$, next, for the query time interval $[4.3, 5]$, the candidate list with moving state is $\{\downarrow o_2, \downarrow o_3, \uparrow o_1\}$. According to the MSUO model, the sequence of o_2 and o_3 may change. The changing time point is calculated as 5.613, which is larger than 5, then the sequence of o_2 and o_3 may not change within $[2.5, 5]$. The MACKNN processing result of example in Fig. 2 is shown in Table 1.

Table 1. Processing Results of MACKNN query for the example of Fig.2

KNN results	sorted candidate list	replace objects	minimal and maximal time
$\langle [0,1.67], \{o_1, o_2\} \rangle$ $\langle [1.67,2.5], \{o_1, o_2 \leftrightarrow o_1\} \rangle$	$\{\uparrow o_1, \downarrow o_2, \downarrow o_3\}$	$o_1 \leftrightarrow o_2$	$t_i=1.67, t_j=2.5$
$\langle [2.5,2.72], \{o_2, o_1\} \rangle$ $\langle [2.72,4.3], \{o_2, o_3 \leftrightarrow o_1\} \rangle$	$\{\downarrow o_2, \uparrow o_1, \downarrow o_3\}$	$o_1 \leftrightarrow o_3$	$t_i=2.72, t_j=4.3$
$\langle [4.3,5], \{o_2, o_3\} \rangle$	$\{\downarrow o_2, \downarrow o_3, \uparrow o_1\}$	$o_2 \leftrightarrow o_3$	$t_i=5.625 > 5,$ $t_j=6.67, \text{terminal}$

5 Experimental Evaluation

Our experiments are implemented in visual C++, running on a PC with 1.86GHZ Pentium dual processor and 1GB memory. For our experiments, we consider an Oldenburg data set consisting of 7,035 nodes [26], the moving objects are generated using the generator proposed in [27], and each object starts at a randomly selected position in the range of interest. Subsequently, objects are uniformly distributed and the speed vector of each object is randomly generated ranging from 0 to 5. When an object o reaches node n along its moving direction in a road network, the next edge on which o moves is randomly chosen from the edges connecting n . In all the following figures, the running time (seconds) or precision is shown in a logarithmic scale.

There are 200 query objects in our system. The speeds of query objects are the same as that of moving objects mentioned above. Similarly, the next edge that the query object moves on is randomly selected once it reaches a network node. We use

up to 100,000 moving objects for the experiment, the default values of the number of nearest neighbors requested by each query object (i.e., K) is 10 and the length of query time interval is 50 time units. In our experiments, we compare precise of the IMA algorithm proposed by Mouratidis *et al.* [19] and refining cost of the CKNN algorithm proposed by Y.K. Huang *et al.* [20] with MACKNN algorithm, respectively.

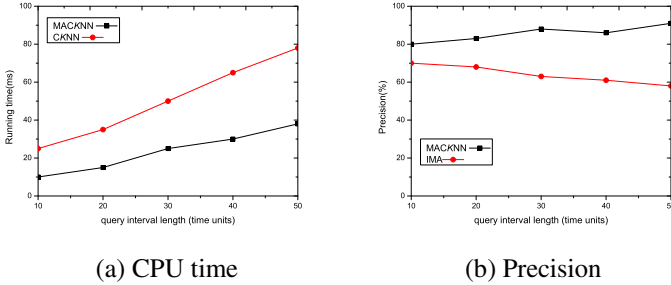


Fig. 4. The performance and precision on query interval length

In figure 4(a), we compare the CPU time cost of MACKNN algorithm with those of CKNN algorithm. From the figure, we can easily observe that the CPU time cost of MACKNN algorithm is significantly lower than that of CKNN algorithm. Here the precision refers to the percentage of real KNNs that are retrieved by executing the MACKNN and IMA algorithms. We use a snapshot approach to compare the current locations of all moving objects to find the K nearest neighbors of the query object at each time unit. It returns the most correct KNN set. Fig.4 (b) shows that the precise of MACKNN algorithm is higher than that of IMA algorithm at each time instant.

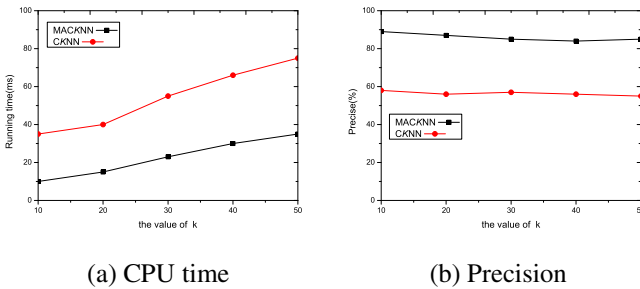


Fig. 5. The performance and precision on the number of K

We compare the performance of MACKNN, CKNN and IMA by varying the value of K from 10 to 50. Figure 5(a) shows that the CPU overheads for these two algorithms grow when k increases and MACKNN outperforms its competitor CKNN in all cases. Figure 5(b) shows that the precision of MACKNN decreases slightly as K increases, but it is above 80 % in all cases. While the precision of IMA is at best about 60%. This demonstrates that our approximate approach outperforms the IMA and CKNN algorithm.

6 Conclusions

In order to process the $CKNN$ queries over the moving objects with uncertain velocity in road networks more efficient, a novel $MACKNN$ technique is proposed to process the approximate $CKNN$ queries over the moving objects. A Moving State of Uncertain Object (MSUO) model is firstly presented to determine the moving state of object by calculating the moving degree of moving object to the query point. With the help of MSUO model, the $MACKNN$ technique is composed of pruning phase and refining phase. The pruning phase is used to prune non-qualifying objects by computing the distance between moving objects and query point q within given time subinterval. The refining phase is designed to determine the subinterval where the query result could be certain. Experimental results show that our approach outperforms $CKNN$ and IMA in terms of running time and precise rate, respectively, while retaining a certain accuracy level.

Acknowledgement. This work was supported by National Natural Science Fund of China under grant 60873030, Doctoral Fund of Ministry of Education of China under grant 20090142110023, the Science and Technology Research Project of Department of Education of Hubei Province under grant Q20102807 and the Teaching Research Project of XianNing University under grant J09025.

References

1. Zhang, J., Zhu, M., Papadias, D., Tao, Y., Lee, D.L.: Location-based spatial queries. In: Proceedings of the SIGMOD Conference, San Diego, CA, pp. 443–454 (2003)
2. de Almedia, V.T.: Towards optimal continuous nearest neighbor queries in spatial databases. In: Proceedings of ACM GIS (November 10–11, 2006)
3. Benetis, R., Jensen, C.S., Karciauskas, G., Saltenis, S.: Nearest neighbor and reverse nearest neighbor queries for moving objects. In: Proceedings of the International Database Engineering and Applications Symposium, Canada, July 17–19 (2002)
4. Benetis, R., Jensen, C.S., Karciauskas, G., Saltenis, S.: Nearest neighbor and reverse nearest neighbor queries for moving objects. *VLDB Journal* 15(3), 229–249 (2006)
5. Kolahdouzan, M.R., Shahabi, C.: Alternative solutions for continuous K nearest neighbor in spatial network databases. *GeoInformatica* 9(4), 321–341 (2004)
6. Kolahdouzan, M.R., Shahabi, C.: Continuous k nearest neighbor queries in spatial network databases. In: Proceedings of the Spatio-Temporal Databases Management (STDBM), Toronto, Canada (August 30, 2004)
7. Raptopoulou, K., Papadopoulos, A.N., Manolopoulos, Y.: Fast nearest-neighbor query processing in moving-object databases. *GeoInformatica* 7(2), 113–137 (2003)
8. Tao, Y., Papadias, D.: Time parameterized queries in spatio-temporal databases. In: Proceedings of the ACM SIGMOD, Madison, Wisconsin (2002)
9. Tao, Y., Papadias, D., Shen, Q.: Continuous nearest neighbor search. In: International Conference on Very Large Data Bases, Hong Kong, China (August 20–23, 2002)
10. Xiong, X., Mokbel, M.F., Aref, W.G.: SEA-CNN: Scalable Processing of Continuous K -Nearest Neighbor Queries in Spatio-temporal Databases. In: Proceedings of the 21st International Conference on Data Engineering (ICDE), Tokyo, Japan, pp. 643–654 (2005)

11. Yu, X., Pu, K.Q., Koudas, N.: Monitoring k-Nearest Neighbor Queries over Moving Objects. In: Proceedings of the 21st International Conference on Data Engineering (ICDE), Tokyo, Japan, pp. 631–642 (2005)
12. Hu, H., Xu, J., Lee, D.: A Generic Framework for Monitoring Continuous Spatial Queries over Moving Objects. In: Proceedings of the SIGMOD Conference, Paris, France, pp. 479–490 (2005)
13. Papadias, D., Zhang, J., Mamoulis, N., Tao, Y.: Query processing in spatial network databases. In: VLDB (2003)
14. Kolahdouzan, M.R., Shahabi, C.: Voronoi-based K nearest neighbor search for spatial network databases. In: VLDB, pp. 840–851 (2004)
15. Mouratidis, K., Hadjieleftheriou, M., Papadias, D.: Conceptual partitioning: An efficient method for continuous nearest neighbor monitoring. In: SIGMOD (2005)
16. Li, G., Fan, P., Li, Y., Du, J.: An Efficient Technique for Continuous K-Nearest Neighbor Query Processing on Moving Objects in a Road Network. In: CIT 2010 (2010)
17. Jensen, C.S., Kolar, J., Pedersen, T.B., Timko, I.: Nearest neighbor queries in road networks. In: Proceedings of the ACM GIS, New Orleans, Louisiana, USA (November 7-8, 2003)
18. Cho, H.-J., Chung, C.-W.: An efficient and scalable approach to cnn queries in a road network. In: Proceedings of the International Conference on Very Large Databases, Trondheim, Norway (2005)
19. Mouratidis, K., Yiu, M.L., Papadias, D., Mamoulis, N.: Continuous nearest neighbor monitoring in road networks. In: Proceedings of the International Conference on Very Large Data Bases, Seoul, Korea (September 12-15, 2006)
20. Huang, Y.K., Chen, Z.W., Lee, C.: Continuous K-Nearest Neighbor Query over Moving Objects in Road Networks. In: Li, Q., Feng, L., Pei, J., Wang, S.X., Zhou, X., Zhu, Q.-M. (eds.) APWeb/WAIM 2009. LNCS, vol. 5446, pp. 27–38. Springer, Heidelberg (2009)
21. Arya, S., Mount, D.M.: Approximate Nearest Neighbor Queries in Fixed Dimensions. In: SODA, pp. 535–544 (2005)
22. Ferhatosmanoglu, H., Tuncel, E.: Vector Approximation based Indexing for Non-Uniform High Dimensional Data Sets. In: Proceedings of the Ninth International Conference on Information and Knowledge Management (2000)
23. Berrani, S.-A., Amsaleg, L., Gros, P.: Approximate Searches: K Neighbors+ Precision. In: Proceedings of the Twelfth International Conference on Information and Knowledge Management (2003)
24. Hsueh, Y.-L., Zimmermann, R., Yang, M.-H.: Approximate continuous K nearest neighbor queries for continuous moving objects with pre-defined paths. In: Akoka, J., Liddle, S.W., Song, I.-Y., Bertolotto, M., Comyn-Wattiau, I., van den Heuvel, W.-J., Kolp, M., Trujillo, J., Kop, C., Mayr, H.C. (eds.) ER Workshops 2005. LNCS, vol. 3770, pp. 270–279. Springer, Heidelberg (2005)
25. Sengupta, A., Pal, T.K.: On comparing interval numbers. *European Journal of Operational Research* 127, 28–43 (2000)
26. US Bureau of the Census, Technical Documentation. TIGER/Line Files (1995)
27. Brinkhoff, T.: A framework for generating network-based moving objects. *GeoInformatica* 6(2), 153–180 (2002)

