Liwei Wang Jingjue Jiang Jiaheng Lu
Liang Hong Bin Liu (Eds.)

LNCS 7142

Web-Age Information Management

WAIM 2011 International Workshops: WGIM 2011, XMLDM 2011, SNA 2011 Wuhan, China, September 2011, Revised Selected Papers

Lecture Notes in Computer Science 7142

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

Liwei Wang Jingjue Jiang Jiaheng Lu
Liang Hong Bin Liu (Eds.)

Web-Age Information Management

WAIM 2011 International Workshops:
WGIM 2011, XMLDM 2011, SNA 2011
Wuhan, China, September 14-16, 2011
Revised Selected Papers

Springer

Volume Editors

Liwei Wang
Wuhan University, Hubei 430072, China
E-mail: liwei.wang@whu.edu.cn

Jingjue Jiang
Wuhan University, Hubei 430072, China
E-mail: whujiang@whu.edu.cn

Jiaheng Lu
Renmin University of China, Beijing 100872, China
E-mail: jiahenglu@ruc.edu.cn

Liang Hong
Wuhan University, Hubei 430072, China
E-mail: hong@whu.edu.cn

Bin Liu
Wuhan University, Hubei 430072, China
E-mail: binliu@whu.edu.cn

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-28634-6 e-ISBN 978-3-642-28635-3
DOI 10.1007/978-3-642-28635-3
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2012932414

CR Subject Classification (1998): H.4, H.3, I.2, C.2, H.5, H.2.8, H.2

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

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

Printed on acid-free paper

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

WAIM 2011 Workshop Chairs' Message

WAIM is a leading international conference for researchers, practitioners, developers and users to share and exchange cutting-edge ideas, results, experience, techniques and tools in connection with all aspects of Web data management. The conference invites original research and industrial papers on the theory, design and implementation of Web-based information systems, as well as proposals for demonstrations, tutorials and panels. Previous WAIM conferences were held in Shanghai (2000), Xian (2001), Beijing (2002), Chengdu (2003), Dalian (2004), Hangzhou (2005), Hong Kong (2006), Huangshan (2007), Zhangjiajie (2008), Suzhou (2009) and Jiuzhaigou (2010). Along with the main conference, WAIM workshops are intended to provide an international group of researchers with a forum for the discussion and exchange of research results related to the conference topics. This WAIM 2011 workshop volume comprises papers from three workshops, which were:

1. The First International Workshop on Web-Based Geographic Information Management (WGIM 2011)
2. The Third International Workshop on XML Data Management (XMLDM 2011)
3. The First International Workshop on Social Network Analysis (SNA 2011)

The contents of these three workshops were selected from a public call-for-proposals process. The workshop organizers put a tremendous amount of effort into soliciting and selecting research papers with a balance of high quality and new ideas and new applications. They also followed a vigorous review process. A total of about 20 papers were accepted. We are very grateful to the main conference organizers. We would also like to take this opportunity to thank all workshop organizers and Program Committee members for their great effort in putting together the workshop program of WAIM 2011.

September 2011

Chengfei Liu
Liwei Wang

The First International Workshop on Web-Based Geographic Information Management (WGIM 2011) Chairs' Message

The rapid development of geographic information infrastructure and increased demand for sharing useful geographic data and information through the Web, has led to a new research focus on how to manage and share geographic information on the Web efficiently and securely. New strategies, tools and systems are required to significantly improve the information sharing and guarantee an easier and quicker access of data from a variety of sources without undermining the ownership of the information. The First International Workshop on Web-Based Geographic Information Management 2011 invited developers, users and expert scientists working in this field all over the world to present their innovative research and development contributions.

We thank the WAIM 2011 organizers for their generous support. The great success of the workshop is indebted to the hard work of all Program Committee members. We also thank all the authors for their contributions.

September 2011

Xiaofang Zhou
Jingjue Jiang

The Third International Workshop on XML Data Management (XMLDM 2011) Chairs' Message

It is our great pleasure to welcome you to the proceedings of the Third International Workshop on XML Data Management (XMLDM 2011).

XML has gained lot of attention from database and Web researchers who are actively working in one or more of the emerging XML areas. XML data are self-describing, and provide a platform-independent means to describe data and therefore can transport data from one platform to another. XML documents can be mapped to one more of the existing data models such as relational and object-relational models, and XML views can be produced dynamically from the pre-existing data models. XML queries can be mapped to the queries of the underlying models and can use their optimization features. XML data integration is useful for e-commerce applications such as comparison-shopping, which requires further study in the domain of data, schema and query-based integration. XML change management is another important area that has attracted attention in the context of Web warehouses. XML has been in use in upcoming areas such as Web services, sensor and biological data management. The Third International Workshop on XML Data Management focused on the convergence of database technology with XML technology, and brought together academics, practitioners, users and vendors to discuss the use and synergy between these technologies.

XMLDM 2011 accepted six full papers from Asia, Europe, and Australia. These papers cover a variety of topics, including XML views, XML query, similarity join, XML database and so on. We hope that they will serve as a valuable starting point for many brilliant projects in XML data management.

The paper "Multidimensional Implementation of Stream AD" introduces a multidimensional implementation of the stream for path-labeling schemes. The authors show that this implementation can be extended in such a way that it supports fast searching of nodes with a content, and it is also necessary to combine two variants of the R-tree (Ordered R-tree and Signature R-tree) for an efficient implementation of the stream ADT.

The paper "Measuring XML Structured-ness with Entropy" proposes and evaluates entropy-based metrics for XML structured-ness which could measure the structural uniformity of path and subtrees, respectively. Furthermore, the authors also study the correlation of these metrics with real and synthetic data sets.

In their paper "Similarity Join on XML Based on k-Generation Set Distance," Wang et al. put forward two new edit operations (reversing and mapping) together with related algorithms concerning similarity join based on the new

defined measure and the method of using k-generation set distance instead of edit distance when comparing similarity between trees.

In the next paper "XML Query Answering Using View," Yao et al. study query answering using the views problem for tree pattern queries (QAV) and deal with this problem by finding an equivalent rewriting of tree pattern queries.

The paper "XIO-SLCA: Optimize SLCA for Effective Keyword Search in XML Documents" studies the problem of effective keyword search over XML documents and proposes that keyword search returns the set of smallest trees, where a tree is designated as smallest if it contains no sub-tree that also contains all keywords. After an in-depth analysis of the Indexed Lookup Eager algorithm (IL), they proposes an optimized method called XIO-SLCA to improve keyword search quality which could achieve both a higher recall and precision when compared with the existing SLCA.

In "The Development of XML-Stored Procedures in XML-Enabled Databases," Fahad Alahmari and Eric Pardede investigate how SQL-stored procedures can be developed to effectively conduct various XQuery and XPath against XML data within the enabled databases.

Making XMLDM possible has been a team effort. First of all, we would like to thank the authors and panelists for providing the content of the program. We would like to express our gratitude to the Program Committee and external reviewers, who worked very hard in reviewing papers and providing suggestions for their improvements. In particular we extend our special thanks to Linlin Zhang for maintaining the XMLDM website and for his effort in organizing the workshop.

We hope that you will find these proceedings interesting and thought-provoking.

September 2011

Jiaheng Lu
Tok Wang Ling
Ge Yu

The First International Workshop on Social Network Analysis (SNA 2011) Chairs' Message

In recent years, social network research has attracted more and more scientist and researchers, thanks to the explosion of the Web which has created and is creating social interactions. Analyzing the information underneath the social interactions, such as community detection, opinion mining, link prediction, product recommendation, expert finding, social ranking, information visualization, will benefit both information providers and information consumers in the application areas of social sciences, economics, psychology and computer sciences.

SNA 2011 aimed at bringing together researchers and practitioners interested in this area to share their perspectives, identify the challenges and opportunities, and discuss future research/application directions. The workshop provided oral presentations where researchers and practitioners could share and exchange their knowledge and experience.

September 2011

Jaideep Srivastava
Bin Liu

The First International Workshop on Web-Based Geographic Information Management (WGIM 2011)

Program Co-chairs

Xiaofang Zhou	University of Queensland, Australia
Jingjue Jiang	Wuhan University, China

Program Committee

Hoyoung Jeung	EPFL, Switzerland
Jiaheng Lu	Renmin Univeristy of China
Ke Deng	University of Queensland, Australia
Mohamed Mokbel	University of Minnesota, USA
Shuliang Wang	Wuhan University, China
Wen-Chih Peng	National Chiao Tung University, Taiwan
Xing Xie	Microsoft Research Asia, China
Yang Yue	Wuhan University, China
Zhiming Ding	Chinese Academy of Sciences Software Institute, China

The Third International Workshop on XML Data Management (XMLDM 2011)

Program Co-chairs

Jiaheng Lu	Renmin University of China, China
Tok Wang Ling	National University of Singpaore, Singapore
Ge Yu	NorthEast University, China

Program Committee

Zhifeng Bao	National University of Singapore, Singapore
Stephane Bressan	National University of Singapore, Singapore
Peter Boncz	Centrum Wiskunde&Informatica, The Netherlands
Chee Yong Chan	National University of Singapore, Singapore
Xiaoyong Du	Renmin University of China, China
Jianhua Feng	Tsignhua University, China
Jun Gao	Peking University, China
Masaru Kitsuregawa	Tokyo University, Japan
Jianzhong Li	Harbin Institute of Technology, China
Guoliang Li	Tsinghua University, China
Xuemin Lin	University of New South Wales, Australia
Xiaofeng Meng	Renmin University of China, China
Xiaochun Yang	North-East University, China
Liang Xu	National University of Singapore, Singapore
Jeffrey Xu Yu	The Chinese University of Hong Kong, China
Zografoula Vagena	Microsoft Research, USA
XiaoLing Wang	Fudan University, China
Hongzhi Wang	Harbin Institute of Technology, China
Peter Wood	University of London, UK
Aoying Zhou	East China Normal University, China
Yongluan Zhou	University of Southern Denmark, Denmark
Xiafeng Li	Texas A&M University, USA

The First International Workshop on Social Network Analysis (SNA 2011)

Program Co-chairs

Jaideep Srivastava	University of Minnesota, USA
Bin Liu	Wuhan University, China

Program Committee

Lei Tang	Yahoo! Labs, USA
Zenglin Xu	Purdue University, USA
Bin Cui	Beijing University, China
Wei Wang	The University of New South Wales, Australia
Xifeng Yan	University of California at Santa Barbara, USA
Lifeng Sun	Tsinghua University, China
Ee-Peng Lim	Singapore Management University, Singapore
Kuo-Wei Hsu	National Chengchi University, Taiwan

Table of Contents

The First International Workshop on Web-Based Geographic Information Management (WGIM 2011)

The Third International Workshop on XML Data Management (XMLDM 2011)

The First International Workshop on Social Network Analysis (SNA 2011)

Enhancing the Quality of Place Resources in Geo-folksonomies

Ehab ElGindy and Alia Abdelmoty

School of Computer Science and Informatics
Cardiff University, Wales, UK
{ehab.elgindy,a.i.abdelmoty}@cs.cardiff.ac.uk

Abstract. Users' interaction and collaboration on Web 2.0 via social bookmarking applications have resulted in creating a new structure of user-generated data, denoted folksonomies, where users, Web resources and tags generated by users are linked together. Some of those applications focus on geographic maps. They allow users to create and annotate geographic places and as such generate geo-folksonomies with geographically referenced resources. Geo-folksonomies suffer from redundancy problem, where users create and tag multiple place resources that reference the same geographic place on the ground. These multiple disjointed references result in fragmented tag collections and limited opportunities for effective analysis and integration of data sets. This paper, (1) defines the quality problem of resources in a geo-folksonomy (2) describes methods for identifying and merging redundant place resources and hence reducing the uncertainty in a geo-folksonomy, and (3) describes the evaluation of the methods proposed on a realistic sample data set. The evaluation results demonstrate the potential value of the approach.

Keywords: Web 2.0, Folksonomy, Geographical Similarity, Social Bookmarking, Tagging, Geo-Tagging.

1 Introduction

Web 2.0 has created a new type of Web-based interaction among Internet users by introducing social bookmarking applications, where users can publish contents to share it with others. The published contents are Web documents such as Web pages, images or PDF documents. In addition, users can provide keywords (tags) to categorize the contents/resources they publish, thus resulting in new structures of information – called folksonomies – that links users, tags and resources together.

Folksonomies directly reflects the vocabulary of users [12], enabling matching of users' real needs and language. Although folksonomies are semantically rich, they are un-controlled, unstructured, sometimes ambiguous and inconsistent. Ongoing research efforts consider the extraction of certain semantics from

L. Wang et al. (Eds.): WAIM 2011 Workshops, LNCS 7142, pp. 1–12, 2012.

folksonomies. For example, Rattenbury et al. [14] extract place and event semantics from Flickr[1] tags using the usage distribution of each tag.