Parallel Detection of Temporal Events from Streaming Data

Hao Wang¹, Ling Feng¹, and Wenwei Xue²

¹ Dept. of Computer Science & Technology, Tsinghua University, Beijing, China
wanghaomails@gmail.com, fengling@tsinghua.edu.cn

² Nokia Research Center, Beijing, China
wayne.xue@nokia.com

Abstract. Advanced applications of sensors, network traffic, and financial markets have produced massive, continuous, and time-ordered data streams, calling for high-performance stream querying and event detection techniques. Beyond the widely adopted *sequence* operator in current data stream management systems, as well as inspired by the great work developed in temporal logic and active database fields, this paper presents a rich set of temporal operators on events, with an emphasis on the temporal properties and relative temporal relationships of events. We outline three temporal operators on unary events (*Within*, *Last*, and *Periodic*), and four ones on binary events (*Concur*, *Sequence*, *Overlap* and *During*). We employ two stream partitioning strategies, i.e., *time-driven* and *task-driven*, for parallel processing of the temporal operators. Our analysis and experimental results with both synthetic and real-data show that the better partitioning scheme in terms of system throughput is the one which can produce balanced data workload and less data duplication among the processing nodes.

1 Introduction

Stream data is flooding in real life. GPS-devices, body and environment sensors, videos, and financial market generate enormous amount of streams every day. Stream data is usually affiliated with an explicit or implicit time-stamp, exhibiting continuity and temporality characteristics. From raw stream data, high-level meaningful events of interest to upper applications can be detected. Most stream processing tasks today are related to continuous event monitoring and detection associated with time.

To facilitate event detection, we first need to establish a way to describe events and their temporal properties. Expressing the temporal occurrence of an event by means of a *precise* and *absolute* time point or time interval in a time line is one way. Beyond that, taking into account that a lot of temporal knowledge in practice is *imprecise* and *relative*, and has little relation to an absolute time point or interval, the temporal references of events could also be implicitly presented by the description of how one event is related to others temporal properties of events. S_1 : “*My colleague and I arrived at the meeting room at exactly the same*

time” (two temporally concurrent events) S_2 : “*I arrived at the meeting room 10 minutes earlier than my colleague*” (two temporally sequential events) S_3 : “*A car accident happened while I was jogging in the street yesterday*” (two temporally containing events) S_4 : “*When I was at home yesterday, a fire went on, and I had to run out of the burning house immediately*” (two temporally overlapping events between my being at home and the fire).

Researchers from the active and stream data management fields have developed a bundle of operators like *sequence*, *conjunction*, *disjunction*, *negation*, *kleene closure*, *periodic*, and *aperiodic* over atomic or composite events [2,7,10]. Beyond these, we are aiming at a richer representation of temporal properties and relative temporal relationships of events. Inspired by the great work developed in temporal logic [3] and active database fields [2], we investigate seven temporal operators upon event(s) beyond the widely adopted temporal *sequence* operator in the current data stream management systems. This set of temporal operators consists of three operators on a single unary event, i.e., *Within* (temporal occurrence of an event), *Last* (temporal duration of an event), and *Periodic* (periodic occurrence of an event with/without frequency), and four operators on two binary events, i.e., *Concur* (two concurrent events), *Sequence* (two sequential events with/without temporal distance), *Overlap* (two overlapped events with/without overlapping interval), and *During* (two temporally encapsulation events). These operators are powerful enough to describe any temporal property and temporal relationship of events.

For efficient processing of these event operators over a large volume of stream data, we further pursue the parallel technology and partition streams into fragments, which are then allocated to different computer nodes to process. We employ two stream partitioning strategies, i.e., *time-driven* and *content-driven*. The first *time-driven* scheme employs a sliding-window approach. It chops a stream into temporally consecutive fragments according to the time-stamps of stream records. The temporal length of each fragment (the size of sliding windows) remains the same. The second *task-driven* partition scheme decomposes an event detection task (i.e., processing of a temporal operator) into several independent sub-tasks. Each sub-task is executed by a computer node and upon a fragment of stream records. We evaluate their performance on both synthetic and real-life GPS. Our experimental results show the throughout of the overall system is influenced substantially by two factors. The first is the total amount of stream records to be processed by the computer nodes. More data overlaps among fragments incur more time cost, and thus less throughput. The second influential factor is even distribution of stream data among nodes, as we count the total elapsed time when the final result is out. Thus, the better partition scheme is the one which causes less data duplication and even data distribution among nodes.

The remainder of the paper is organized as follows. We describe some closely related work on stream processing in Section 2. A set of temporal operators on events are presented in Section 3. We employ two stream partitioning strategies

for parallel event detection from streaming data in Section 4, and report our performance study in Section 5. We conclude the paper in Section 6.

2 Related Work

There are four typical types of stream data processing languages in the literature, which are *relation*-based, *object*-based, *procedure*-based, and *event algebra*-based. *Relation-based languages* like CQL [4], StreaQuel [19], and GSQL [9] extend the SQL language to query timestamped relations with the support of windows and ordering [13] [18]. *Object-based languages* also resemble SQL, but use abstract data types and associated methods for processing stream data [1, 14]. *Procedural-based languages* construct queries by defining data flows through various operators [8]. *Event algebra-based languages* such as CEDR [6] and [5, 2, 7] integrate knowledge from active and streaming data management fields for complex event specification and detection.

Parallelism magnifies the power of individual computers and offers high computing performance. Early technologies of parallel data management are outlined in the good survey paper [17]. *Control flow* and *data flow* are two basic types of parallel mechanisms [11, 21], and data partitioning is mostly based on *round-robin*, *hash*, *range*, or *hybrid-range* [12]. [17, 15] introduces two stream partitioning strategies, i.e., *window split* and *window distribute*. [16] proposes to dynamically choose the optimal partitioning scheme by analyzing the query set.

Different from the previous work, this study aims to study a richer set of unary and binary temporal operators over streams, and parallel optimization and execution of these temporal operators.

3 Temporal Operators on Events

Detection of events are based on time-stamped data streams. A data stream is a sequence of continuous data records of the same data structure. Taking position tracking for examples, a GPS data stream S_{GPS} has a schema with three attributes $S_{GPS} = (t, id, xyPosi)$. Each record rec in S_{GPS} details a physical coordination $xyPosi$ of a person/device of identity id at time-stamp t .

There is no doubt that temporality is one of the most characteristics that distinguish data streams from other kind of data. To facilitate event detection from time-stamped streams, we first present the notion of time and event, followed by a set of fine-grained temporal operators, which are used to describe temporal properties and relationships of events.

Time. We distinguish the notion of *time point* and *time interval*. “*time-point-format* = *clock-time-format*, *time-interval-format* = [*clock-time-format*, *clock-time-format*], *clock-time-format* = [Year:Month:Week:Day:] Hour: Minute: Second”. where [Year:Month:Week:Day:] is optional. The domain of *time point* t , denoted as $Dom(t)$, is assumed to be discrete with an equal-distance. A partial temporal order \leq_t is used to declare two time points t_1 and t_2 in sequence. $t_1 <_t (>_t) t_2$

states that t_1 is earlier (later) than t_2 in the time line. ($t_1 =_t t_2$) indicates two exactly the same time points. \leq_t (\geq_t) denotes not earlier (later) than temporal relationship.

A *time interval* $[t^-, t^+]$ has a start time point and an ending time point, where $(t^-, t^+ \in Dom(t))$ and $(t^- \leq_t t^+)$. The *time length* of $[t^-, t^+]$ is the time difference of t^+ and t^- , represented as $(t^+ -_t t^-)$. When $(t^- =_t t^+)$, the time interval degrades to a time point, becoming a special case of time interval with a zero time length.

We call a time interval $[t'^-, t'^+]$ a *sub-interval* of $[t^-, t^+]$, denoted as $[t'^-, t'^+] \subseteq_t [t^-, t^+]$, if and only if $(t'^- \geq_t t^-)$ and $(t'^+ \leq_t t^+)$. A time point t *belongs to* time interval $[t^-, t^+]$, denoted as $t \in_t [t^-, t^+]$, if and only if t falls in the time interval, that is, $(t^- \leq_t t) \wedge (t \leq_t t^+)$.

Event. An event is an atomic (happening completely or not at all) occurrence in the world [2]. An event can be either an *instantaneous* event (e.g., turning off the light in a room, entering the room, etc.) or a *persistent* one (e.g., staying in a shop to buy something, having a traffic jam, etc.). The former may last for a very short moment, while the later may take a time period, represented using a time interval. From a discrete collection/sampling point of view, an instantaneous event occurs at a time point. Taking instantaneous events' moment nature into account, and meanwhile for uniformly describing temporality of instantaneous and persistent events, we affiliate a time interval $[t^-, t^+]$ ($t^- \leq_t t^+$) to signify the happening period of an event E . We use $E.att$ to denote attribute att of event E .