Some social bookmarking applications, such as Tagzania[2], are specialized in tagging geographic places using a map-based Web interface. These applications generate a special kind of folksonomy, denoted geo-folksonomy in this work. The tagging behaviour - which generates folksonomies - allows users to choose keywords to describe/index the information in a specific web resource such as a web page. For example, a user can tag an article he read about a good cooking recipe as ("best", "recipe", "for", "making", "fajita"). However, when it comes to tag a place, users create a place resource using a map-based interface which represents a place in reality, and then tags are provided to describe the place in reality although they are attached to the place resource. For example, a user can create a place resource named "Cardiff university" and set its spatial location using a map interface such as Google maps, and then the user can attach relevant tags such as ("University", "Study", "Research").

Place resources in geo-folksonomies have some characteristics which do not exist in normal web resources:

1. Place resources are created by the social bookmarking applications to reference places in the real world, while normal web resources already exist in the web space and they are just referenced using unique URLs.
2. Although it is possible to assign a unique URI for any resource (including place resources [2]), URIs are not used to locate places as people always refer to places by spatial and thematic attributes such as location and place name respectively.
3. The values of spatial attributes - such as longitude and latitude - are acquired using a map-based applet. This method of acquiring data can be imprecise and is dependent on the user being able to identify and digitize a precise location on a map offered on the user interface of these applications. The accuracy is also related to the map scales offered to users and the difficulty in matching the precise location across map scales.
4. The values of thematic attributes - such as place names - are acquired using a free-text input. Although they add valuable semantics to the place resources, they are associated complexity, where people use non-standard, vernacular, place names [5] and abbreviations.

Most of the applications that generate geo-folksonomies aim to collect as much information as possible about places, which can be one of the reasons why such applications do not allow users to share place resources and why they require a new place resource to be created each time a user wants to tag a place. Such design can result in having multiple place resources that reference the same place in real world. We argue that, such redundancy in the geo-folksonomy structure can produce inaccurate results when using folksonomy analysis techniques such as tag-similarity methods.

[1] http://www.flickr.com
[2] http://www.tagzania.com

The combination of inaccuracies in place location and fuzziness in naming place entities complicates the task of uniquely identifying place resources and can hence lead to the presence of redundant resources in the geo-folksonomy, degrading its quality. Hence, identifying and relating those place resources can lead to more consistent and useful analysis of geo-folksonomies and support integrating place resources from different data sources.

The work presented in this paper defines and formulates a quality problem in geo-folksonomy resources. Methods are proposed for addressing this problem and for creating an enriched geo-folksonomy. The solution involves using online Web resources to first qualify place instances with identifiers that can then be used in a process of clustering and aggregation to uniquely identify related resources.

The enriched geo-folksonomy contains more certain information, as the method takes into consideration the user votes and agreements.

The rest of this paper is organized as follows. Related work is discussed in section 2. The research problem is described in section 3, followed by the proposed methods in section 4. Experimental results and evaluation are given in section 5 and the paper concludes by some discussion and outlook on future work 6.

2 Related Work

Folksonomies are user-generated data created by users' interaction and collaboration using social bookmarking applications. Typically, such applications are designed to acquire the input from users in free-text format to simplify the user interface. As a result, the generated folksonomies contain uncontrolled vocabulary of keywords (tags) with several problems such as polysemy (a word which has multiple related meanings) and synonymy (different words that have identical or very similar meanings) [6]. On the other hand, folksonomies can be considered as a rich data source that contain embedded semantics. As such, many research works targeted the problem of extracting semantics from folksonomies including the problems mentioned above [20,15,8,13,18,1,17,3]. The extracted semantics are usually represented by a simple lightweight ontology, which is a simple tree hierarchy of terms where parent terms are semantically general/broader than their children.

Folksonomies are typically modeled by a tripartite graph with hyper edges [13]. Vertices are partitioned into three disjoint sets of tags, resources and users and each edge connect three vertices (a vertex from each set). A fundamental step in extracting semantics from folksonomies is to transform the tripartite graph into a bi-graph of tags and resources to reveal their inter-relationships.

Map-based and geo-enabled collaborative applications on Web 2.0 generate geo-folksonomies - folksonomies with a geographical dimension - using geographic places as resources. Applications such as Google Maps[3], Tagzania[4], Openstreetmap[5] and Geonames[6] allow users to create place resources and give

[3] http://maps.google.com
[4] http://www.tagzania.com
[5] http://www.openstreetmap.org
[6] http://www.geonames.org

them spatial (such as longitude and latitude) and thematic attributes (such as place name and description). These applications are becoming increasingly popular and currently store millions of references to geographical places. On the other hand, geo-enabled applications such as Flickr[7] and Wikipedia[8] allow users to create Web resources - images in Flickr and Web pages in Wikipedia - and "geo-tag" those resources by assigning them a spatial location or place reference.

Pre-processing folksonomies to enhance the results of folksonomy analysis methods [15,14,9] has been tackled in the literature on different scales. One scale was to process the tags by removing the stop words and stemming the tags such in [18]. Another scale of pre-processing was to enhance the structure of the folksonomy such in [11], which introduced four different aggregation methods to enrich the folksonomy structure, by adding weights that represent the level of users agreement on resource-tag pairs. All the above work targets the general folksonomies, which can be used in geo-folksonomies as well. However, up to our knowledge, there is no research work covers the problem of pre-processing the resources in geo-folksonomies, which is the problem covered by this research work.

3 Problem Definition

The term 'folksonomy' (from folk and taxonomy) was coined by Vander Wal in 2004 [19]. Folksonomy can be seen as a user generated index to classify and organize the Web resources. In social bookmarking applications, a folksonomy tuple, also called tag application [4], is created every time a user tags a Web resource. It can be formalized as follows:

$$F = \{S, U, R, T, \pi\} \tag{1}$$

Where S is the social bookmarking application that hosts the folksonomy tuple, U is a User, R is a Resource, T is a Tag and π is the time stamp of the creation of the tuple.

Users are usually identified by IDs. A user ID is always represented by a unique user name chosen by the user. Resources are Web documents such as Web pages, images or PDF files. Each resource can be located using a unique URI. Tags are single keywords supplied by users to describe and index the resources. The social bookmarking applications store the creation date of the folksonomy tuples which can be used later for temporal analysis. For simplicity, the folksonomy tuple can be redefined as:

$$F = \{U, R, T\} \tag{2}$$

where multiple resources and temporal analysis are not considered in this work.

Each tuple in the folksonomy represents a relation between a user, a resource and a tag. A simple query on such data can answer questions such as: what are the most used tags for annotating resources, or, who is the most active

[7] http://www.flickr.com
[8] http://www.wikipedia.org

user. These are typical data retrieval questions that can be answered by simple database queries. However, questions such as, what are the most related tags to the tag 'Cardiff', are more complicated where the answer requires co-occurrence analysis of tags to calculate tag similarity.

Web resources, e.g. documents, can be easily located and identified using URIs[9], where each document has a unique address on the World Wide Web. In social bookmarking applications, two users are considered to be tagging the same Web resource only if the resources they are tagging have the same URI.

Unlike Web resources, place resources in geo-social bookmarking applications can't be easily identified and located on the World Wide Web, as such resources are not represented as Web documents and consequently don't have URIs. Typically, place resources are associated with spatial attributes for representing the place location and thematic attributes, e.g. a place name and a place type, encoded as free text. Hence, two users can be considered to be tagging the same place resource only if the resources they are tagging are 'spatially close' and have similar names.

The spatial location of place resources is acquired via a map-based user interface. Users click on the location of the place they want to tag and the mouse location on the applet is translated to the corresponding longitude and latitude. While tagging a new place, the map interface does not reveal any places created by other users in the same area and thus a place resource can be created and tagged a multiple of times by different users. The same place may be given different names. For example, both "Cardiff University" and "Cardiff uni." is used to refer to the same place by different users. Also, both instances may not be digitized at the exact same spatial location.

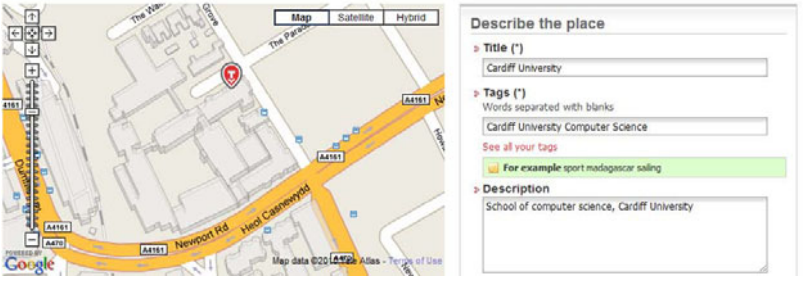


Fig. 1. User interface for creating a new place resource in Tagzania

Figure 1 shows the map-based user interface of Tagzania.com used for tagging new place resources. The map-based interface allows the current user to click on the map to locate the place and add required attributes such as the place name, tags and description in free-text from.

As discussed above, a real-world place entity can be referred to using more than one place resource/instance in the geo-folksonomy. These redundant place

[9] Unique Resource Identifier.

resources are not linked and can thus lead to an increased uncertainty in the information content of the folksonomy and will adversely affect the result of any co-occurrence analysis applied on it.

4 Identifying Redundant Place Resources

Generally, two place instances r_1 and r_2 refer to the same real world place entity if (1) they have the same spatial location and (2) they have the same place name. In this work, exact matching methods are not appropriate and fuzzy similarity matching is used. To identify the redundant place resources in a folksonomy, two stages of analysis should be used:

- Spatially cluster places that are in close proximity to each other.
- In each cluster, identify resources that have similar place names.

4.1 Spatial Clustering

The main objective of using a spatial similarity measure is to find place instances that are in close proximity to each other. This can be achieved by using cluster analysis algorithm or by consulting external reverse geo-coders to assign a unique area code for each place resource, and then area codes can be used as clusters identifiers.

Cluster analysis methods are unsupervised learning methods which aim to group a set of observations into subsets if they are similar in some sense. The feasibility of using cluster analysis is tested in this work by testing Quality Threshold (QT) Clustering [7] on a subset of the folksonomy data. QT is seen as the best candidate algorithm for this work as it does not require the number of clusters to be priori defined.

The Yahoo Where on Earth ID (WOEID) and postcode reverse geo-coders are the external data sources considered here to cluster the place resources. The WEOID web service provides a unique identifier, by reverse geo-coding APIs, for every location on earth. It represents the closest street to any given spatial coordinate. Hence, place instances with the same WOEID are spatially close as they are close to the same street.

Table 1 shows the details of a subset of place resources that represent the place "Big Ben" in London. Each resource is shown with its WOEID, postcode and the calculated QT cluster ID. As shown in the table, all the "Big Ben" instances are grouped into one WOEID while the postcode divides the resources into two groups. Postcode failed as each postcode value represents a very tight area of buildings while the resources in the dataset are not that close. The table also shows the place resources are grouped into one group by using the district level of the postcodes. Also, it shows that QT clustering algorithm could successfully cluster the place resources in this dataset.

Although using district level of postcodes and WOEIDs can produce the same results, the usage of postcodes is only limited to UK. In addition, although the

Table 1. Postcodes and WOEIDs of Big Ben place resources

ID	WOEID	Postcode	District Level PC	QT cluster ID
31758	44417	SW1A 0AA	SW1A	ID0
31759	44417	SW1A 0AA	SW1A	ID0
31760	44417	SW1A 2JR	SW1A	ID0
31761	44417	SW1A 2JR	SW1A	ID0
31762	44417	SW1A 0AA	SW1A	ID0
49775	44417	SW1A 2JR	SW1A	ID0
49776	44417	SW1A 0AA	SW1A	ID0
49777	44417	SW1A 0AA	SW1A	ID0

QT clustering algorithm also can produce the same results of WOEID, the time complexity of running this algorithm limits using it on large datasets. Thus, WOEIDs were found to be more suitable in the scope of this work as the geo-folksonomy dataset used for the experiments is not limited to UK.

4.2 Textual Clustering

After grouping place instances that are spatially similar, a further similarity check can be applied to find place instances with similar names within that group. A simple text similarity method based on "Levenshtein Distance" [10] is used here to find similar place names. The Levenshtein Distance between two strings is the minimum number of edits (insertion, deletion, or substitution) needed to transform the first string to the second string. The text similarity method can be defined by the following equation:

$$\sigma_t(n(r_1), n(r_2)) = 1 - \frac{LD(n(r_1), n(r_2))}{Max((n(r_1), n(r_2)))} \tag{3}$$

where LD is the Levenshtein Distance function and Max is the maximum length of the names of the two place instances.

4.3 Clustering Place Resources

Figures 2 and 3 show two views of an area around "Big Ben" in London. Figure 2 shows the place resources, grouped in colour-coded clusters, after applying the spatial clustering method. Figure 3 shows the same place resources, in different clusters, after identifying similar resources using both the spatial and textual clustering methods. The box in Figure 2 bounds the place resources with a unique WOEID including the place Big Ben in the first view. In Figure 3 the smaller box identifies the place resources which all refer to the place Big Ben. The first box spans an area of 750 m. across its diagonal, where as in second box the area shrinks to around a 1/3 of this size. This demonstrates the quality and accuracy of the location of these place resources.

By identifying redundant place resources, resources that references the same place in the real world are grouped into place clusters and the enriched

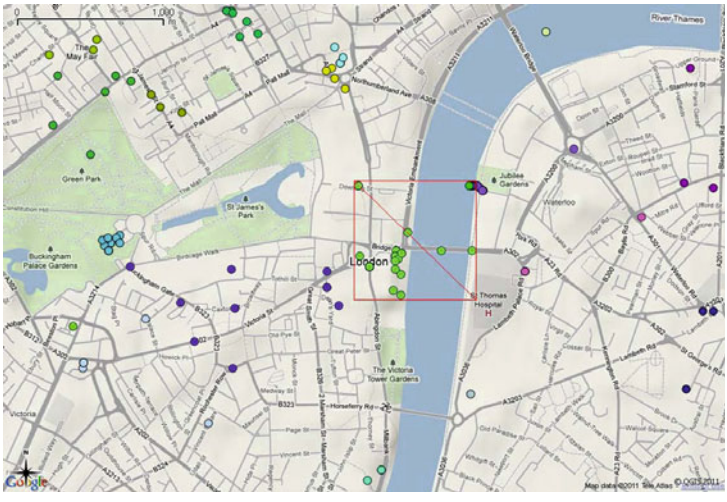


Fig. 2. place resources spatially clustered using WOEID

geo-folksonomy tuples can be defined as $Geo - F = \{U, R, PC, T\}$ where PC is a cluster of similar place resources, of which R is one.

The spatial and thematic attributes for the place clusters can be defined using the instances in those clusters. Different methods can be applied. For example, the spatial location of a place cluster can be computed as either the location of the most central place instance in the cluster, or the centroid of the polygon enclosing the set of place instances in the cluster. Similarly, the place name associated with the cluster can be chosen as the most commonly used name in the cluster, etc.

5 Experiment and Evaluation

5.1 Experiment

The dataset used for evaluation is a geo-folksonomy collected using a crawler software - developed for this work - designed to scan pages on Web 2.0 mapping sites and to index the geo-folksonomies stored on those pages. In this experiment, the crawler was set to process the site: www.tagzania.com. The collected geo-folksonomy dataset includes 22,126 place instances in the UK and USA, 2,930 users and 11,696 distinct tags. The number of geo-folksonomy tuples collected is 65,893. In addition, 10,119 unique WOEID values - cover the entire place instances in the dataset - were obtained by calling Yahoo's reverse geocoding APIs which are exposed via Flickr's Web service.

The method proposed in section 4.3 was used to enrich the collected geo-folksonomy. The text similarity threshold β was set to 0.8 (this was found to be sufficient for this experiment). After applying the method, the number of clusters (unique places) decreased to 19,614. Hence, the method resulted in merging 2,512 place instances (around 11% of the total number of place resources).

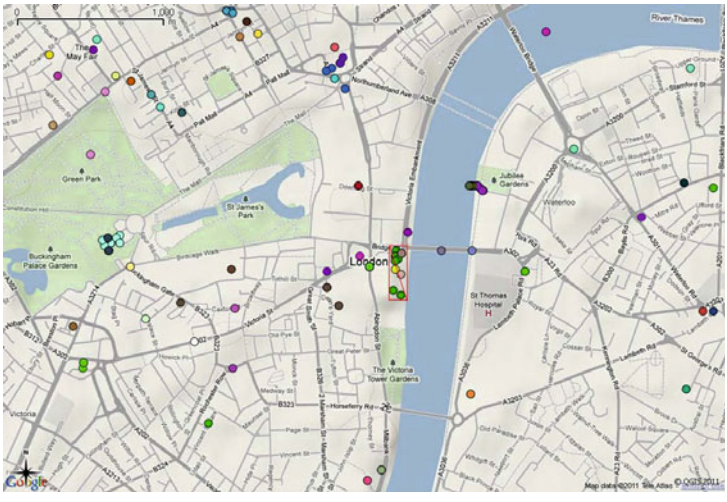


Fig. 3. place clusters after applying spatial and textual clustering

5.2 Measuring the Uncertainty

In order to measure the uncertainty of the Folksonomy Shannon's information gain [16] is used as follows:

$$I(t) = -\sum_{i=1}^{m} \log_2 p(x_i) \tag{4}$$

Where t is any given tag. m is the number of places annotated by the tag t and $p(x_i)$ defined by:

$$P(x) = \frac{w_{t,x}}{\sum_{j=1}^{m} w_{t,x_j}} \tag{5}$$

Where w is equal to the weight of the link between t and place x. The value of $p(x)$ will increase if the number of user votes increases and vice versa, high values of $p(x)$ indicates a high degree of certainty (lower information gain) of using tag t with place x.

5.3 Evaluation Results

To understand the density of the spatial groups (considering WOEID as group) it is worth seeing how the place instances are distributed over the WOEIDs. Figure 4 shows the histogram of the number of place instances over WOEIDs; the WOEIDs that group only 2 place instance are 1653 groups, this number drops to 627 (less than half) for the WOEIDs that group only 3 place instance. Again, this number drops to 350 (around half) for the WOEIDs that group only 4 places and so on.

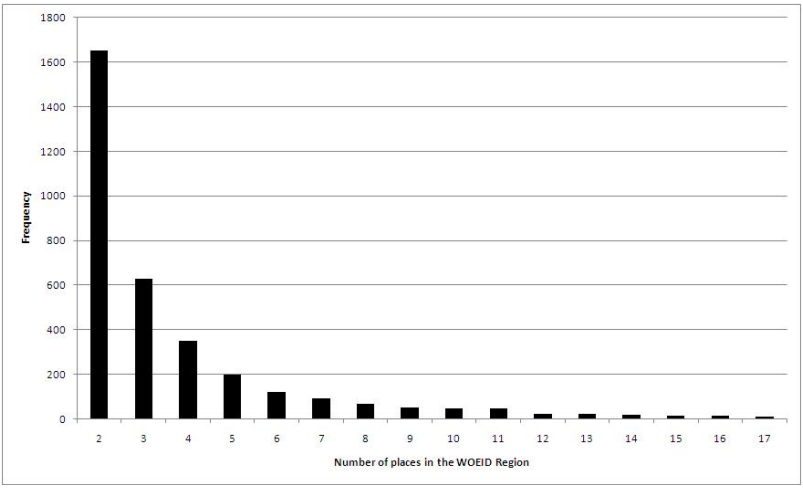


Fig. 4. Histogram of the number of places groupd by WOEIDs

To evaluate the effect of identifying the place instances of the same place concept and build a richer geo-folksonomy that includes user, the information gain is calculated for the Geo-Folksonomy before and after using the proposed method. The results show that the information gain before is 4011.54 and after is 3442.716 which is around 14% reduction in the uncertainty.

The uncertainty reduction is caused by the regions that have increased place annotation activities, in which it is likely to have multiple users annotating the same place using similar names. Table 2 shows a sample of WOEID regions, the number of places in each region and the information content before and after using the proposed method.

Table 2. Information content (Uncertainty) sample

WOEID	Instances	(I) Before	(I) After	Reduction %
2441564	106	126	115	8.7%
2491521	86	11.7	6.9	41%
2352127	83	129	119	7.8%
2377112	80	23.6	18.8	20.3%
2480201	68	24.6	21.6	12.2%

6 Discussion and Future Work

The geo-folksonomy generated in Web 2.0 mapping-based social bookmarking application has introduced a different type of resource on the Web, namely, geographic places. However, these resources cannot be uniquely identified even within the same social bookmarking application. Technically, the cause of this problem is the user interface used to annotate the places. Exposing existing

places resources already annotated by users to new users might address this problem. However, this is controversial and is not adopted by current applications, as this may influence the tagging behaviour of those new users.

An alternative solution is to create a centralized Web service that is responsible for creating and maintaining unique identifiers for place entities. Whenever any social bookmarking application needs to create a new place instance, it can query the centralized service with attributes such as name and location and get a unique identifier for this place. Yahoo's WOEID Web service is an example of this centralized service. However, Yahoo's WOEID Web service generates unique IDs for collection of places up to street level and not to the level of individual places.

Despite creating an overhead, where social bookmarking applications need to integrate with the centralised service to maintain the unique IDs, this solution will support standardised reference to place instances across different applications and therefore can allow the linking and integration of multiple resources.

The methods used for identification and clustering place instances in this work were shown to be successful in removing a significant percentage of redundant place instances. Moreover, the number of links between tags and place resources was dropped from 65,893 to 62,759 where each link is weighted by the number of users who agreed to use the tag-resource pair it connects.

References

1. Almeida, A., Sotomayor, B., Abaitua, J., López-de-Ipiña, D.: folk2onto: Bridging the gap between social tags and ontologies. In: 1st International Workshop on Knowledge Reuse and Reengineering Over the Semantic Web (2008)
2. Auer, S., Lehmann, J., Hellmann, S.: LinkedGeoData: Adding a Spatial Dimension to the Web of Data. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 731–746. Springer, Heidelberg (2009)
3. Hao Chen, W., Cai, Y., Fung Leung, H., Li, Q.: Generating ontologies with basic level concepts from folksonomies. ICCS 2010 1(1), 573–581 (2010)
4. Farooq, U., Kannampallil, T., Song, Y., Ganoe, C., Carroll, J., Giles, L.: Evaluating tagging behavior in social bookmarking systems: metrics and design heuristics. In: Proceedings of the 2007 International ACM Conference on Supporting Group Work, pp. 351–360. ACM (2007)
5. Twaroch, F.A., Jones, C., Abdelmoty, A.: Acquisition of Vernacular Place Footprints from Web Sources. In: Baeza-Yates, R., King, I. (eds.) Weaving Services and People on the World Wide Web, pp. 195–214. Springer, Heidelberg (2009)
6. Golder, S.A., Huberman, B.A.: Usage patterns of collaborative tagging systems. Journal of Information Science 32(2), 198–208 (2006)
7. Heyer, L., Kruglyak, S., Yooseph, S.: Exploring expression data: identification and analysis of coexpressed genes. Genome Research 9(11), 1106 (1999)
8. Heymann, P., Garcia-Molina, H.: Collaborative creation of communal hierarchical taxonomies in social tagging systems. Technical Report 2006-10, Stanford InfoLab (April 2006)

9. Lee, S., Won, D., McLeod, D.: Tag-geotag correlation in social networks. In: Proceeding of the 2008 ACM Workshop on Search in Social Media, pp. 59–66. ACM (2008)
10. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics Doklady 10, 707–710 (1966)
11. Markines, B., Cattuto, C., Menczer, F., Benz, D., Hotho, A., Stumme, G.: Evaluating similarity measures for emergent semantics of social tagging. In: Proceedings of the 18th International Conference on World Wide Web, pp. 641–650. ACM, New York (2009)
12. Mathes, A.: Folksonomies-cooperative classification and communication through shared metadata. In: Computer Mediated Communication, LIS590CMC (Doctoral Seminar), Graduate School of Library and Information Science. University of Illinois Urbana-Champaign (2004)
13. Mika, P.: Ontologies are us: A unified model of social networks and semantics. In: Web Semantics: Science, Services and Agents on the World Wide Web, vol. 5 (2007)
14. Rattenbury, T., Good, N., Naaman, M.: Towards automatic extraction of event and place semantics from flickr tags. In: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 103–110. ACM, New York (2007)
15. Schmitz, P.: Inducing ontology from flickr tags. In: Collaborative Web Tagging Workshop at World Wide Web, Edinburgh, Scotland (2006)
16. Shannon, C.: A mathematical theory of communication. ACM SIGMOBILE Mobile Computing and Communications Review 5(1), 55 (2001)
17. Tsui, E., Wang, W.M., Cheung, C.F., Lau, A.S.M.: A concept-relationship acquisition and inference approach for hierarchical taxonomy construction from tags. Inf. Process. Manage. 46(1), 44–57 (2010)
18. Van Damme, C., Hepp, M., Siorpaes, K.: Folksontology: An integrated approach for turning folksonomies into ontologies. Bridging the Gap between Semantic Web and Web 2, 57–70 (2007)
19. Wal, T.V.: Folksonomy (2007), http://www.vanderwal.net/folksonomy.html
20. Wu, H., Zubair, M., Maly, K.: Harvesting social knowledge from folksonomies. In: Proceedings of the Seventeenth Conference on Hypertext and Hypermedia, p. 114. ACM (2006)

Generating Semantic-Based Trajectories for Indoor Moving Objects

Huaishuai Wang, Peiquan Jin, Lei Zhao, Lanlan Zhang, and Lihua Yue

School of Computer Science and Technology, University of Science and Technology of China
jpq@ustc.edu.cn

Abstract. This paper presents a novel method to generate semantic-based trajectories for indoor moving objects. Indoor moving objects management has been a research focus in recent years. In order to get the trajectory data of indoor moving objects, we have to deply numerous positioning equipments, such as RFID readers and tags. In addition, it is a very complex and costly process to construct different environment settings for various indoor applications. To solve those problems, we propose to use virtual positioning equipments, e.g. RFID readers and tags, to simulate indoor environment. Furthermore, we present a semantic-based approach to generating trajectories for indoor moving objects, which takes into account the type of moving objects, the relationship between moving objects and locations, and the distribution of the trajectories. Compared with previous approaches, our method is more realistic for the simulation of indoor scenarios, and can provide useful trajectory data for further indoor data management analysis. Finally, we design and implement a tool for the generation of semantic-based trajectories for indoor moving objects, and conduct a case study to demonstrate its effectiveness. The results show that it can generate semantic-based trajectories for indoor moving objects according to different parameters and semantic settings.