The occurrence of event E from one or more data streams S_1, S_2, \dots, S_n can be detected via an *event detection expression*, taking the form of a conjunction of disjunctions of boolean predicates on event's attributes. $Detect(E, S_1, S_2, \dots, S_n) = (q_{1,1} \vee \dots \vee q_{1,n1}) \wedge \dots \wedge (q_{m,1} \vee \dots \vee q_{m,nm})$, where $q_{i,j}$ is an either positive or negative predicate literal. A TRUE value indicates the occurrence of event E , and FALSE, otherwise.

Temporal Operators on Events. To richly express temporal properties and relationships of events, we present the following set of temporal operators on events. This work draws inspiration from Allen's excellent work on temporal relationships in the domain of temporal logic [3].

1) $Within(E, [t'^-, t'^+]) = TRUE$, if and only if $([E.t^-, E.t^+] \subseteq_t [t'^-, t'^+])$. The *Within* operator examines whether the happening time interval of event E is within a larger time scope $[t'^-, t'^+]$. The event that "someone entered the market at mid-night during 6am and 11pm" can be expressed as: $Detect(E, SGPS) = Near(E.xyPosi, "market") \wedge Within(E, [6 : 0 : 0, 23 : 0 : 0])$.

2) $Last(E, \Delta t) = TRUE$, if and only if $(E.t^+ -_t E.t^- =_t \Delta t)$. The *Last* operator examines whether event E lasts for a certain period of time length Δt . For simplicity, we use $|E|$ to denote the time length of event E , i.e., $|E| = (E.t^+ -_t E.t^-)$. The event that "person of id 1 has been at the ABCShop for 1 hour" can

be detected via the event expression: $Detect(E, S_{GPS}) = EQ(E.id, 1) \wedge Near(E.xyPosi, "ABCShop") \wedge Last(E_1, 1 : 0 : 0)$.

3) $\underline{Periodic}(E, \{n, \}\{\Delta t, \}[t^-, t^+])=TRUE$, if and only if event E occurs n times with a frequency Δt within time interval $[t^-, t^+]$, where $([E.t^-, E.t^+] \subseteq_t [t^-, t^+])$ and n is a positive integer. $\{n\}$ and $\{\Delta t\}$ are optional. The *Periodic* operator can express such event that "person of id 2 went to the ABCShop four times from 8am till 12am", whose event detection expression is: $Detect(E, S_{GPS}) = EQ(E.id, 2) \wedge Near(E.xyPosi, "ABCShop") \wedge Periodic(E, 4, [8 : 0 : 0, 12 : 0 : 0])$.

4) $\underline{Concur}(E_1, E_2)=TRUE$, if and only if $(E_1.t^- =_t E_2.t^-) \wedge (E_1.t^+ =_t E_2.t^+)$. The *Concur* operator indicates that the two events happen concurrently during the same time interval. Statement that "person of id 1 and person of id 2 went together to the cinema to watch the movie" can be detected via the expression: $Detect(E_1, E_2, S_{GPS}) = EQ(E_1.id, 1) \wedge Near(E_1.xyPosi, "cinema") \wedge EQ(E_2.id, 2) \wedge Near(E_2.xyPosi, "cinema") \wedge Concur(E_1, E_2)$.

5) $\underline{Sequence}(E_1, E_2, \{\Delta t\})=TRUE$, if and only if $(E_1.t^+ \leq_t E_2.t^-) \wedge (E_2.t^- -_t E_1.t^+ =_t \Delta t)$. The *Sequence* operator is similar to the traditional *sequence* operator, except that it indicates specifically that event E_2 happens Δt time later than event E_1 . The omission of Δt signifies any value not less than 0:0:0 of time length. When $\Delta t =_t 0 : 0 : 0$, the *Sequence* operator states a temporally joining relationship that one event is followed exactly by another event without any temporal gap in between. Statement that "a person went to a shop first and 40 minutes later entered the cinema" describes the sequential relationship between two events. $Detect(E_1, E_2, S_{GPS}) = EQ(E_1.id, E_2.id) \wedge Near(E_1.xyPosi, "shop") \wedge Near(E_2.xyPosi, "cinema") \wedge Sequence(E_1, E_2, 0 : 40 : 0)$.

6) $\underline{During}(E_1, E_2)=TRUE$, if and only if $((E_1.t^- >_t E_2.t^-) \wedge (E_1.t^+ \leq_t E_2.t^-)) \vee ((E_1.t^- \geq_t E_2.t^-) \wedge (E_1.t^+ <_t E_2.t^-))$. The *During* operator indicates that event E_1 happens within the time interval of E_2 . This is different from *Within*, which states an absolute time scope where an event happens. Statement like "person of id 1 arrived at the campus when person of id 2 was at the lecture hall to give a lecture" can be represented as $Detect(E_1, E_2, S_{GPS}) = EQ(E_1.id, 1) \wedge Near(E_1.xyPosi, "campus") \wedge EQ(E_2.id, 2) \wedge Near(E_2.xyPosi, "lectureHall") \wedge During(E_1, E_2)$.

7) $\underline{Overlap}(E_1, E_2, \{\Delta t\})=TRUE$, if and only if $(E_1.t^- <_t E_2.t^-) \wedge (E_1.t^+ <_t E_2.t^+) \wedge (E_1.t^+ -_t E_2.t^- =_t \Delta t)$. The *Overlap* operator indicates that event E_1 happens earlier than event E_2 , and the two events have an intersectional time interval of length Δt . The omission of Δt signifies any value not less than 0:0:0 of time length. Statement that "when person of id 1 saw person of id 2 entered the building, person of id 1 ran out of the building" exhibits the temporal *Overlap* relationship between two events associated with person of id 1 and 2, respectively. $Detect(E_1, E_2, S_{GPS}) = EQ(E_1.id, 1) \wedge Near(E_1.xyPosi, "house") \wedge EQ(E_2.id, 2) \wedge Near(E_2.xyPosi, "house") \wedge Overlap(E_1, E_2)$.

4 Parallel Processing of Temporal Operators

For high-performance process of the above temporal operators upon massive stream data, we adopt a divide-and-conquer strategy and magnify the processing with parallelism.

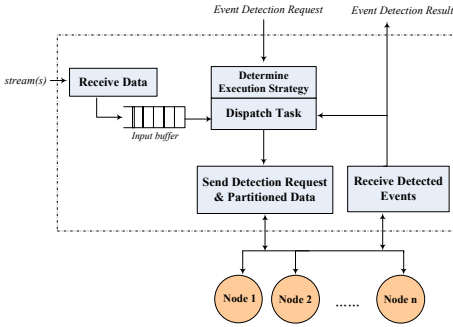


Fig. 1. Parallel event detection

Figure 1 outlines our parallel processing framework. Incoming stream data is first put into an input buffer by the *data receiving* module. The *task dispatching* module breaks down the incoming stream data into smaller pieces according to the user's event detection request, as well as the stream partition scheme returned from the *strategy determination module*. The *request-and-data sending* module then distributes different workload (i.e., detection request and partitioned stream data) to different computer nodes to process. The *events' receiving* module

finally collects and combines events detected by different nodes and returns them to the user.

4.1 Stream Partitioning

To ensure the stream does not undergo semantic change during partitioning, we enforce the following two correctness criteria during partitioning. 1) *Completeness*. If a stream S is partitioned into several pieces S_1, S_2, \dots, S_n , each record in S must appear in one or more of S_i . This property ensure data lossless decomposition, which is identical to lossless fragmentation in distributed databases [20]. 2) *Integrity*. If an event E can be detected from stream S , it can also be detected from one and only one partitioned piece of S_i . This property ensures each processing node to obtain an integral set of stream records in order to detect an event independently. We examine two types of data partitioning schemes, i.e., *time-driven* and *task-driven*, for parallel detection of events from streams.

I. Time-Driven Stream Partition

A stream is partitioned into smaller fragments based on time-stamps of stream records and temporal properties of events to be detected. Each fragment contains a temporally consecutive list of stream records. The temporal length of a fragment, $|frag|$, is thus the same.

Since the paper focuses on temporal operators, we detail our time-driven partitioning for different types of operators in the following diagrams. Stream fragments are allocated to computer nodes in a round-robin manner.

A fragment may contain η number of pieces, whose spanning time length varies from the temporal operators, as illustrated in Figure 4.1. For $Within(E, [t^-, t^+])$,

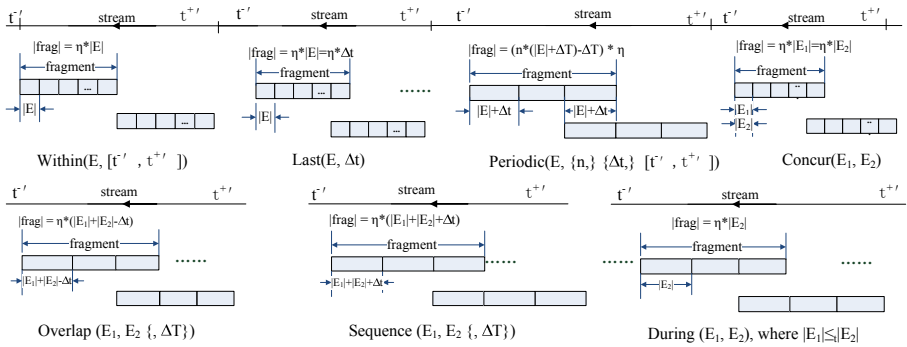


Fig. 2. Time-driven partition for operators

each piece has a time length of event E , i.e., $|E|$. If $|E|$ is unknown from the detection requirements, we set it to the time distance between every two stream records, i.e., the sampling frequency when acquiring stream data records. No overlap exists in this case. For $Overlap(E_1, E_2, \Delta t)$, each piece has time length $|E_1| + |E_2| - \Delta t$.

A fragment may contain η number of pieces. Except the *Within* operator, some data duplication exists among every two consecutive fragments. This is for the sake of complete detection of all events, and the length of duplication is equal to the piece length.