Keywords: indoor space, trajectory data, moving objects, simulation.

1 Introduction

Indoor space has been a new research hot topic, as many people work and live in indoor environment in most of their lives. Analyzing the moving patterns of indoor moving objects is very useful to realize personalized customer relationship management, hot indoor locations determination, as well as monitoring indoor moving objects. However, it is very hard to perform studies towards indoor moving objects data management, due to the lack of experimental data. In order to get the trajectory data of indoor moving objects, we have to deply numerous positioning equipments, such as RFID readers and tags, which takes lots of money, time, and human recourses. In addition, it is a very complex and costly process to construct different environment settings for various indoor applications.

L. Wang et al. (Eds.): WAIM 2011 Workshops, LNCS 7142, pp. 13–25, 2012.

Therefore, in this paper we aim at providing simulated data for indoor moving objects management and particularly focus on the simulation of trajectory data for indoor moving objects. Previous simulation approaches for indoor moving objects are lack of the consideration on moving semantics. On the other side, most indoor moving objects show some specific patterns when they move in indoor space. For example, for most employees, when they enter into the working building, they usually directly move to their office, and then they may go to the rest room after working for some time or go to the boss office for discussion. This implies that moving objects usually have their own moving patterns in indoor space, which should be emphasized when we simulate the trajectory data for indoor moving objects.

In this paper, we use virtual positioning equipments, namely virtual RFID readers and tags, to construct the virtual indoor environment, and then generate semantic-based trajectories for indoor moving objects according to some specific rules that are used to control the moving patterns of moving objects. The main contributions of the paper can be summarized as follows:

(1) We propose a semantic-based method to generate trajectories for indoor moving objects.Compared with previous works, our approach is able to generate realistic moving patterns for indoor objects (Section 3).

(2) We introduce three types of semantics into the generation of indoor trajectories, which are the type of moving objects, the relationship between moving objects and locations, and the distribution of trajectories (Section 4).

(3) We implement a tool called IndoorSTG to generate the semantic-based trajectories for indoor moving objects, and perform a case study to demonstrate its effectiveness (Section 5).

2 Related Work

Indoor space has received much attention in recent years. Previous work related with indoor space focused on the modeling of indoor space. The indoor space models can be divided into three categories according to the different ways to describe indoor objects, which are the *object feature model*, the *geometric model* and the *symbolic model*. Among them, the object feature model mainly expresses the properties of indoor space and the relationship between operations and types. In the literature [6], the authors used the UML-based class model, CityUML/IndoorML, to describe the relationship among objects in indoor space. In [7], an ontology-based model named ONALIN was proposed for the navigation in the indoor space. The geometry model concerns about the geometric representation of indoor space, which is mainly used to visualize the indoor space. The 2D-3D hybrid model proposed in [8] supports the visualization of indoor space and the navigation in indoor space. The prismatic model

in [9] can well analyze the topology of indoor space. A topology-based semantic model was presented in [10], in which the indoor space is represented as a single set of objects for the analysis of indoor space. The lattice-based semantic model [11] used lattice structure to represent the indoor space, which is mainly used for the navigation in indoor space.

There are some related works on data generation for indoor space [1-5]. In [1], the authors proposed some principles of how to simulate the indoor environment and some evaluation rules to be used to evaluate the model functions. The simulated experimental data used by [4, 5] is randomly generated. They did not consider the moving patterns of indoor moving objects so that the data set has limited use in indoor space analysis. In [2, 3], a simulation tool for generating RFID streaming data was introduced. Although they take into account the distribution of the trajectories, this approach used constant probabilty for each edge in a trajectory. This is not realistic, because indoor moving objects have different moving patterns (or potential locations) when thet are in different locations. Additionaly, the correlation between objects and locations is neglected in previous work. Usually, the objects are of certain relationship with some interested locations, such as PhD students and their laboratories, professors and their offices. Thus that relationship should be taken into consideration when generating trajectories for indoor moving objects.

3 Overview of IndoorSTG

3.1 Design Considerations of IndoorSTG

In order to define realistic moving patterns and generate various trajectories for different indoor moving objects, we need to make the data generation process more adaptive and flexible. For this purpose, we introduce some parameters in this paper to well define the generation actions. The parameters are defined in Table 1. The symbol N_R represents the number of RFID readers deployed in the indoor environment, whose domain is between one and a maximum value that can be set according to your requirements. The symbol *Loc* represents the <*X*, *Y*> location of a RFID reader, where the value of *X* or *Y* is set according to the screen coordinate system. The notion R_R represents the coverage of the reader, i.e., the sensing radius of the reader, whose value can be ranged from 25cm to 50cm, or even a larger one. The notion N_M represents the number of moving objects. S_p represents the speed of moving objects that move in the virtual environment, which can be a fixed value or a randomly-selected value from a given range. The notion T_{Tra} represents the travel time that the moving objects move from one reader to another reader, which is calculated by Dis/S_p (*Dis* is the distance between those two readers). The notion T_{sta} represents the stay time period of the moving objects that stay within the coverage range of the reader, which is set according to the current location described in Table 2. Moreover, we can

also define other parameters. Taking the sampling period of the reader as an example, although this feature can well simulate the functionality of a RFID reader, it will result in many redundant data which have to be filtered when used in applications.

As different objects have different moving patterns, in this paper we divide moving objects into several types, such as teachers, students and visitors. Besides, in a typical indoor environment, a moving object is usually associated with some specific locations. For instance, a student in a lab building is associated with his or her laboratory room and supervisor's office. Hence, when a moving object enters into the indoor environment, we assume that it will stay for a long period at its associated locations and stay for a little time at other locations. Meanwhile, we assume that moving objects tend to moving among their associated locations. As a result, the locations are classified into the types listed in Table 2. Here, the *primary locations* represent the focused locations of a moving object, such as working room, where the moving object may stay for a long time, and has a larger probability to firstly arrive at this location. The *secondary locations* represent the locations that are related with the moving object. For example, for students, the secondary locations may include the supervisor's office, while for teachers the secondary locations may include the laboratory that his or her students stay in. Because of this relationship, the students (teachers) have a larger probability to arrive teacher's office (laboratory). The *thirdly locations* can be also named *service locations*, such as ATM, rest room, and so on, which may have a small probability to reach during the travel of a moving object.. Taking the rest room as an example, you rarely directly move into the rest room when you firstly enter into the office building. The types from 0 to 2 can be called *interested locations*. The *negligible locations* are the rooms that are not included in the above three type of locations, which have a smallest probability during the journey of a moving object.

Table 1. Parameters to configure the virtual indoor environment

Parameter	Description
N_R	The number of readers for testing
Loc	The location of the RFID
R_R	The coverage of the reader
N_M	The number of Moving object
T_{Tra}	The travel time between two readers
T_{Sta}	The stay time at the cover range of the reader
S_p	The speed of moving object

Table 2. Locations Classification

Type	Description	Probability
0	Primary Locations	[0--1]
1	Secondary Locations	[0--1]
2	Thirdly Locations	[0--1]
3	Negligible Locations	[0--1]

3.2 Architecture of IndoorSTG

In this section, we discuss about the architecture of IndoorSTG which is shown in Fig.1. The architecture is designed based on above explained requirements and functions.

In order to generate the trajectory data of moving objects in indoor environment, positioning sensors such as RFID readers and WLAN adapters should be first deployed. In our system, we use the RFID readers as the basic sensors, as at present RFID sensors are very popular in indoor positioning. While the RFID readers are deployed in the indoor environment, we attach a RFID tag to each moving object so that they can be captured when passing by the RFID readers. Therefore, we use two virtual devices, namely *Virtual RFID Controller* and *Virtual Tag Controller*, to simulate RFID readers and moving objects respectively. When a moving object identified by a unique virtual tag enters into the range covered by some RIFD reader, we generate a record formed *<ID, tag_ID, reader_ID, enter_time, leave_time, move_time>* to record the movement. In addition to the *Virtual RFID Controller* and the *Virtual Tag Controller*, the virtual environment also contains a *Virtual Location Controller*, which is used to maintain the information about the *interested locations* listed in Table 2. The *T-L Relation Controller* in Fig.1 is designed to define the

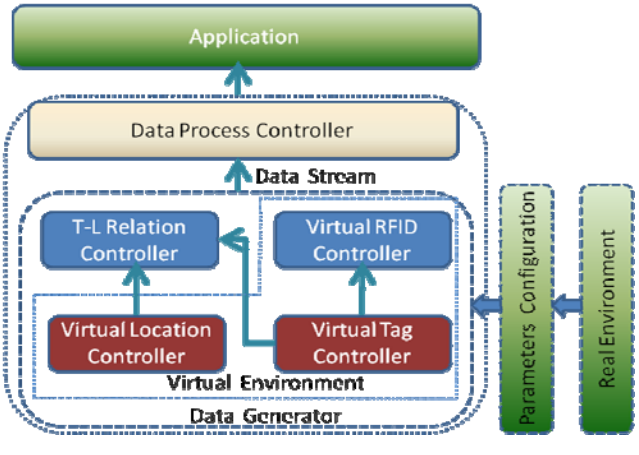


Fig. 1. Architecture of IndoorSTG

relationships between moving objects (i.e., virtual tags) and interested locations. The last component of IndoorSTG, the *Data Processing Controller*, is the user interface to generate semantic-based trajectories, which also offers flexible configuration on the parameters that are necessary to run IndoorSTG. The final trajectory data generated is placed in a file and can be used for further analysis, such as moving pattern classification, hot trajectory determination, and moving objects clustering.

4 Introducing Semantics in IndoorSTG

4.1 Correlation between Objects and Locations

Indoor moving objects usually have different moving patterns; therefore it is important to first make a classification on those moving objects. First, we divide moving objects into several types. Taking the office building of our department as an example, the moving objects can be divided into three types, such as teachers, students, and visitors. Since IndoorSTG aims at simulating the moving patterns of indoor moving objects, we simply divide all the moving objects (virtual tags) into three sub-sets, with each sub-set contains a certain number of moving objects. For instance, suppose that there are n moving objects, then each sub-set will contain $\alpha*n$, $\beta*n$, and $\gamma*n$ moving objects respectively, where $\alpha+\beta+\gamma=1$.

Secondly, we introduce semantics into locations and divide all the locations into four types, namely primary location, secondary locations, thirdly locations, and negligible locations, as mentioned in Table 2.

Finally, we use the algorithm *OLCorrelation* (shown in Fig.2) to construct the relationships between moving objects and locations. First, we assume that each object is associated with some locations among the whole four types of ones (Line 1 to 18). It means each object has a set of primary locations, a set of secondary locations, and so on. In Fig.2, for simplification we assume that all the moving objects are classified into three types, i.e., students, teachers, and visitors. However, those types can be adjusted according to the specific scenario of the simulated indoor environment. Line 3 to 6 is used to set the locations for students and Line 8 to 11 is for the locations of teachers. For visitors, we set his or her primary and secondary location as NULL (Line 13 to 14), which means his or her moving pattern is random. For students and teachers, we set the service rooms associated with them, such as ATM and rest room. In Line 19 to 31, we set the location probabilities that the moving objects move towards it. At this time, we must consider the type of moving objects, because they have different probabilities to move towards different types of associated locations. Generally, the probability for primary location is much bigger than the probability for other type.