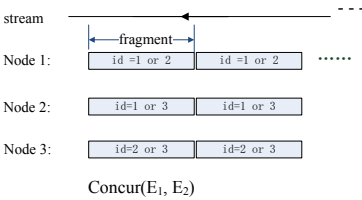


Fig. 3. Task-driven partition for operator *Concur* based on *id*

II. Task-Driven Stream Partitioning

The *task-driven* partition is to decompose an event detection task into a few sub-tasks. Each sub-task is executed by a computer node independently. The results from all the computer nodes are later combined to have the final result. Data needed for each sub-task will be allocated to the corresponding node for processing. For instance, to detect an event

that “*someone appeared at a park for 3 hours*” from a GPS stream, we can divide this detection task into several sub-tasks based on person’s *id*. In this case, we can assign node 1 to process stream records related to person of *id* 1, assign node 2 to process records related to person of *id*=2, node 3 to person of *id* 3, and so on. In this case, no fragment overlaps exist. However, in another example, when checking “*whether two persons stayed in the same place from the beginning to the end*”, and assuming the stream records three persons’ GPS data, such a partition schema decomposes the detection into three sub-tasks, and assigns node 1, 2, and 3 to process $Concur(E_1^1, E_2^2)$, $Concur(E_1^1, E_2^3)$, and $Concur(E_2^1, E_2^3)$, respectively, where E_1^i ($1 \leq i \leq 3$) represents the event subject is person of *id* = i . Inevitably, fragments detailing person of *id*=1 are distributed to node 1 and 2, and fragments detailing person of *id*=2 are to node 1 and 3,

and fragments detailing person of $id=3$ are to node 2 and 3 in Figure 3. High data duplication exists in this situation.

4.2 Analysis of The Two Partitioning Strategies

Different partitioning strategies lead to different stream workload and duplication for different nodes to process. Table 1 gives the parameters and their meanings.

Table 1. Parameters and Meanings

Para	Meaning	Para	Meaning
s	sampling rate	η	number of pieces per fragment
μ_t	elapsed time period of a piece	α	number of ids
n	number of nodes	γ_t	elapsed time period

We define the *imbalance degree* of stream workload in the system as the maximum difference in terms of the number of stream records between two processing nodes. That is,

$$\begin{aligned}
 ImB_{time} &= \begin{cases} \frac{\eta * \mu_t * s - (1 - \frac{\gamma_t * s}{\eta * \mu_t * s} - \lfloor \frac{\gamma_t * s}{\eta * \mu_t * s} \rfloor)}{r} & \text{if } (\frac{\gamma_t * s}{\{(\eta - 1) | \eta\} * \mu_t * s} \bmod n) \geq 1 \\ \frac{(\frac{\gamma_t * s}{\eta * \mu_t * s} - \lfloor \frac{\gamma_t * s}{\eta * \mu_t * s} \rfloor) * \eta * \mu_t * s}{\gamma_t} & \text{if } (\frac{\gamma_t * s}{\{(\eta - 1) | \eta\} * \mu_t * s} \bmod n) < 1 \end{cases} \\
 ImB_{task} &= \begin{cases} \frac{1}{C_\alpha^{2|1}} & \text{if } (C_\alpha^{2|1} \bmod n) \neq 0 \\ 0 & \text{if } (C_\alpha^{2|1} \bmod n) = 0 \\ \text{if unary operators } C_\alpha^1, \text{ otherwise } C_\alpha^2 \end{cases}
 \end{aligned}$$

For ImB_{time} , when data duplication across two consecutive fragments exists, the denominator in $(\frac{\gamma}{\{(\eta - 1) | \eta\} * \mu_t * s} \bmod n)$ takes $(\eta - 1) * \mu_t * s$, otherwise, it takes $\eta * \mu_t * s$. For ImB_{task} , when sub-tasks are dispatched to processing nodes equally, the imbalance degree is 0, otherwise, $\frac{1}{C_\alpha^{2|1}}$ (number of subtasks in task-driven partition).

$$\begin{aligned}
 Dup_{time} &= \begin{cases} \frac{1}{(\eta - 1)} - \frac{\mu_t * s}{\gamma} & \text{if data duplication exists} \\ 0 & \text{if otherwise} \end{cases} \\
 Dup_{task} &= \begin{cases} n & \text{for binary operators} \\ 0 & \text{for unary operators} \end{cases}
 \end{aligned}$$

We further define the *data duplication degree* among the processing nodes as the ratio of total number of duplication records versus the total number of arriving records in an elapsed time period. It describes the redundant workload of the system. Apparently, the smaller the above two indicators are, the better the throughput is.

We can select an appropriate partitioning strategy according to their imbalance and duplication degrees. Considering the overwhelming amount of streaming data to be processed, the influence of duplication is greater than data imbalance. Hence, we first examine their *duplication* degrees, followed by *imbalance* degrees.

In addition, for the time-driven partition, parameter η determines the temporal length of a fragment, as shown in Figure 4.1. Appropriate setting of η leads to good performance. When duplication degree Dup_{time} is zero, we set η value which can produce the smallest $ImB_{time}(\eta)$. When duplication is not zero, we choose η which can have the smallest $Dup_{time}(\eta)$. We prove this in our experiment.

5 Performance Study

We evaluate the performance of the two partitioning methods, implemented using C++ and MPICH2 parallel programming tools. System performance is measured by throughput, i.e., within a period of time, how many stream records are processed.

5.1 Experiments on Synthetic Stream Data

We study the scalability of the two partitioning methods on a synthetic GPS-like stream of scheme $S_{GPS} = (t, id, xyPosi)$. For each id , we simulate its GPS movement in four possible directions (i.e., east, south, west, or north), and then randomly assign a distance from its previous physical location to get the current $xyPosi$. The sampling of GPS is 1 record per second for each id ($s = 1 \text{ rec/sec}$), and the number of ids considered varies from 10 to 50. Experiment 1 and 2 examine the system throughput under different elapsed time periods and total id numbers, respectively. Experiment 3 examines the behavior of the time-driven partitioning strategy under different η settings.

Experiment 1: Effect of the Elapsed Time Period

The parameter settings in Experiment 1 are $(\eta, \alpha, n) = (3, 25, 3)$ and γ_t is set to 400, 800, 1200, 1800, 2000 seconds. Figure 4 shows the behavior of the two stream partitioning strategies. It is interesting to note that when processing binary-event operators (*Concur*, *Sequence*, *Overlap*, and *During*), the time-driven partitioning strategy performs consistently better than the task-driven strategy. This is because the task-partitioning strategy has to allocate a large volume of stream records to more than one processing node to ensure partition integrity. For instance, for *Overlap*, $Dup_{task} = 300\% > Dup_{time} = 48.8\%$, so time-driven partition is better than task-driven partition.

However, when processing the two unary-event operators (*Last* and *Periodic*), as each node is responsible for detecting events of certain ids without id duplication among nodes, there is no data overlap under the task-driven partitioning. It thus achieves a higher throughput than the time-driven partitioning which has data overlap among every two nodes. For example, in *Last*, $Dup_{task} = 0\% < Dup_{time} = 48.8\%$, so task-driven partition is better. Surprisingly, unary-event operator *Within* in Figure 4 gives a contradictory result. Because both

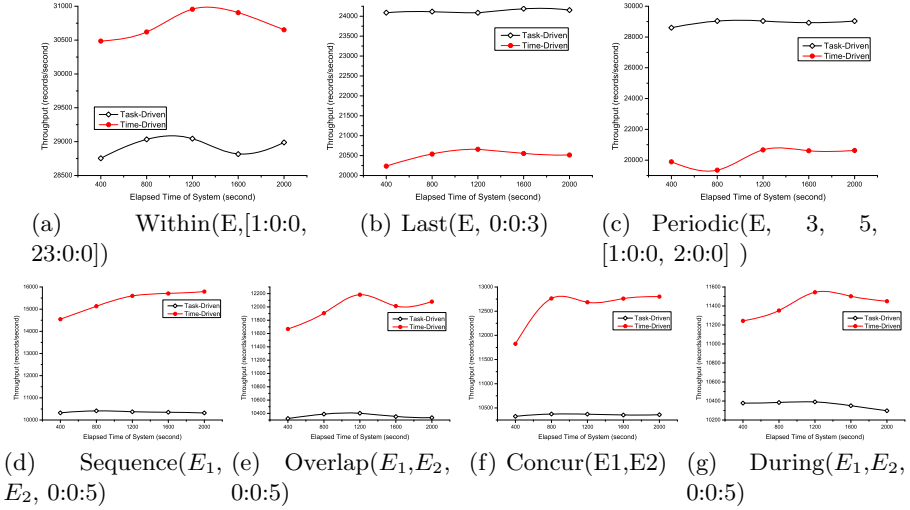


Fig. 4. Exp-1: Elapsed time vs. throughput

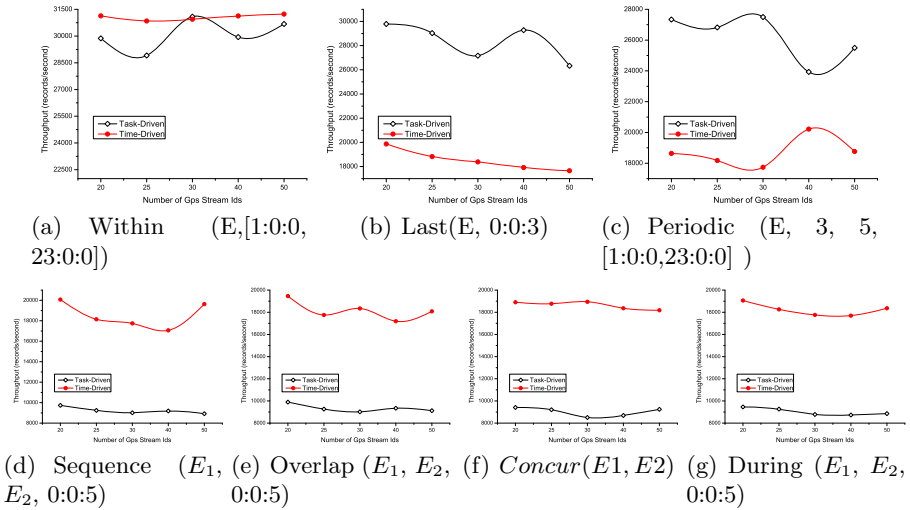


Fig. 5. Exp-2: Number of record ids vs. throughput

partitions for *Within* operator don't have data duplication, $ImB_{task} = 4\% > ImB_{time} = 0.8\%$, so the time-driven method is better.