Algorithm *OLCorrelation /*the Correlation between Objects and Locations*/*
Input: The set of moving objects $P = \{P_1, P_2, \ldots, P_n\}$, where the type of each P_i has been annotated. The set of location $L = \{L_1, L_2, \ldots, L_n\}$, where L has been divided into four types.
Output: The revised set of moving objects $P = \{P_1, P_2, \ldots, P_n\}$, where each P_i has been associated with some locations.
Preliminary: The indoor environment is a six-floored building. All the moving objects are classified into students, teachers, and visitors. However, those types can be adjusted when running IndoorSTG.

```
1:  for each Pi ∈ P { // associate rooms with objects
2:    if Pi is student then{
3:        room ← randomly select a room locating in 1-5th floor;
4:        Pi.setPrimaryLocation (room);
5:        room ← randomly select a room locating in 6th floor
6:        Pi.setSecondaryLocation(room); }
7:    else if Pi .getType is teacher then {
8:        room ← randomly select a room locating in 6th floor
9:        Pi.setPrimaryLocation(room);
10:       room ← randomly select a room locating in 1-5th floor;
11:       Pi.setSecondaryLocation(room);
12:   else { // the type of moving object is visitor
13:       Pi.set PrimaryLocation(NULL);
14:       Pi.setSecondaryLocation(NULL);
15:   }
16:   Pi.setThirdlyLocation(); // set service rooms for each type of objects
17:   if Pi is not visitor then Pi.setNegligibleLocation();
18: }
19: for each Pi ∈ P {
20:   if Pi .getType is student then {//probabilities for each associated location
21:       Pi.setPrimayLocProb(PROB_STU); // for primary locations
22:       Pi.setSecondaryLocProb((1- PROB_STU)*0.8); /
23:       Pi.setThirdlyLocProb((1- PROB_STU)*0.15);
24:       Pi.setNegligibleLocProb((1- PROB_STU)*0.05);
25:   }
26:   if Pi .getType is teacher then {
27:       Pi.setPrimayLocProb(PROB_TEA);
28:       Pi.setSecondaryLocProb((1- PROB_TEA)*0.8);
29:       Pi.setThirdlyLocProb((1- PROB_TEA)*0.15);
30:       Pi.setNegligibleLocProb((1- PROB_TEA)*0.05);
31:    }
32: }
```

Fig. 2. The *OLCorrelation* Algorithm

4.1 Clustering of Movements

In this section, we define the algorithm *GenTrajectory* to generate trajectories for indoor moving objects (as shown in Fig.3).

Algorithm *GenTrajectory*

Input: The set of moving objects $P = \{P_1, P_2, \ldots, P_n\}$, where each P_i has been associated with interested locations. Graph $G=<N, E>$, where $N = \{N_1, N_2, \ldots, N_n\}$, N_i represents a RFID reader, $E = \{E_1, E_2, \ldots, E_n\}$ is a set of edges and $E_i = <N_i, N_j>$ means that the object can move from N_i to N_j.

Output: The trajectory record *<ID, tag_ID, reader_ID, enter_time, leave_time, move_time>* of moving object.

```
1:  for each P_i ∈ P {
2:    while (n > 0) { // for each object, we generate n trajectory records
3:      if P_i is a student or a teacher then {
4:        if now is the working time then { // now is a self-defined time
5:          get the probabilities for P_i to move to the locations associated;
            /* get the Dijkstra min path and return the count of hops
6:          desLoc←P_i.getDesLocation(); // pick up a destination
7:          count←Dijkstra(P_i.curLoc, desLoc);
             /* get the weight of the edge; PROB is the probability that the moving
              object directly move from current location to destination location. */
8:        Prob(weight) = pow(PROB, count-1);
9:          set the weights of the directly-connected edges of the current location;
10:         P_i.curLoc←find the next location for P_i according to the weights;
11:       if P_i.curLoc is not the destination location then {
12:           generate a trajectory record and write it out;
13:           Goto Line 6; // continue to move towards the destination
14:         }
15:       else { // arrive at the destination
16:           generate a trajectory record and write it out;
17:           Goto Line 5; // continue to find another destination location
18:       }
19:     else{ // time to leave the building, i.e., [11:30, 12:00] or after 18:00
20:         count←Dijkstra(P_i.curLoc, entrance); // moving to the entrance
21:         generate trajectory records using the procedures from Line 9 to 16;
22:         n ← n − 1;
23:         if now ∈ [11:30 AM, 12:00 AM] then // tentative leaving
24:           wait until the working time and go to Line 2; // re-enter
25:         else // re-enter in the 7:00 AM of the next day
26:           wait until 7:00 AM in the next day and go to Line 2;
27:     }
27:   else {//visitors
28:     set the weights of the directly-connected edges of the current location;
29:     P_i.curLoc←find the next location for P_i according to the weights;
30:     generate a trajectory record and write it out;
31:     n ← n − 1;}
32:   }
33: }
```

Fig. 3. The *GenTrajectory* Algorithm

First, we check whether at present the moving object that should leave the building (perhaps because of work off or diner time). If yes then the object must leave the building at its current location, otherwise we pick up a destination location with respect to the object type, and use the *Dijkstra* algorithm to get the minimum path from the current location to the destination location. Based on this result, we set the weight of each edge in the path connecting the current location with the destination, and generate a probability to determine the next location until arrive at the destination. At last, we calculate the probabilities of other types of locations in order to get the next destination location, and repeat the above work.

5 Implementation and a Case Study

In this section, we will discuss the implementation of IndoorSTG while the indoor environment is set as the department building of the School of Computer Science and Technology at University of Science and Technology of China. The building has six floors. The teachers' offices are located at the sixth floor, and the students' laboratories are at the other floors. We will explain how to set the parameters for simulating the real indoor environment and how to associate the locations with the objects, and then generating trajectory data. Finally, we present a case study to show the process of generating trajectory data.

5.1 Implementation of IndoorSTG

We deploy 95 RFID readers in the building and the readers are deployed on the room, lift, floor, and passage way. The data structure to describe RFID readers is shown in Table 3. We classify the moving objects into three types, namely teachers, students and visitors. Each object will record the information of locations that associate with it. The data structure of moving objects is shown in Table 4. In order to generate semantic-based trajectory data, we divide the locations into four types, and use the data structure shown in Table 5 to maintain the location information. For the purpose of reducing the complexity of the indoor environment, we assume that objects are moving on the graph which is composed of RFID readers. The data structure of graph is shown in Table 6.

After the above settings, we use the algorithm *GenTrajectory* to generate the trajectory of moving objects. The format of output data is *<ID, tag_id, reader_id, enter_time, leave_time, move_time>*, which means the *tag_id* is detected by the r*eader_id* at *enter_time* and leave the sensing range at *leave_time*, and the *move_time* represents the travel time from the prior location to the current location. Table 7 shows an example, in which the object stays a long time period in the location 59 as this is the primary location, and stay 30 seconds at other locations. At the location 59, the object will stay a random time between 30 minutes and 1 hour, while at the other locations, the object has a probability of 5% to stay a random time period between 1 minute and 5 minutes. In other cases, the object directly passes through the RFID reader, and the spent time is determined by the coverage radius and the moving speed of the object.

Table 3. Data structure to describe RFID readers

Data Item	Description
RID	The number of RFID
<X,Y>	The location of RFID, using screen coordinates
range	The detection range of RFID reader
floorID	The floor number that the RFID locates

Table 4. Data structure to describe moving objects

Data Item	Description
tagID	The number of moving object
roomID	The number of primary location
type	The type of moving object
assRoomID	The number of secondary location
location[]	The structure that stores the information of interesting locations and the probability that the object move to it when the object arrives at one destination location.

Table 5. Data structure to describe locations

Data Item	Description
type	The type of location
locID	The number of location
Prob	The probability that the object move to the locID when the object arrives at one destination location.

Table 6. Data structure to describe graph

Data Item	Description
RFID[]	The information of the location of RFID readers
Edge1[][]	The weight of two nodes in first floor
Edge2[][]	The weight of two nodes in other floors
edgeLift[][]	The weight of two nodes at the location of lift
edgeFloor[][]	The weight of two nodes at the location of corridor

Table 7. An example of trajectory data generated

ID	tag_id	reader_id	enter_time	leave_time	move_time
1	1	50	2011-01-02 16:40:22	2011-01-02 16:40:52	120s
2	1	54	2011-01-02 16:41:08	2011-01-02 16:41:38	16s
3	1	62	2011-01-02 16:43:38	2011-01-02 16:44:08	120s
4	1	59	2011-01-02 16:44:36	2011-01-02 17:17:36	28s

5.2 A Case Study

5.2.1 Scenario of Indoor Space

In this section, we will give the deployment of the first floor, which is shown in Fig.4. The numbers in circles represent RFID readers, and the circles represent the coverage areas of the RFID readers. The RFID readers identify the locations in the building. For example, And the No.1 reader represents the entrance of the building, and the No.2 represents the location of ATM. The symbols like "1-101" represent rooms, and we deploy a RFID reader at the door of each room.

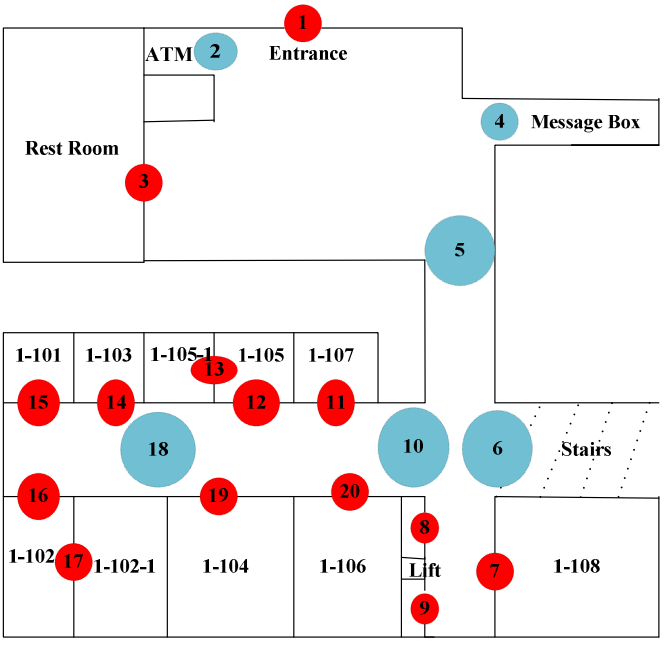


Fig. 4. The deployment of the first floor

5.2.2 Generating Trajectories

In this section, we will discuss about the process of generating trajectory data of one object. First, we define the probability settings for the locations (as shown in Table 8). Because the object may have different moving patterns from the entrance to the destination room at different time, we assign different probabilities to the interested locations of the object. Particularly, we assume that moving objects just entering into the building have different locations preferences from those that are already in the building. For instance, when a student just enters into the department building, his or her first destination location is usually the laboratory. However, when he or she leaves the laboratory, the first destination location is surely not the laboratory itself. Hence, we differ those two cases and use different probabilities for them. Those two probabilities are shown in the right column of Table 8, in which the left is the probability of the location when the object just enters into the building, and the right is the probability of the location when the object has been in a certain room of the

building. Secondly, given that one object just enters into the building from location 1, we first pick up the destination of the object according to the predefined probabilities. Suppose that the destination is location 19, which is the primary location for the object, then we will use the algorithm *Dijkstra* to get the edge count from location 1 to location 19, which is 3, and set the probability (*pow*(0.9,1/3)) of those edges. The current location is 1, and we will set the probability of edges that connect with location 1, which are [1,2,(1-prob)/3], [1,3, (1-prob)/3],[1,4, (1-prob)/3],[1,5,prob] respectively. The object will stay at current location for a random time period ranging from 1 minute to 3 minutes because it is an interested location of the object. Then we randomly select the next location and repeat the above steps until the object arrives at location 19. When the object arrives at location 19, we set the probability of other types of locations with respect to the rules in Table 8. After that, we generate the next destination, and repeat the above processes. Moreover, when the current time is between 11:30 AM and 12:00 AM or after 18:00 PM, the object will leave the building (i.e., moving from its current location to location 1). The partial result is shown in Table 7.

Table 8. The probability setting for locations

Type	Location	Probability
0	roomID	0.9 / 0.8
1	assRoomID	(1-getProb(0))*0.8
2	ATM_ID	(1-getProb(0))*0.05
2	RestRoom_ID	(1-getProb(0))*0.025 / (1-getProb(0))*0.125
2	MessageBox_ID	(1-getProb(0))*0.125 / (1-getProb(0))*0.025
3	NegLoc_ID	(1-getProb(0))*0.05

5.2.3 Discussion

In the process of generating the trajectory data, we use the destination location to determine the overall trajectory of the moving object. By using this method, we can easily express the typical movements in indoor environment. For instance, a student arrives at laboratory now, and then he goes to teacher's office for discussion after some times, and then goes to laboratory again or go to the rest room for a rest. Besides, all the semantic rules in our system can be defined by users, including the types of objects, the relationships between objects and locations, as well as the probabilities of locations. Thus it can generate trajectory data for different application scenarios.

6 Conclusions

Indoor moving objects management will be a new research focus with the development of indoor positioning sensors and the Internet of Things (IoT). We need much experimental data to perform algorithm study for indoor data modeling, indexing, query processing, and data mining. However, at present it is not possible to obtain real data set especially those with specific moving patterns. Therefore, to develop a simulation tool and generate semantic-based moving objects data is a feasible solution in current research on indoor moving objects.

In this paper, we propose a semantic-based method to generate trajectories for indoor moving objects. This method takes into account a lot of semantic rules that are typically implied in indoor enviroment, including the type of moving objects, the relationship between moving objects and locations, and the distribution of the trajectories. We built a simulation tool called IndoorSTG to generate trajectory data for indoor moving objects, and introduce those semantics into the generation process. Furthermore, all the semantic factors are defined by some specfic parameters, which can be adjusted according to the characteristics of different indoor applications.

There are alot of future works that are valuable to be explored, such as to make the simulation tool more realistic, to integrate with out-door environment, and to suit for different types of indoor space including subway and airports.