Experiment 2: Effect of Total Number of Record ids

In Experiment 2, $(\gamma, n, \gamma_t) = (2000, 3, 2000)$ and α is set to 20, 25, 30, 40, 50. The number of *ids* in a stream affects the throughput when processing the binary-event operators negatively, as evidenced by Figure 5(d)(e)(f)(g). This is

obvious, as large amount of duplication data has to be processed. $Dup_{task}=30\%$, but $Dup_{time}=48.5\%$. Therefore, the time-driven method performs better.

Nevertheless, when processing the unary-event operators in Figure 5(a)(b)(c), taking *Last* for example, $Dup_{task}=0\%$ and $Dup_{time}>0$, so the task-driven approach is better. For the *Within* operator, both partitions have no data duplication, but the imbalance degree varies. When α (number of *ids*) varies at 20, 25, 30, 40, 50, ImB_{time} = 0.53%, ImB_{task} is 5%, 4%, 0%, 2.5%, 2%, respectively. So only when total *id* number is 30, the task-driven partition is better than the time-driven partition.

Experiment 3: Effect of Fragment Length in Time – Driven Partition

In Experiment 3, $(\alpha, n, \gamma_t) = (25, 3, 2000)$ and η is set to 4, 6, 8, 10, 12. From the experimental results presented in Figure 6, we can find that fragmentation length affects the throughput performance of the time-driven partitioning scheme. The reason is obvious, as a larger fragment length leads to a less number of total fragments to be dispatched to the processing nodes, thus the less amount of data overlaps among the nodes to process. That’s why we can see a growing trend with the increase of the number of pieces per fragment (i.e., η). However, when η reaches a certain value, such an improvement gets lost. The reason is one or two nodes need to process one more fragment while the third node has no fragment allocated when the stream comes to the end.

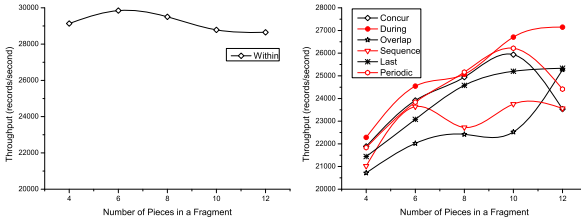


Fig. 6. Exp-3: Number of pieces per fragment (η) vs. throughput under the time-driven partition

To set η properly, for the *Within* operator, $Dup_{time} = 0$, so we only consider the imbalance factor. Let η be 4, 6, 8, 10, 12. ImB_{time} is 1%, 0.51%, 1%, 1%. As we can see in Figure 6, when $\eta = 6$, the system obtain the best throughput.

For the rest operators, say the *Overlap* operator, we consider Dup_{time} and ImB_{time} in order. Let η be 4, 6, 8, 10, 12. Dup_{time} is 3.2%, 1.9%, 1.3%, 1.1%, 0.85%. As we can see in Figure 6, when $\eta = 12$, the system obtain the best throughput. This is consistent to our previous analysis result.

5.2 Experiments on Real GPS Data

We investigate the applicability of the two partitioning methods upon a real GPS stream $S_{GPS} = (t, id, xyPosi)$. We trace a person’s GPS location every 30 second for 10 days, 10 hours each day and obtain 12K records. The number of GPS *ids* is 1 and η is set to 3.

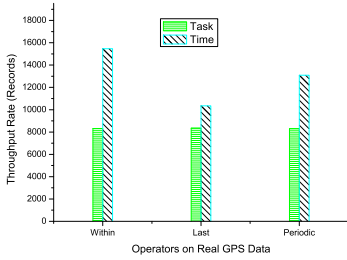


Fig. 7. Experiment on real GPS stream

Experimental results on the real GPS stream are consistent with those on the synthetic data, as shown in Figure 7. For the real GPS stream, as the total number of *ids* is 1, the task-driven partitioning based on *id* allocates all the task to only one node. In this situation, it is equivalent to a single node's processing. Compared with the parallel execution under the time-based partitioning strategy, the later performs consistently better than the former as expected, as illustrated in Figure 7.

6 Conclusion

We describe two types of temporal operators upon events to facilitate event detection from stream data. Two stream partitioning methods (i.e., *time*-driven and *task*-driven) are adopted for parallel processing of the temporal operators. The experiments with both synthetic and real life stream data sets show that the amount of data duplication among processing nodes, as well as the balanced sub-task assignment (under the second *task*-driven approach), influence the throughput performance of the overall system substantially. Adopting which partitioning strategy must take these two factors into consideration. Issues related to out-of-order data arriving and more temporal operators in one query are also interesting topics to pursue in the future.

Acknowledgement. The work is funded by National Natural Science Foundation of China (61073004) and Chinese Major State Basic Research Development 973 Program(2011CB302200).

References

1. Cougar, <http://www.cs.cornell.edu/database/cougar>
2. Adaikkalavan, R., Chakravarthy, S.: Snoopib: interval-based event specification and detection for active databases. *Data Knowl. Eng.* 59 (2006)
3. Allen, J.F., Ferguson, G.: Actions and events in interval temporal logic. *Journal of Logic and Computation* 4 (1994)
4. Arasu, A., Babu, S., Widom, J.: Cql: A language for continuous queries over streams and relations. In: *DBPL* (2004)
5. Bai, Y., Wang, F., Liu, P., Zaniolo, C., Liu, S.: Rfid data processing with a data stream query language. In: *ICDE* (2007)
6. Barga, R.S., Goldstein, J., Ali, M.H., Hong, M.: Consistent streaming through time: A vision for event stream processing. *CoRR* (2006)
7. Catziu, S., Dittrich, K.R.: Samos: an active object oriented database system. *Data Engineering* 15 (1992)
8. Cetintemel, U., Abadi, D., Ahmad, Y., et al.: The aurora and borealis stream processing engines (2006)

9. Cranor, C., Johnson, T., Spataschek, O., Shkapenyuk, V.: Gigascope: a stream database for network applications. In: SIGMOD (2003)
10. Dayal, U., et al.: The hipac project: combining active databases and timing constraints. In: SIGMOD Rec., vol. 17 (1988)
11. DeWitt, D., Gray, J.: Parallel database systems: the future of high performance database systems. *Commun. ACM* 35 (1992)
12. Ghandeharizadeh, S., DeWitt, D.J.: Hybrid-range partitioning strategy: a new declustering strategy for multiprocessor databases machines. In: VLDB (1990)
13. Golab, L., Özsu, M.T.: Issues in data stream management. In: SIGMOD (2003)
14. Hammad, M., Mokbel, M., Ali, M., et al.: Nile: a query processing engine for data streams. In: ICDE (2004)
15. Jaeger, U., Obermaier, J.K.: Parallel event detection in active database systems: The heart of the matter. In: ARTDB (1997)
16. Johnson, T., Muthukrishnan, M.S., Shkapenyuk, V., et al.: Query-aware partitioning for monitoring massive network data streams. In: SIGMOD (2008)
17. Khan, M.F., Paul, R., Ahmed, I., Ghafoor, A.: Intensive data management in parallel systems: A survey. *Distrib. Parallel Databases* 7 (1999)
18. Law, Y.-N., Wang, H., Zaniolo, C.: Aquery: Query languages and data models for database sequences and data streams. In: VLDB (2004)
19. Lerner, A., Shasha, D.: Aquery: query language for ordered data, optimization techniques, and experiments. In: VLDB (2003)
20. Özsu, T.M., Valduriez, P.: Principles of distributed database systems (1999)
21. Teeuw, W.B., Blanken, H.M.: Control versus data flow in parallel database machines. *IEEE Trans. Parallel Distrib. Syst.* 4 (1993)

Towards Effective Event Detection, Tracking and Summarization on Microblog Data

Rui Long, Haofen Wang, Yuqiang Chen, Ou Jin, and Yong Yu

Apex Data & Knowledge Management Lab, Shanghai Jiao Tong University
No. 800 Dongchuan Road, Shanghai, 200240, China
{longrui,whfcarter,yuqiangchen,kingohm,yyu}@apex.sjtu.edu.cn

Abstract. Microblogging has become one of the most popular social Web applications in recent years. Posting short messages (i.e., a maximum of 140 characters) to the Web at any time and at any place lowers the usage barrier, accelerates the information diffusion process, and makes it possible for instant publication. Among those daily user-published posts, many are related to recent or real-time events occurring in our daily life. While microblog sites usually display a list of words representing the trend topics during a time period (e.g., 24 hours, a week or even longer) on their homepages, the topical words do not make any sense to let the users have a comprehensive view of the topic, especially for those without any background knowledge. Additionally, users can only open each post in the relevant list to learn the topic details. In this paper, we propose a unified workflow of event detection, tracking and summarization on microblog data. Particularly, we introduce novel features considering the characteristics of microblog data for topical words selection, and thus for event detection. In the tracking phase, a bipartite graph is constructed to capture the relationship between two events occurring at adjacent time. The matched event pair is grouped into an event chain. Furthermore, inspired by diversity theory in Web search, we are the first to summarize event chains by considering the content coverage and evolution over time. The experimental results show the effectiveness of our approach on microblog data.

1 Introduction

Since Twitter launched in 2007, microblogging has been becoming more and more popular. In China, Sina Microblog¹, the most popular China microblog site, has attracted over 100 million registered users since it went online in 2009. Everyday users publish millions of short messages covering different topics and describing various events in our life. The microblog data contains rich contents such as text, social networks, multimedia, and temporal information. The huge amount of user generated data has arisen strong interests from the industry world. Google, Microsoft and Yahoo have ordered Twitter data for real-time search. On the

¹ <http://t.sina.com.cn>

other hand, it also brings many challenges to the academia especially in the field of natural language processing, Web mining and machine learning.

The character length restriction makes a microblog post rather short, which is quite different from the many other contents published on the Web. Facing the shortness of the content, many existing mining algorithms always suffer from poor performance due to the feature sparsity. Meanwhile, the microblog community is producing a continuously changing data stream corpus. The microblog data reflects the dynamics of our social society and records what is happening in our daily life. With respect to the two key characteristics of microblog data mentioned above, one cannot expect to have the comprehensive view of an event through reading one or several posts.

Let us present a concrete example in Sina Microblog. On March 2nd, 2011, several posts mention “Steven Jobs” in their contents. Later “Steven Jobs” become a topical word appearing in many posts, and thus was listed as a hot topic in the Sina Microblog trend notice². However, a single word “Steven Jobs” in the trend notice cannot tell sufficient information about what happened or is happening. In order to better understand the event, users have to look up the news wire or read many related posts, which might take much time. Finally, they found the Apple Corporation organized a conference to release its new tablet PC iPad2 on March 2nd, 2011. Steven Jobs presented the conference and introduced the product. In the following days, users continued to post messages to discuss the conference and express their opinions on iPad2.