Acknowledgements. This work is supported by the National High Technology Research and Development Program ("863" Program) of China (No. 2009AA12 Z204), and the USTC Youth Innovation Foundation.

References

1. Park, C., Ryu, W., Hong, B.: RFID Middleware Evaluation Toolkit Based on a Virtual Reader Emulator. In: Proc. of the First International Conference on Emerging Databases, pp. 154–157 (2009)
2. Zhang, H.P., Ryu, W.S., Hong, B.H.: A Test Data Generation Tool for Testing RFID Middleware. In: International Conference on Computers and Industrial Engineering (CIE), Awaji, Japan (2010)
3. Zhang, H.P., Ryu, W.S., Hong, B.H.: A Graph Model Based Simulation Tool for Generating RFID Streaming Data. In: Du, X., Fan, W., Wang, J., Peng, Z., Sharaf, M.A. (eds.) APWeb 2011. LNCS, vol. 6612, pp. 290–300. Springer, Heidelberg (2011)
4. Jensen, C.S., Lu, H., Yang, B.: Indexing the Trajectories of Moving Objects in Symbolic Indoor Space. In: Mamoulis, N., Seidl, T., Pedersen, T.B., Torp, K., Assent, I. (eds.) SSTD 2009. LNCS, vol. 5644, pp. 208–227. Springer, Heidelberg (2009)
5. Jensen, C.S., Lu, H., Yang, B.: Graph Model Based Indoor Tracking. In: Proc. of MDM, pp. 122–131 (2009)
6. Kolbe, T., Goger, G., Plumer, L.: CityGML: Interoperable Access to 3D City Models. In: Proc. of the First Int. Symposium on Geo-information for Disaster Management, pp. 883–899 (2005)
7. Dudas, P., Ghafourian, M., Karimi, H.A.: ONALIN: Ontology and Algorithm for Indoor Routing. In: Proc. of MDM, pp. 720–725 (2009)
8. Kim, H., Jun, C., Yi, H.: A SDBMS-based 2D-3D Hybrid Model for Indoor Routing. In: Proc. of MDM, pp. 726–730 (2009)
9. Kim, J.-S., Kang, H.-Y., Lee, T.-H., Li, K.-J.: Topology of the Prism Model for 3D Indoor Spatial Objects. In: Proc. of MDM, pp. 698–703 (2009)
10. Li, D., Lee, D.L.: A Topology-based Semantic Location Model for Indoor Applications. In: Proc. of ACM GIS, Article No. 6 (2008)
11. Li, D., Lee, D.L.: A Lattice-Based Semantic Location Model for Indoor Navigation. In: Proc. of MDM, Beijing, China, pp. 17–24 (2008)
12. EPC Tag Data Standard, `http://www.gs1.org/sites/default/files/docs/tds/tds_1_5-standard-20100818.pdf` (accessed in May 2011)
13. Rifidi Tag Streamer, `http://wiki.rifidi.org/index.php/Tag_Streamer_User%27s_Guide_1.1` (accessed in May 2011)

HTPR*-Tree: An Efficient Index for Moving Objects to Support Predictive Query and Partial History Query

Ying Fang, Jiaheng Cao, Junzhou Wang, Yuwei Peng, and Wei Song

School of Computer, Wuhan University, Wuhan, China
{fangying,jhcao,ywpeng,songwei}@whu.edu.cn

Abstract. Developing efficient index structures is an important issue for moving object database. Currently, most indexing methods of moving objects are focused on the past position, or on the present and future one. In this paper, we propose a novel indexing method, called HTPR*-tree (History Time-Parameterized R-tree), which not only supports predictive queries but also partial history ones involved from the most recent update instant of each object to the last update time. Based on the TPR*-tree, our HTPR*-tree adds creation or update time of moving objects to leaf node entries. This index is the foundation of indexing the past, present and future positions of moving objects. In order to improve the update performance, we present a bottom-up update strategy for the HTPR*-tree by supplementing three auxiliary structures which include hash index, bit vector, and direct access table. Experimental results show that the update performance of the HTPR*-tree is better than that of the TD_HTPR*- and TPR*-tree. Moreover, the HTPR*-tree can support partial history queries compared with TPR*-tree although the predictive query performance is a bit less.

Keywords: moving object indexing, HTPR*-tree, predictive query, partial history query, bottom-up update strategy.

1 Introduction

Developing efficient index structures is an important research issue for moving object database. Traditional spatial index structures are not appropriate for indexing moving objects because the constantly changing locations of objects requires constant updates to the index structures and thus greatly degrades their performance.

Some index structures have been proposed for moving objects. They can be classified into two major categories depending on whether they deal with past information retrieval or future prediction. Currently, some indices suitable for history and future information retrieval of moving objects have also been studied. However, they are too complicated and could not efficiently handle queries involved from the past to the future.

In general, indexing about past trajectories of moving objects only stores history information from some past time until the time of the most recent position sample (t^o_{mru}) of each object o. However, indexing of the current and anticipated future

L. Wang et al. (Eds.): WAIM 2011 Workshops, LNCS 7142, pp. 26–39, 2012.

positions can only supports the query from the last update time ($t_{lu}=max(t^{o}_{mru}|o\in O)$ to the future. In other words, the history trajectories of moving objects from the most recent update instant to the last update time (or the current time CT) are omited. In order to support the queries involved from the past to the future in moving object databases, current and future positions indexing structure such as TPR-tree [1] and TPR*-tree [2] should be extended to support partial history queries.

The TPR*-tree is the most useful indexing method which indices the current and future position of moving object through Time-Parameterized Rectangles. In the TPR*-tree, partial history trajectories of moving objects which do not update at the last update time are implicit, but they couldn't be queried. In order to query history trajectories in the TPR*-tree, in this paper, we develop a novel index structure which not only supports predictive queries, but also supports partial history queries involved from the most recent update instant of each object to the last update time. This novel index structure is very important to support the queries involved from the past to the future. In order to support frequent update, bottom-up update strategy is also applied to the new index structure.

The main contributions of this paper can be summarized as:

1. We present an new index structure, named History Time-Parameterized R-tree (HTPR*-tree), which takes into account moving object creation time or update time in the leaf node entry, and supports partial history query.
2. We propose a bottom-up update approach referencing the R-tree update technique[3] to support frequent update operation of the HTPR*-tree.
3. We prove through extensive experiments that the update performance of the HTPR*-tree is better than that of the TD_HTPR*- and TPR*-tree.

The organization of this paper is as follows. Section 2 presents related works. Section 3 shows the basic structure of the HTPR*-tree, the corresponding dynamic maintenance, and query algorithms. In section 4, we discuss the bottom-up update algorithm of the HTPR*-tree. Section 5 contains an extensive experimental evaluation, and section 6 is the conclusion of our work.

2 Related Works

A number of index structures have been proposed for moving object database. Most of these index structures are classified into two categories; one of them is to handle past positions or trajectories [4-6], and the other is to handle current and future positions [1,2,7-12]. In addition, some indices suitable for history and future queries of moving objects have also been studied [13,14].

History trajectories indices such as STR-tree and TB-tree [4] are used to index positions for an object only up to the time of the most recent sample. They can play an important role in supporting the queries involved from the past to the future.

The search of indexing methods for current and future positions of moving objects are very challenging. In general, there are three approaches to study the indices for predictive queries. One is indexing the future trajectories of objects moved in d-dimensional space as lines in $(d+1)$-dimensional space [7]. Another is mapping the trajectories to points in a higher-dimensional space which are then indexed [8]. The third is to index the original time-space with parametric rectangles [1,2,9].

By introducing parametric bounding rectangles in R-tree, the TPR-tree provides the capability to answer the queries about current positions and future positions. The TPR*-Tree improved upon the TPR-Tree by introducing a new set of penalty functions based on a revised query cost model. In recent years, based on the B^+-tree, indices for moving objects not only supporting queries efficiently but also supporting frequent updates are proposed. Jensen et al. proposed the B^x-tree [10], which employs space partitioning and data/query transformations to index object positions and has good update performance. Chen et al. also proposed the ST^2B-tree[11], a Self-Tunable Spatio-Temporal B+-Tree index for moving object database, which is amenable to tuning.

Pelanis et al. proposed R^{PPF}-tree [13] to support both the past and the future movement of the objects. The implemented query types are only timestamp ones. Raptopoulou [14] et al. extended the XBR-tree to deal with future prediction as well. But it only can support timestamp queries involved from the past to the future and history window queries.

Our work aims to extend the current and future positions indexing structure to support partial history queries. This new index structure is the foundation of indexing the past, present and future positions of moving objects. In addition, bottom-up update strategy is applied to the new index structure in order to support frequent update.

3 The HTPR*-Tree

In this section, we will discuss a novel index structure called History TPR*-tree (HTPR*-tree). First, we describe the basic structure of the HTPR*-tree. Then, the insertion and deletion algorithms are shown. The query algorithm is given at the end.

3.1 Index Structure

The HTPR*-tree is a height-balanced tree similar to a R-tree. Leaf nodes of the HTPR*-Tree contain entries of the form (*oid, MBR, VBR, st*). Here *oid* is the identifier of the moving object, and *st* is creation or update time of object. *MBR* denotes object extent at time *st,* and *VBR* denotes velocity bounding rectangle of object at time *st*.

For example, a two-dimensional moving object is represented with *MBR* $o_R=\{o_{R1-}, o_{R1+},o_{R2-},o_{R2+}\}$ where o_{Ri-} (o_{Ri+}) describes the lower (upper) boundary of o_R along the i-th dimension ($1\le i\le 2$), and *VBR* $o_V=\{o_{V1-},o_{V1+},o_{V2-},o_{V2+}\}$ where o_{Vi-} (o_{Vi+}) describes the velocity of the lower (upper) boundary of o_R along the i-th dimension ($1\le i\le 2$). Figure 1 shows the MBRs and VBRs of 4 objects a,b,c,d. The arrows (numbers) denote the directions (values) of their velocities. The *MBR* and *VBR* of b are $b_R=\{3,4,4,5\}$ and $b_V=\{1,1,1,1\}$, respectively.

The structure of each non-leaf node entry of R_s is in the form of *(ptr, MBR, VBR, st1, st2).* Here *ptr* is a pointer that points to the child node. *St1* is the minimal creation or update time of moving objects included in the child node pointed by *ptr,* and *st2* is the maximum value compare to *st1. MBR* is the minimum bounding rectangle at *st1*, and *VBR* is the velocity bounding rectangle at *st1*. Figure 2 shows a leaf node *e* including four point objects $\{o_1,o_2,o_3,o_4\}$ in one-dimensional HTPR*-Tree. So, in parent node of *e,* the corresponding entry includes *MBR, VBR*, *st1*, *st2* and *ptr* that points to node *e.* Here *MBR*={3,4}, *VBR*={-0.2, 0.3}, *st1*=2 and *st2*=4.

3.2 Insertion and Deletion

1. Insertion

Because the creation or update time of moving objects is included in leaf node entries of HTPR*-Tree, and non-leaf node entry is different from leaf node entry, the insertion algorithm of HTPR*-Tree is a bit more complicated than that of TPR*-Tree.

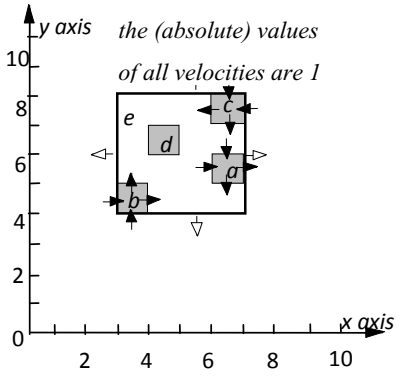


Fig. 1. MBRs and VBRs at reference time

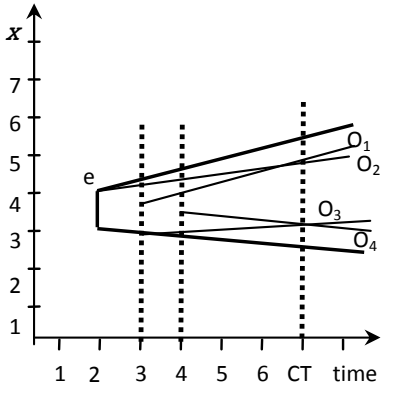


Fig. 2. A leaf node *e* including four point objects

Algorithm 1 shows the insertion of a moving object in the HTPR*-Tree. First, the insertion algorithm initializes two empty re-insertion lists $L_{reinsert1}$ and $L_{reinsert2}$ to accommodate re-insertion moving objects and non-leaf node entries, respectively. Then, the algorithm calls different functions according to whether the root of HTPR*-tree is a leaf node or not. Finally, the algorithm inserts each object in $L_{reinsert1}$ and each entry in $L_{reinsert2}$ to the HTPR*-tree.

Algorithm 2 describes inserting a moving object to a leaf node. However, algorithm 3 describes inserting a moving object to the HTPR*-tree rooted by a non-leaf node. Because the insertion of a node entry can cause node split, the insertion algorithm (algorithm 4) of non-leaf node entry in the HTPR*-Tree is also important.