Thus, it is necessary to generate a summary which represents the event context for users to understand and follow. Here, how to capture the embedded semantics of posts related to an event is an issue. Besides, how to organize the mined topical information for a comprehensive summary is another issue to deal with. Furthermore, since the event evolves over time, change tracking should be considered and the event summary should reflect the topic drifts.

In this paper, we propose a unified workflow to detect events, track events and summarize event streams from microblog posts, which tries to address all the above issues. The main contributions are summarized as follows.

- We introduce novel features considering the characteristics of microblog data to select topical words, and thus detect events.
- We formulate the event tracking task as a graph matching problem on a bipartite graph. Through the maximum-weight bipartite graph matching algorithm, we achieve the global optimization for tracking events in adjacent time, and group the related events in form of event chains.
- We propose a novel approach to summarize events. Instead of generating simple text snippets or a single sentence to describe the event, we present the summary with several selected posts by considering their relevance to the event as well as their topical coverage and abilities of reflecting event evolution over time.

² <http://t.sina.com.cn/pub/top/alltopic>

This paper is organized as follows. We discuss the related work in Section 2. Section 3 presents the overview of the workflow and the approach details. In Section 4 we show the experimental results, and finally we conclude the paper.

2 Related Work

Nowadays, microblogging especially Twitter has caught much attentions from researchers. In 2007, Java et al. [2] studied the social network of Twitter users and investigated the motivation of these users. Takeshi Sakaki et al. [6] made use of the Twitter data to detect earthquake by the real-time nature of tweets. Jaime Teevan et al. [9] made a comprehensive comparison between microblog search and traditional Web Search through analyzing a large amount of Twitter query logs. They found that people tend to search temporal relevant information (e.g., breaking news and popular trends) or social information like sentiments and user opinions on some specific events or entities). There also exists some sociological study [5] which tells you how to become visible in Twitter.

In the technical point of view, the most related task is summarization. While sentence summarization within documents and event summarization of news articles have been studied for years, summarization over microblog data is still in its infancy. [8] is the only recent effort who works on summarizing tweets. Beaux Sharifi et al. proposed an algorithm called “Phrase Refinement” to summarize multiple tweets representing the same event and output the final summary with just one sentence. Compare with our work (i.e., event stream summarization), they failed to consider the topical coverage and topic evolution over time.

Another related task is called TDT (Topic Detection and Tracking) [3]. It aims at extracting and tracking events from textual corpora including news wires and corporation documents. Qiankun Zhao et al. [10] proposed a method to detect events from social text streams based on temporal and Information flow. In [7], Hassan Sayyadi introduced a way to detect events based on a graph built by keywords from news documents. However, these approaches cannot be directly applied to microblog data. Different from general textual data, a microblog post is very short with a maximum of 140 characters. Its content reflects real-time events in our life. It also contains rich social information (e.g., following and followed relations) and temporal attributes. In this paper, we adapt the existing methods by integrating novel features appropriate to the characteristics of microblog data.

3 Approach Details

The whole process is composed of three steps. First, we detect events from daily microblog posts. Then related events are tracked based on our novel tracking algorithm. Finally, tracked event chains are summarized for user to better understand what are happening. Figure 1 shows the overall workflow.

³ <http://www.itl.nist.gov/iad/mig/tests/tdt/>

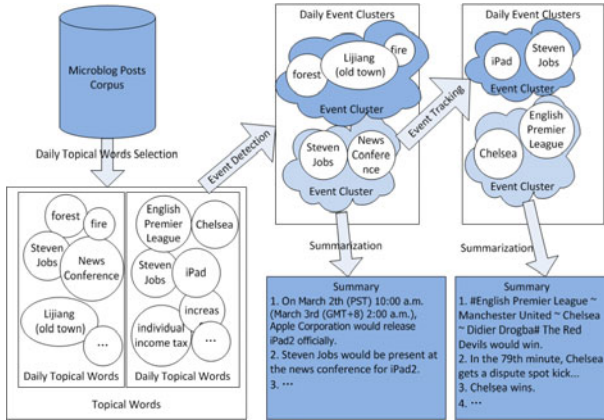


Fig. 1. The unified workflow of event detection, tracking and summarization

3.1 Clustering-Based Event Detection from Topical Words

In this section, we present a clustering-based method to detect events based on carefully selected topical words. In the context of microblogging, events are usually hot topics occurring in posts followed or retweeted by many users. For example, the event “Apple Cooperation publishes iPad2 officially” attracted many attentions from Apple fans from all over the world. They give comments and share the information to their friends. Among the related posts, several words like “Apple” and “iPad2” frequently co-occur. They can be regarded as a compact representation of the event. We call these words *topical words* of an event. Moreover, the co-occurrence degree reflects the relationship strength between two topical words. Thus, the event detection task is reduced to two sub-tasks: 1) how to select representative topical words for some given event; and 2) how to group topical words into clusters in which each cluster represents an event.

Topical Words Selection

A topical word is usually more popular than others with respect to an event. Document Frequency [1] is widely used for feature selection in textual corpora and therefore can be adapted to select topical words.

$$df_i = \frac{|\{d : t_i \in d\}|}{|D|} \quad (1)$$

where $|D|$ denotes the total number of documents, and $|\{d : t_i \in d\}|$ represents the number of documents where the term t_i appears. Here, we consider the total document set as a one day’s post because it allows us to capture emergent events within one day. Regarding long-term events, we divide them into pieces (each per day) as event chains for tracking. For the purpose of frequency smoothing, we further calculate the average document frequency of the word w_i in the

consecutive three days (i.e., the current day and the previous two days). The value of df_i is then updated by the newly calculated score.

Besides words from microblog posts themselves, hashtags are indicators to reflect users' intension and are usually related to events. Therefore, the frequently occurred words in hashtags are more likely to be topical words. We introduce "Word Occurrence in Hashtag" as an unique measure to calculate the probability of a word to be a topical word. The equation is as follows.

$$poh_i = \frac{n_i}{N} \quad (2)$$

where n_i is the number of occurrences of the word w_i in hashtags, and N is the total number of hashtags during some time period. Here, we also consider one day as the basic time duration unit.

In addition, we observe topical words are not uniformly distributed within one day. Particularly, for some emergent events, a related topical word appear much more times in some period. In this way, we borrow the idea of entropy from the information theory [4] to measure this characteristic. Given that we divide one day into 24 hours, the probability of a word appearing in the i th hour is $P_i = \frac{TF_i}{TF}$, where TF_i is the frequency of the word w_i within posts as a single document in the i th hour, and TF is the total term frequency in the whole day's posts. The entropy of the word w_i is defined as

$$entropy = - \sum_{i=1}^k p_i * \log(p_i) \quad (3)$$

When a word satisfies all the following criterions at the same time, it is selected as a topical word.

- The document frequency of a topical word should be larger than the average document frequency of all words appearing in the same day's posts;
- the increasing rate of the word's document frequency should be larger than some empirical threshold;
- the probability of the word occurrence in hashtags should be larger than the average value of total words;
- the word entropy should be smaller than the average value of total words.

Grouping Topical Words into Event Clusters

As mentioned above, if the two topical words co-occur frequently, they might be related with each other to represent different aspects of an event. In order to group related topical words together to clusters in which each represents a unique event, we build a co-occurrence graph $G(V, E)$ from the selected topical words. Each vertex $v_i \in V$ represents a topical word w_i , and an edge $e_{i,j} \in E$ between v_i and v_j denotes that w_i and w_j co-occur in some posts. The weight of the edge reflects the co-occurrence strength. Then a hierarchical divisive clustering approach is used on the graph G to divide topical words into k clusters. To achieve a better clustering result, the adjacency matrix of G is normalized. As a result,

related topical words are grouped together to event clusters. Here, we perform event clustering using Cluto toolkit⁴ which has the algorithm implemented.

3.2 Graph-Based Event Tracking

Maximum-Weighted Bipartite Graph Matching

While we discover events in form of clusters for each day, events occurring at different days might be related. In this section, we try to further track the changes among related events happening at different time. Here, we formulate the event tracking task as a bipartite graph matching problem.

We organize the consecutive days' event clusters on different side. Edges are added between two clusters if the two events are related to form an event chain. Formally speaking, a bipartite graph G is in the form of $(V_1 \cup V_2, E)$ where V_1 and V_2 represent the two event cluster sets in the consecutive days, and $E = \{(v_i, v_j), v_i \in V_1, v_j \in V_2\}$ in which each (v_i, v_j) represents two events related to form an event chain, and should be linked together. Furthermore, we extend the bipartite graph G to G' by assigning edge weights. The weight of an edge shows the probability of the event cluster pair to be linked together and is denoted as $w_{i,j}$ where $(v_i \in V_1, v_j \in V_2)$. Here we adopt maximum-weighted bipartite graph matching (MWBGM) to G' . The objective of MWBGM is to maximize the multiplication of probabilities of the finally linked event cluster pairs. The process of MWBGM is to estimate the maximum likelihood, which can be simply reduce to maximize the sum of those log probabilities of the final linkages. We apply the Kuhn-Munkres algorithm⁵ to solve this problem. The similarity between two event clusters is measured by their intersection. Finally the related event clusters are grouped in form of event chains.

Assessing the Accuracy of the Linkage

In order for tracking events, the basic building block in maximum weighted bipartite graph matching is similarity calculation between two event clusters. An intuitive way to measure the relatedness between two events is to use Jaccard coefficient⁶ which calculates the ratio of the intersection between two clusters to their union. The intersection counts for the common words in both clusters and the union considers the total number of topical words from either cluster. However, the above measure encounters a problem that it usually leads to very low scores so that it is hard for us to distinguish related events from others. For example, two related events might share only one word. As a result, for some high threshold, we will filter the linkage between them out. It causes a false negative example. On the other hand, when we set relative low threshold, we would remain incorrect linkages which belong to false positive examples. Hence, we slightly change the Jaccard coefficient to $r = \frac{\sum_i df_i \in I}{\sum_j df_j \in U}$, where I denotes the intersection word set, U is the union word set, and df_i (df_j) represents

⁴ <http://glaros.dtc.umn.edu/gkhome/views/cluto>

⁵ http://en.wikipedia.org/wiki/Kuhn-Munkres_algorithm

⁶ http://en.wikipedia.org/wiki/Jaccard_index

the document frequency of the word w_i (w_j). By substituting the document frequency for direct counting, we can still have a large linkage score for the above example if the common word is of high frequency.

3.3 Event Summarization

In order for users to understand what are happening, an event summary of an event chain generated from the previous step (i.e., event tracking) is useful to capture the event context. Similar to document summarization, content coverage should be considered. Therefore, we select relevant but as much content coverage as possible microblog posts with respect to events in the event chain from the whole data set.

More precisely, each event cluster denoted as c is represented as a vector of topical words. All relevant posts in which each contains at least one topical word are called D_c . The similarity $sim(c, d)$ between a post $d \in D_c$ and the cluster c is measured by the cosine similarity between their feature vectors. Similarly, we can calculate the similarity $sim(d, d')$ between two posts d and d' . We reduce the event summarization task into finding a subset $S \subset D_c$ which is composed of top- k most relevant posts with as much content coverage as possible. To reflect the content coverage characteristic between two related posts, we incorporate the normalized time interval into their original similarity function. The normalized time interval is denoted as $time_interval(d, d') = |t(d) - t(d')|/24$, where $t(d)$ and $t(d')$ represent the time stamp of the post d and d' respectively. This normalization is intuitive to group one days' posts into different hour periods, so 24 is chose as a normalization constant. The updated similarity function $sim'(d, d')$ can be represented as the multiplication of $sim(d, d')$ and $time_interval(d, d')$. Therefore, two similar posts tend to have a larger similarity score if their time interval is longer. The similarity between the subset S and a post $d \notin S$ is then defined as

$$sim(S, d) = \max_{d' \in S} sim'(d, d') \quad (4)$$

where we calculate the maximal similarity score between d and one post belonging to S and treat it as the similarity score between the post and the subset. Since the content coverage is usually NP-hard, we use a greedy algorithm to select the subset S with k posts. The whole process is as follows.

- when S is an empty set, we select a post $d = \arg \max_{d \in D_c} sim(c, d)$, and update S to $\{d\}$
- then we continue to add a post d' to the subset until we have k posts in S . Here, we pick up a post if $d' = \arg \min_{d' \in D_c} (sim(c, d') \times sim(S, d'))$.

Once we generate the summary for any event cluster in an event chain returned in the tracking step, we can get the event chain summary composed of these event summaries plotting on the time line. Using this greedy algorithm, we have an overall computational complexity of $O(|S| \cdot |D_c|)$ for choosing each successive post, or $O(k^2 \cdot |D_c|)$ for re-ranking the top k posts, where $|S|$ ($|D_c|$) denotes the size of S (D_c).

4 Experiments

4.1 Data Collection and Preprocessing

While our proposed approach is language-independent, we use Sina Microblog as our experimental data source. We crawled Sina Microblog from December 23th, 2010 to March 8th, 2011 through its API. As a result, we have collected more than 22 million microblog posts published by 6.8 million different user. The data collection was divided into 76 pieces in which each consists of one day’s posts.

Before event detection and the following steps, we pre-processed the data collection as follows. Firstly, we removed emotion phrases (e.g., ha-ha), short URLs (e.g., <http://sinaurl.cn/hVyH0>) formatted by Sina Microblog and reposting marks like “RT” and “//”. Then we used an English stop-word list to filter English stop words such as “the”. Finally, we used “ICTCLAS”⁷ for Chinese word segmentation in all posts.

4.2 Evaluation of Event Detection

Although TDT (Topic Detection and Tracking) has sophisticated evaluation metrics, we cannot directly use these metrics for our task. Different from TDT, we do not know the number of events occurring in one day and we do not know what these events are about. In other words, there are no ground truths available for event detection and tracking in microblogging. Thus, we select precision as our only evaluation metric. Since event detection over microblog data does not have any restrictions on event details, we cannot compare our method with those used in TDT. There are two aspects that will influence the accuracy of whether one cluster actually represents an event: features used for topical word selection and algorithms used for clustering. Hence, we divide topical word feature selection into three levels (i.e. document frequency only denoted as D, document frequency + entropy as DE, and further including word occurrence in hashtags as DEH). We also list three clustering options namely K-means, hierarchy clustering and k-way top-down divisive clustering used in this paper. Then we compare the precision performance using different clustering algorithms based on various feature selections for topical words.

As shown in Figure 2, we can see our proposed feature selection outperforms considering document frequency only and document frequency with entropy consistently using any clustering method. It also shows that the hierarchy clustering and K-means get rather poor performance no matter what k to use. Here, k is a parameter to control the termination of the clustering process. Moreover, our proposed clustering with the carefully selected topical words (based on DEH) achieves the best precision at the value around 0.4. When considering the contribution of k , we find that returning 20 event clusters is always the optimal choice which tend to return as many correct events as possible and avoid incorrect ones. Compared with the performance results reported by TDT, ours is relatively low. However, since our task does not restrict the number and other details of events

⁷ <http://ictclas.org/>

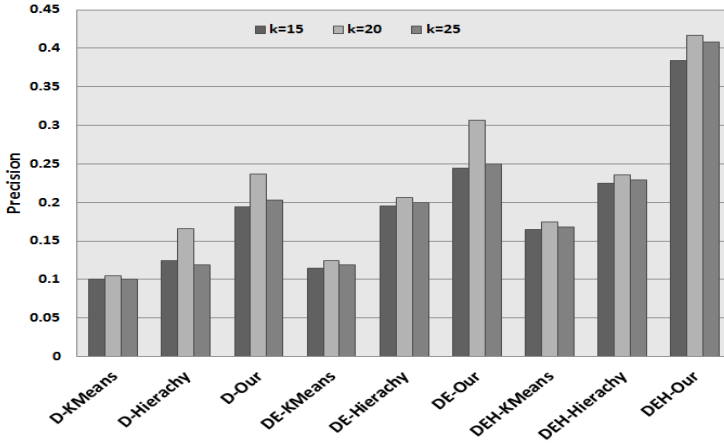


Fig. 2. Precision of events detection under different criteria of topical words selection and different clustering methods

and runs on microblog data instead of news articles (much easier to process), the precision performance by our approach is still acceptable.

4.3 Evaluation of Event Tracking

As mentioned in Section 3.2, we introduced a slightly changed Jaccard coefficient r as an event linkage threshold. In the evaluation of event tracking, we set r to different values (i.e., 0.05, 0.2, 0.4, 0.6, 0.8, and 1.0). Once we get the event tracking results under r using a different value, we check the correctness of linkages between events happening in consecutive days. Furthermore, we manually add missing linkages. Thus, both precision and recall can be calculated for event tracking performance evaluation. Figure 3 presents the experimental results. More precisely, Figure 3(a) shows the Precision-Recall curve under different r values. With the decrease of r , we relax the requirement to filter out linkage candidates. Hence, we have discovered more linkages, which results in higher coverage but lower precision due to the fact that we introduce more noise. Figure 3(b) illustrates the corresponding F1-measure curve where we get the best performance when r is chosen around 0.4.

4.4 Evaluation of Event Summarization

In the multi-document summarization task, ROUGE [3] is widely used to measure the similarity between an automated summary and each of the manual labeled summaries. Different from multi-document summarization (in form of several selected keyphrases), we return k most relevant but diverse posts to users to capture the event context. Since we do not have manually labeled summary in hand, we cannot directly use an automatic evaluation framework like

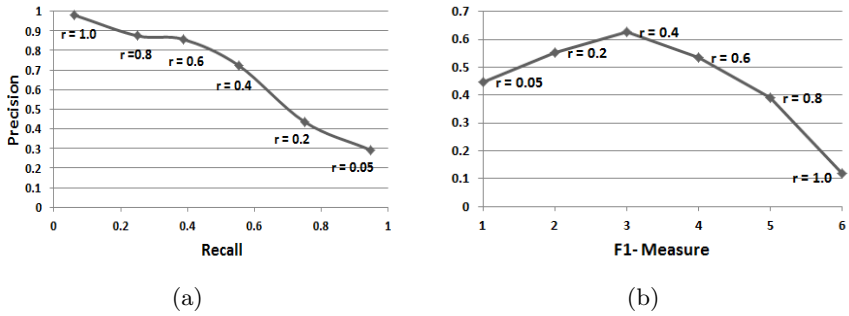


Fig. 3. Event tracking result w.r.t Precision-Recall and F1-Measure

ROUGE. In our experiment for event summarization evaluation, we use *precision @ n* ($P@n$) to assess the relevance of an event summary. Note that n denotes the number of posts selected to represent an event. Figure 4 shows the average precision value where n ranges from 1 to 10. From the table, we can always get a precision score larger than 0.5, which is acceptable with respect to the very hard open event summarization on microblog data. Meanwhile, when n ranges from 1 to 3, the precision score is large and acceptable. Furthermore, with the increase of n , it is no surprising that the precision falls down. While the summary might contain noise posts, it provides much comprehensive view of the whole events.

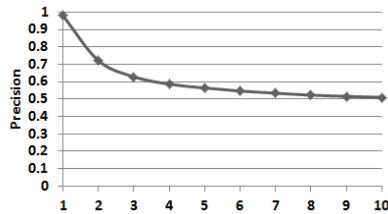


Fig. 4. $P@n$ results of relevance of event summary

We then evaluate the content coverage of S . Intuitively, we could order all the posts related to an event cluster by the similarity between a post and an event cluster in decreasing order, and then choose the top k posts. This method is called Top K Selection. Our method is compared to the method above. Before the evaluation metric is introduced, an assumption, that posts in an event summary could represent information contained by the unchosen posts P_r in all the posts related to the event cluster, is made. With the assumption above, the sum of maximum cosine similarity between an event summary S and the unchosen posts P_r could be used to measure coverage. The exact equation is defined as $sum_of_similarity = \sum_{i=1}^n \arg \max_{p \in S} sim(p_i, p)$, with p_i as the i th post in P_r , and n as the size of P_r . The larger the value of $sum_of_similarity$

is, the better the coverage is. As Table 1 illustrating, our method achieves much better result than the other method. It is clear that the relevance criterion is not sufficient during considering the content coverage of an event summary. The event evolution over time should be considered.