Algorithm 1. Insert (*r, o*)
/*Input: o is a moving object with *oid, MBR, VBR, st;* r is the HTPR*-tree */
1. root=Root(r) /*achive the root of the HTPR*-tree
2. $re\text{-}inserted_i$=false for all levels $1 \le i \le h-1$ (h is the tree height)
3. initialize two empty re-insertion list $L_{reinsert1}$ and $L_{reinsert2}$
4. if root is leaf node invoke **Leaf Node Insert** (*root*, *o*)
5. else invoke **Non-Leaf Node Insert** (*root, o*)
6. for each data *o'* in the $L_{reinsert1}$
7. if root is leaf node invoke **Leaf Node Insert** (*root*, *o'*)
8. else invoke **Non-Leaf Node Insert** (*root, o'*)
9. for each entry *e* in the $L_{reinsert2}$
10. invoke **Non-Leaf Node Insert_e** (*root, e*)
End Insert

Algorithm 2. Leaf Node Insert (*N*, *o*)
/* **Input**: *N* is the leaf node where object *o* is inserted */
1. enter the information of *o*
2. if *N* overflows
3. if $re\text{-}inserted_0$=false //no re-insertion at leaf level yet
4. invoke **Pick Data Worst** to select a set S_{worst} of objects
5. remove objects in S_{worst} from *N*; add them to $L_{reinsert1}$
6. $re\text{-}inserted_0$=true
7. else
8. invoke **Leaf Node Split** to split *N* into itself and *N'*
9. obtain entry *e* describe node *N'*
10. invoke **Non-Leaf Node Insert_e** (*P*,*e*) /**P* be the parent of *N*/
11. adjust the MBR/VBR/st1/st2 of the node *N*
End Leaf Node Insert (*N*, *o*)

Algorithm 3. Non-Leaf Node Insert (*N*, *o*)
/* **Input**: *N* is the root node of tree rooted by *N* */
1. obtain the son node *N'* of *N* to insert *o* through the path achieve by **Choose Data Path**
2. if *N'* is the leaf node invoke **Leaf Node Insert** (*N'*, *o*)
3. else invoke **Non-Leaf Node Insert** (*N', o*)
4. adjust the MBR/VBR/st1/st2 of the node *N*
End Non-Leaf Node Insert(*N*, *o*)

Algorithm 4. Non-Leaf Node Insert_e (*N*, *e*)
/* **Input**: *e* is non-leaf node entry
1. if *e*.level=*N*.level
2. enter *e* to *N*
3. if *N* overflows
4. if $re\text{-}inserted_i$=false //no re-insertion at i level (e.level)yet

```
5.            invoke Pick Entry Worst to select a set S_worst of entrys
6.            remove entrys in S_worst from N; add them to L_reinsert2
7.            re-inserted_i=true
8.         else
9.            invoke Non-Leaf Node Split to split N into itself and N'
10.           obtain entry e describe node N'
11.           invoke Non-Leaf Node Insert_e (P, e)    /*P be the parent of N*/
12.   else
13.      obtain the son node N' of N through the path achieve by Choose Entry Path
14.      invoke Non-Leaf Node Insert_e (N', e)
15.   adjust the MBR/VBR/st1/st2 of the node N
End Non-Leaf Node Insert_e(N, e)
```

Similiar to that in the TPR*-tree, algorithm *Choose Path* in the HTPR*-tree aims at finding the best insertion path globally with a minimum cost increment (minimal increase in equation 1). If a moving object is inserted, *Choose Path* is instantiated by *Choose Data Path*, and non-leaf node entry inserting calls *Choose Entry Path.* Because the creation or update time of moving objects is included in leaf node entries, the enlarge of entry (caused by inserting a moving object in HTPR*-tree node) involves history information, and the enlarged entry can support history query. This is the major difference between HTPR*-tree and TPR*-tree. Of course, the predictive query performance is somehow less than that of TPR*-tree.

The query cost model of the HTPR*-tree is the average number of node accesses for answering query q:

$$Cost(q)=\sum_{\text{every node } N} A_{SR}(N', q_T) \quad (1)$$

where N is the moving rectangle (interval as for one-dimensional object) representing a node, N' is the transformed rectangle (interval as for one-dimensional object) of N with respect to q, and $A_{SR}(N',q_T)$ is the extent of region swept by N' during q_T.

Figure 3 shows two leaf nodes *e1* and *e2* that are sons of the root *e0*. Here the entry corresponding to *e1* is {*pt1,*{3.2,4},{-0.2,0.3},2,3}, and the entry to *e2* is {*pt2,*{5,6},{0,0.4},3,4}. Consider the insertion of point object O_6={ O_6,4.6,0.5,7} at current time 7. *Choose Data Path* returns the insertion path with the minimum increment in equation 1. The cost increment is 1.2 and 0.9 when *o6* is inserted to *e1* and *e2,* respectively. Figure 4 describes insertion *o6* to *e2*, which is the best insertion node.

Insertion to a full node N generates an overflow, in which the HTPR*-tree uses *Pick Worst* algorithm that selects a fraction of the entries from the node N and re-inserts them. *Node Split* algorithm splits a full node N into *N1* and *N2*. The split algorithm selects split axis and split position minimizing equation 2:

$$\Delta A_{SR}=A_{SR}(N_1',q_T)+A_{SR}(N_2',q_T)-A_{SR}(N',q_T) \quad (2)$$

Another important difference between the insertion of HTPR*-tree and that of TPR*-tree is that the *MBR*, *VBR*, *st1* and *st2* of node N after inserting have to be modified.

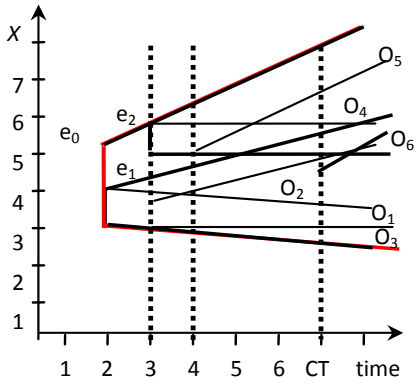


Fig. 3. Insert a moving point object *o6*

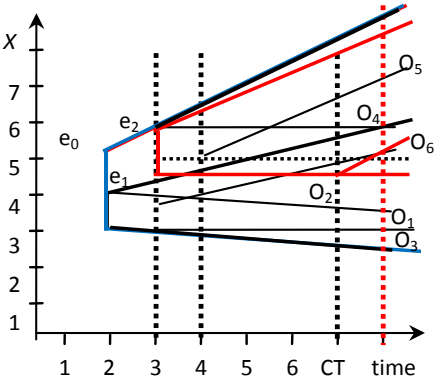


Fig. 4. Insert *o6* to node e2

2. Deletion

Algorithm 5 describes deleting a moving object in HTPR*-Tree. To remove an object *o,* the deletion algorithm first identifies the leaf node that contains *o*. In algorithm 5, two empty re-insertion lists $L_{reinsert3}$ and $L_{reinsert4}$ are initialized to accommodate re-insertion leaf node entries (moving objects) and non-leaf node entries, respectively.

Algorithm 5. Delete (r, **o**)

/*Input: o is a moving object with *oid, MBR, VBR, st*; r is the HTPR*-tree */
1. root=Root(r) /*achive the root of the HTPR*-tree
2. initialize an empty re-insertion list $L_{reinsert3}$ and $L_{reinsert4}$
3. if root is leaf node invoke **Leaf Node Delete** (*root*, *o*)
4. else invoke **Non-Leaf Node Delete** (*root, o*)
5. for each data *o'* in the $L_{reinsert3}$
6. invoke **Non-Leaf Node Insert** (*root*, *o'*)
7. for each entry *e* in the $L_{reinsert4}$
8. invoke **Non-Leaf Node Insert_e** (*root*, *e*)

End Delete (r, **o**)

Deletion of moving object in leaf node *N* may generate an underflow, in which case the HTPR*-tree removes all objects in node *N* to $L_{reinsert3}$, and deletes entry *e* describes *N* in parent node *N'*. If moving object *o* is deleted in the HTPR*-tree root by non-leaf node *N*, the deletion algorithm calls *Leaf Node Delete* or *Non-Leaf Node Delete* in all son node *N'* of *N* until *o* is deleted. In algorithm *Leaf Node Delete* or *Non-Leaf Node Delete,* if deletion of moving object *o* in the HTPR*-tree root by node *N* changes node *N*, adjustment is needed from parent node *N'* of *N* to root.

3.3 Search Procedure

The HTPR*-tree supports three kinds of predictive queries: timeslice query $Q=(R, t)$, window query $Q=(R, t_1, t_2)$, and moving query $Q=(R_1, R_2, t_1, t_2)$. At the same time, the HTPR*-tree supports partial history query. Figure 5 describes timeslice query and spatio-temporal range query of moving objects. The dashed parts of objects trajectories are stored in the HTPR*-tree, and support predictive queries and partial history queries.

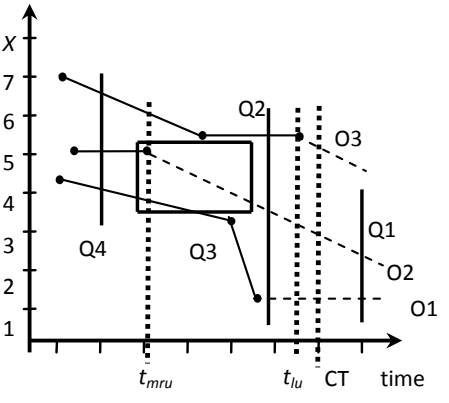


Fig. 5. Querying the Positions of Moving Objects

For example, query *Q1* is predictive timeslice query, and gets object *O1* and *O2*. Query *Q2*, *Q3,* and *Q4* are history queries. Query *Q2* intersects with partial history trajectories after the most recent update instant (t^o_{mru}) of objects *O1* and *O2* stored in HTPR*-tree. However, HTPR*-tree couldn't realize query *Q2* completely because the query time is earlier than the last update time (t_{lu}). In order to realize the queries involved from the past to the future, HTPR*-tree should be combined with some indices for describing history trajectories such as TB-tree.

Algorithm 6 is an illustration of spatio-temporal range query in the HTPR*-Tree. In algorithm *RQuery*(r, w, T_1, T_2), if *st1* of root is larger than T_2, query is unsuccessful. Else query calls *a*lgorithm *LeafNodeRQuery* (root is leaf node) or *nonLeafNodeRQuery* (root is non-leaf node). If T_1 is larger than *st2* of root and T_2 is small than or equal to the current time, algorithm *RQuery* can realize history range query completely, which is very similar to that find in predictive range query. Algorithm *IN*(*o*, w, T_1, T_2) determines whether object *o* is located in range w from time T_1 to T_2.

Algorithm 6. RQuery(r, w, T_1, T_2)

1. root=Root(r) /*achive the root of the HTPR*-tree
2. get *st1* and *st2* of root
3. if T2<*st1* return null
4. else if root is leaf node invoke **LeafNodeRQuery** (root, w, T_1, T_2)
5. else invoke **nonLeafNodeRQuery** (root, w, T_1, T_2)

ENDRQuery

Algorithm 7. Leaf NodeRQuery (*N*, w, T1, T2)
1. for each *o* of *N*
2. if **IN**(*o*, w, T1, T2) return *o*
ENDLeaf Node RQuery

Algorithm 8. nonLeafNodeRQuery (*N*, w, T1, T2)
1. for each son node *N'* of *N*
2. if *N'* is leafnode **Leaf NodeRQuery** (*N'*, w, T1, T2)
3. else **nonLeafNodeRQuery** (*N'*, w, T1, T2)
ENDnonLeaf Node RQuery

4 Bottom-Up Update

It is well known that the update efficiency of TPR*-tree is not very high since it is worked in a top-down manner. This is also the case for the top-down update HTPR*-tree. In order to support frequent update, bottom-up update strategy is adopted by the HTPR*-tree.

To support bottom-up update strategy, the HTPR*-Tree supplements auxiliary structures which include hash table, and compact main memory summary structures such as bit vector and direct access table. Figure 6 shows auxiliary structures for HTPR*-Tree.

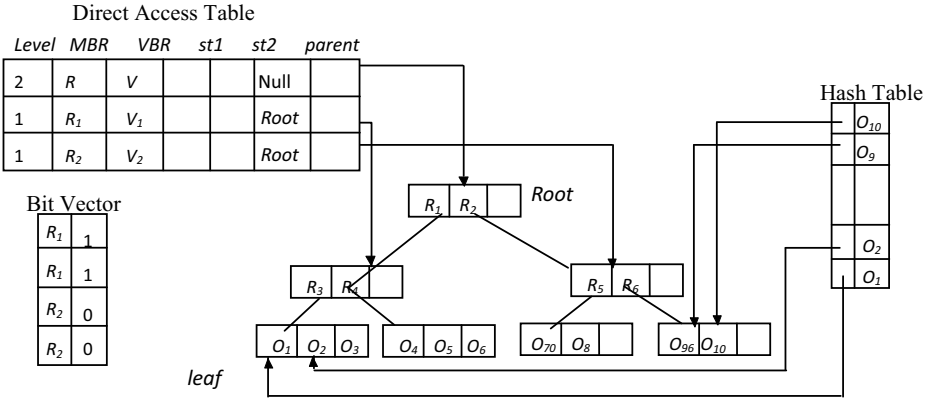


Fig. 6. Auxiliary structures for the HTPR*-Tree to support bottom-up update

Hash table allows us to locate the leaf node where the updated object *o* resides in one disk I/O instead of doing the expensive query on the tree.