Table 1. Event summary coverage comparison between two methods

	Our Method	Top K Selection
Coverage	105.271	90.268

Figure 5 presents a snapshot of partial results of event detection, tracking, and summarization in the consecutive three days since February 26th, 2011. Each oval node represents an event and its color indicates the popularity of the event. The deeper the color is, the more popular the event is. In the right part of the figure, we can see some events are becoming more and more popular while some others attract less attention. In the left part, we list the summary for a given event in which the selected relevant posts are sorted by their time stamps. Users can also track an event chain through reading posts in the event summaries. Thus, we provide a way to present the complete view of event chains with the summaries for involved events in detail.

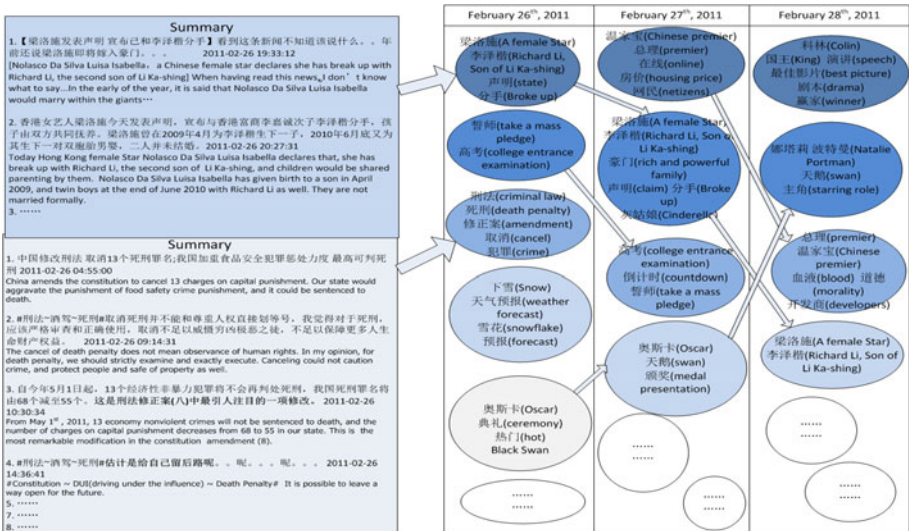


Fig. 5. A snapshot of partial results of event detection, tracking, and summarization in the consecutive three days since February 26th, 2011

5 Conclusion

In this paper, we proposed a unified workflow for event detection, tracking and summarization on microblog data. It covers three main steps including event detection, event tracking and summarization. Compared from existing work, we introduced novel features considering the characteristics of microblog data to select topical words for event detection. Moreover, we are the first to return a comprehensive summary which considers the content relevance and coverage with respect to the topic as well as the event evolution over time. In the future, we plan to apply our workflow on other microblog sites like Twitter and try more sophisticated semi-supervised clustering for this task.

References

1. Baeza-Yates, R.A., Ribeiro-Neto, B.: *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston (1999)
2. Java, A., Song, X., Finin, T., Tseng, B.: Why we twitter: understanding microblogging usage and communities. In: *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 Workshop on Web Mining and Social Network Analysis, WebKDD/SNA-KDD 2007*, pp. 56–65. ACM, New York (2007)
3. Lin, C.Y., Hovy, E.: Automatic evaluation of summaries using n-gram co-occurrence statistics. In: *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, NAACL 2003*, vol. 1. Association for Computational Linguistics, Stroudsburg (2003)
4. Mitchell, T.: *Machine Learning*. McGraw-Hill Education (ISE Editions) (October 1997)
5. Sakaki, T., Matsuo, Y.: How to become famous in the microblog world. In: *ICWSM (2010)*
6. Sakaki, T., Okazaki, M., Matsuo, Y.: Earthquake shakes twitter users: real-time event detection by social sensors. In: *World Wide Web Conference Series*, pp. 851–860 (2010)
7. Sayyadi, H., Hurst, M., Maykov, A.: Event detection and tracking in social streams. In: Adar, E., Hurst, M., Finin, T., Glance, N.S., Nicolov, N., Tseng, B.L. (eds.) *ICWSM. The AAAI Press, Menlo Park (2009)*
8. Sharifi, B., Hutton, M.A., Kalita, J.: Summarizing microblogs automatically. In: *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, HLT 2010*, pp. 685–688. Association for Computational Linguistics, Stroudsburg (2010)
9. Teevan, J., Ramage, D., Morris, M.R.: #twittersearch: a comparison of microblog search and web search. In: *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, WSDM 2011*, pp. 35–44. ACM, New York (2011)
10. Zhao, Q., Mitra, P., Chen, B.: Temporal and information flow based event detection from social text streams. In: *Proceedings of the 22nd National Conference on Artificial Intelligence*, vol. 2, pp. 1501–1506. AAAI Press, Menlo Park (2007)

Author Index

- Agrawal, Divyakant 3
Alhashmi, Saadat M. 144, 157
- Bamieh, Bassam 3
Bao, Shenghua 327, 442
Bao, Yubin 379
Belkhatir, Mohammed 144, 157
Budak, Ceren 3
- Cai, Xiaoyan 379
Chen, Chen 238
Chen, Hong 614
Chen, Jimeng 191
Chen, Kejia 214
Chen, Mo 602
Chen, Qun 566
Chen, Wei 530
Chen, Yu 542
Chen, Yuqiang 652
Chong, Zhihong 340
- Deng, Lei 68
Deng, Yu 56
Ding, Linlin 238
Ding, Mao 493
Du, Xiaoyong 251, 578
- El Abbadi, Amr 3
- Fan, Ming 505
Fan, Ping 627
Feng, Dengguo 264
Feng, Ling 639
Flanagin, Andrew 3
Fu, Li 554
Fu, Xiong 214
Fung, Gabriel 430
- Gan, Lu 590
Gao, Lei 31
Gao, Sheng 226
Gao, Shuai 302
Gao, Yan 554
Ge, Hefei 493
Gu, Fangming 18
- Gu, Xiwu 94, 106
Gu, Yu 602
Guo, Huaping 505
- Han, Dingyi 327
Han, Jingyu 214
He, Jiazhen 81
Hong, Cheng 264
Hu, Lei 340
Huang, Benxiong 68
Huang, Weijing 530
Huang, Yalou 191
Huang, Zeqian 493
Hung, Edward 56
- Jalali, Vahid 366
Jia, Zixi 602
Jiang, Han 554
Jiang, Tao 201
Jiao, Min 467, 480
Jin, Cheqing 169
Jin, Hongxia 314
Jin, Ou 442, 652
- Kitsuregawa, Masaru 1
Kong, Liang 554
Kou, Yue 353
- Le, Jiajin 415
Leng, Fangling 379
Li, Chuanwen 602
Li, Guohui 226, 627
Li, Piji 302
Li, Qiang 566
Li, Qing 403
Li, Ruixuan 94, 106, 454
Li, Xue 81
Li, Xuefei 415
Li, Yingjun 43
Li, Yuhua 454
Li, Yukun 201
Li, Zhanhuai 181, 566
Li, Zhoujun 277
Liang, Yicong 403
Liao, Chengxuan 614

- Lin, Quan 454
 Liu, Dayou 18
 Liu, Hailong 566
 Liu, Jie 191
 Liu, Kuicheng 578
 Liu, Peng 590
 Liu, Tianbi 191
 Liu, Wenbin 181
 Liu, Xiangyu 118
 Liu, Yang 31
 Long, Rui 652
 Long, Yun 277
 Lu, Jiaheng 614
 Lu, Jing 131
 Lu, Shan 340
 Lu, Wei 251
 Lu, Zhengding 454
 Lv, Qiannan 18
 lv, Zhiquan 264
- Ma, Jun 302
 Ma, Qiang 290
 Maree, Mohammed 144, 157
 Miao, Miao 181
- Ni, Weiwei 340
 Nie, Tiezheng 43, 353
 Nie, Yanming 566
- Pan, Wei 566
 Patterson, Stacy 3
 Peng, Jian 542
 Peng, Shanglian 566
 Peng, Yuwei 493
- Qian, Tieyun 403
 Qiao, Baiyou 238
 Qin, Yongrui 590
- Rong, Chuitian 251
- Sha, Chaofeng 517
 Shan, Jing 43
 Shang, Xuequn 181
 Shen, Derong 43, 353
 Shi, Jingang 379
 Su, Zhong 327, 442
 Sun, Weiwei 590
 Sun, Xingzhi 131
 Szeto, Chi-Cheong 56
- Tang, Changjie 542
 Toyoda, Masashi 1
- Wang, Chuandong 214
 Wang, Chuanjian 493
 Wang, Fei 68
 Wang, Guoren 238
 Wang, Hao 639
 Wang, Haofen 131, 652
 Wang, Liwei 430
 Wang, Miao 181
 Wang, Qihua 314
 Wang, Shan 467, 480, 578
 Wang, Shengsheng 18
 Wang, Tengjiao 530
 Wang, Xianbing 94, 106
 Wang, Xiaoling 517
 Wang, Xiaoye 201
 Wang, Yong 81
 Wang, Yue 530
 Wang, Zhanwei 467, 480
 Wen, Aiming 454
 Wen, Kunmei 94, 106
 Winiwarter, Werner 2
 Wu, Jingjing 590
 Wu, Yuqing 366
- Xiao, Weijun 94, 106
 Xiao, Yingyuan 201
 Xu, Ling 290
 Xu, Linhao 131
 Xue, Wenwei 639
- Yan, Rui 554
 Yang, Bin 517
 Yang, Bo 415
 Yang, Dan 353
 Yang, Dongqing 530
 Yang, Lichun 327
 Yang, Ning 542
 Yang, Xiaochun 118
 Yang, Yufei 94, 106
 Yoshikawa, Masatoshi 290
 Yu, Ge 43, 353, 379, 602
 Yu, Weiren 415
 yu, Xiaohui 31
 Yu, Yong 327, 442, 652
 Yuan, Ling 627
 Yuan, Peisen 517
 Yue, Kou 43

- Zhan, Zhenjiang 327
Zhang, Min 264
Zhang, Xiao 251, 578
Zhang, Xiaoming 277
Zhang, Yan 554
Zhang, Yang 81
Zhang, Yansong 467, 480
Zhang, Yizhen 169
Zhao, Dongyan 390
Zheng, Jinwang 340
Zheng, Weiguo 390
Zhou, Aoying 169, 517
Zhou, Mo 366
Zhou, Xuan 467, 480
Zhu, Jia 430
Zou, Lei 390
Zou, Li 578