Direct access table facilitates quick access to a node's parent in the HTPR*-Tree. An entry of the direct access table corresponds to a non-leaf node of the HTPR*-Tree,

and all the entries are organized according to the levels of the internal nodes they correspond to. An entry in the direct access table is a 7 tuple of the form < *Level, MBR, VBR, st1, st2, parentptr, ptr*>, where *Level* is the level of the node, *MBR* and *VBR* is the bounding box and the velocity bounding rectangle of the node at time *st,* respectively, *parentptr* is a pointer to the node' parent, *ptr* is a pointer to the node itself, *st1* is the minimal creation or update time of moving objects included in node, and *st2* is the maximum value compare to *st1*. Bit vector on the leaf nodes indicates whether they are full or not, which avoids additional disk accesses to find a suitable sibling.

The maintenance cost for the main-memory summary structure is relatively inexpensive. Since most of the node splits occur in the leaf level due to the high node fan-out, inserting a new entry into the direct access table will be very infrequent. Meanwhile, only when the leaf node is split or deleted, a new entry need be inserted into bit vector or be deleted.

The size of each entry in the direct access table is a small fraction of the size of the corresponding HTPR*-Tree node. And the size of bit vector is much smaller than that of direct access table, and even neglectable. Overall, the space consumption of the main-memory summary structure is very low relative to HTPR*-Tree.

The bottom-up strategy aims to offer a comprehensive solution to support frequent updates in the HTPR*-Tree. The main idea of bottom-up update algorithm is described as follows: when an object issues an update request, the algorithm adopts different update method according to object position and velocity after updating, and update time. The detailed update method is as follows:

1. If new position lies outside MBR of root or new velocity lies outside VBR of root, the algorithm issues a top-down update.

2. If new position and velocity of moving object lie in the MBR and VBR of current leaf node, the algorithm modifies the object entry in leaf node directly. At the same time, the algorithm constructs update path from leaf to root using the direct access table and tightens all nodes on that path.

3. If new position and velocity of moving object lie outside the MBR and VBR of current leaf node, and the removal of moving object causes leaf node to underflow, the algorithm issues a top-down update.

4. If new position and velocity of moving object lie in the MBR and VBR of non-null sibling node, and the removal of moving object couldn't cause leaf node to underflow, the algorithm deletes old entry and inserts new entry in right sibling node. At the same time, the algorithm constructs update path from leaf to root using the direct access table and tightens all nodes on that path.

5. If the new position and velocity of moving object lie in the MBR and VBR of a subtree (intermediate node), the algorithm ascends the HTPR*-tree branches to find a local subtree and performs a standard top-down update.

Algorithm 9 describes bottom-up strategy of the HTPR*-Tree.

Algorithm 9. Update (*o, o',r,DAT,BV,H)*

```
1.  entry=DAT[0]
2.  if o'.MBR⊄ entry.MBR or o'.VBR⊄ entry.VBR
3.       TD_Update(o, o',r,DAT,BV,H)
4.     return
5.  leaf=H.getLeafNode(o)
6.  if o'.MBR⊂ leaf.MBR and o'.VBR⊂ leaf.VBR
7.       write out leaf
8.       non-leaf=getParent(leaf)
9.       if non-leaf is not tigten
10.           tigten non-leaf
11.           non-leaf'=DAT.GetParent(non-leaf)
12.           while (non-leaf' is not tigten and non-leaf' is not null)
13.              tigten non-leaf' , non-leaf =non-leaf'
14.              non-leaf'=DAT.GetParent(non-leaf)
15.    return
16.  if leaf.numEntry=m      TD_Update(o, o',r,DAT,BV,H)
17.    return
18.  leaf.delete(o)
19.  construct update-path from leaf to root using the DAT, and tighten all nodes on
       that path similar to lines 8-14
20.  {siblings}=BV.getsiblings(leaf )
21.  for Si∈ {siblings}
22.    if o'.MBR⊂ Si.MBR and o'.VBR⊂ Si.VBR and BV.notFull(Si)
23.      if o'.st> Si.st2
24.           si.insert(o' )
25.           construct update-path from si to root using the DAT, and change st2 of
           all nodes on that   path into st similar to lines 8-14
26.      else   si.insert(o' )
27.   return
28.  non-leaf =getParent(leaf)
29.  while (o'.MBR⊄non-leaf.MBR or o'.VBR⊄ non-leaf.VBR)
30.       non-leaf =DAT. GetParent(non-leaf)
31.  TD_Update(o, o', non-leaf, DAT,BV,H)
32.  construct update-path from non-leaf to root using the DAT, and tighten all
     nodes on that path similar to lines 8-14
33.  return
END  Update
```

5 Performance Study

5.1 Experimental Setting and Details

In this section, we evaluate the query and update performance of the HTPR*-tree with the TPR*- and TD_HTPR*-tree (top-down update strategy). Due to the lack of real

datasets, we use synthetic data simulating moving aircrafts like[2]. First 5000 rectangles are sampled from a real spatial dataset (LA/LB)[15] and their centroids serve as the "airports". At timestamp 0, 100k aircrafts (point objects) are generated such that for each aircraft *o*, (i) its location is at one (random) airport, (ii) it (randomly) chooses a destination airport, and (iii) its velocity value *o.Vel* uniformly distributes in [20,50], and (iv) the velocity direction is decided by the source and destination airports. For each dataset, we construct a HTPR*- and TPR*-tree, whose horizons are fixed to 50, by first inserting 100k aircrafts at timestamp 0.

Since the HTPR*-tree only stores history trajectories after the most recent update of each object, and history trajectories index such as TB-tree only store trajectories before the most recent update instant, it is improper to compare the history query performance of HTPR*-tree with that of history trajectories index. In our experiments, we only study the predictive queries. However, HTPR*-tree can be combined with history trajectories index to support the queries involved from the past to the future. This will be our future work.

5.2 Performance Analysis

In order to study the deterioration of the indices with time, we measure the performance of the HTPR*-, TPR*- and TD_HTPR*-tree, using the same query workload, after every 10k updates.

• Update Cost Comparison

Figure 7 compares the average update cost (for datasets generated from LA and LB as mentioned above) as a function of the number of updates. The HTPR*- and TPR*-tree have nearly constant update cost. However the node accesses needed in the HTPR*-tree update operation are much less than the TPR*- and TD_HTPR*-tree. This is due to the fact that the HTPR*-tree adopts bottom-up update strategy to avoid the excessive node accesses for top-down deletion search and insertion search, and the TPR*- and TD_HTPR*-tree process update in top-down manner. Since node overlap in the TD_HTPR*-tree is larger than that in the TPR*-tree, the query cost increasing with the number of updates improves the update cost of TD_HTPR*-tree.

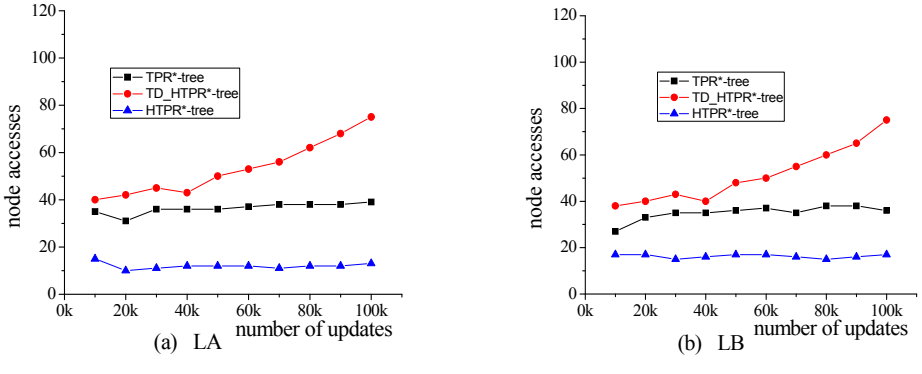


Fig. 7. Update cost comparison

- **Query Cost Comparison**

The query cost is measured again as the average number of node accesses in executing 200 predicted window queries with the same parameters $q_R len$, $q_V len$, $q_T len$.

In Figure 8, we plot the query cost as a function of the number of updates, using workloads with different parameter: for Figure 8 (a) we fix parameters $q_T len$=50, $q_R len$=400 and $q_V len$=10, while the parameters $q_T len$=50, $q_R len$=1000 and $q_V len$=10 are fixed in Figure 8 (b).

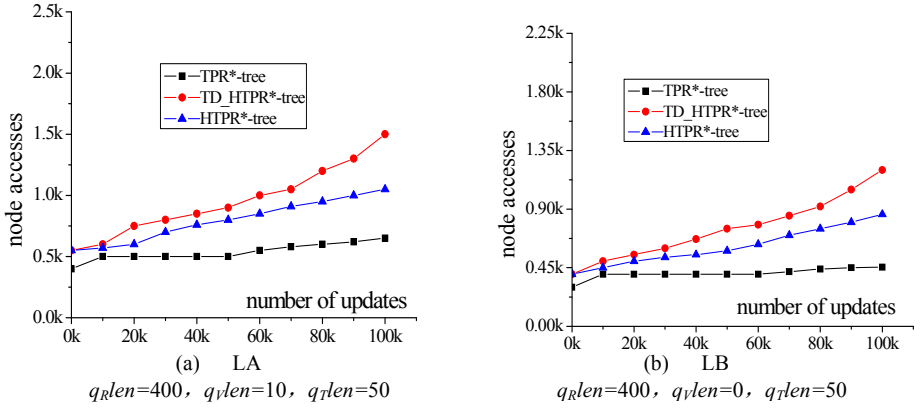


Fig. 8. Query cost comparison

It is clear that the query cost increases with the number of updates. The query cost of the HTPR*-tree is less than that of the TD_HTPR*-tree. Since the node overlap in the HTPR*-tree is larger than that in the TPR*-tree, the query cost of the HTPR*-tree is a bit higher than that of the TPR*-tree. Despite this, the important fact is that HTPR*-tree can support history query.

6 Conclusion

In this paper, we develop a novel index structure named the HTPR*-tree which not only supports predictive queries but also partial history ones. At the same time, we propose a bottom-up update approach to support frequent update operation of the HTPR*-tree. Extensive experiments prove that the update performance of the HTPR*-tree is better than that of the TD_HTPR*- and TPR*-tree. Moreover, the HTPR*-tree can support history query compared with TPR*-tree although the predictive query performance is a bit less.

For the future work, we will combine the HTPR*-tree with history trajectory indices such as TB-tree to implement historical and future information retrieval.

Acknowledgments. This work is supported by the National Natural Science Foundation of China (Grant No.90718027) and the Natural Science Foundation of Hubei Province (Grant No.2008CDA007).

References

[1] Saltenis, S., Jensen, C.S., Leutenegger, S.T., Lopez, M.A.: Indexing the Positions of Continuously Moving Objects. In: ACM SIGMOD, pp. 331–342 (2000)
[2] Tao, Y., Papadias, D., Sun, J.: The TPR*-Tree: An Optimized spatiotemporal Access Method for Predictive Queries. In: VLDB, pp. 790–801 (2003)
[3] Lee, M., Hsu, W., Jensen, C., et al.: Supporting Frequent Updates in R-Trees: A Bottom-Up Approach. In: VLDB, pp. 608–619 (2003)
[4] Pfoser, D., Jensen, C.S., Theodoridis, Y.: Novel Approaches to the Indexing of Moving Object Trajectories. In: VLDB, pp. 395–406 (2000)
[5] Theodoridis, Y., Vazirgiannis, M., Sellis, T.: Spatio-temporal Indexing for Large Multimedia Applications. In: Conf. on Multimedia Computing and Systems, pp. 441–448 (1996)
[6] Nascimento, M.A., Silva, J.R.O.: Towards Historical R-trees. In: Proc. of the ACM Symposium on Applied Computing, pp. 235–240 (1998)
[7] Tayeb, J., Ulusoy, O., Wolfson, O.: A Quadtree-Based Dynamic Attribute Indexing Method. The Computer Journal 41(3), 185–200 (1998)
[8] Jignesh, M., Yun, P., Chen, V., Chakka, P.: TRIPES: An Efficient Index for Predicted Trajectories. In: ACM SIGMOD, pp. 637–646 (2004)
[9] Liao, W., Tang, G.F., Jing, N., Zhong, Z.-N.: Hybrid Indexing of Moving Objects Based on Velocity Distribution. Chinese Journal of Computers 30(4), 661–671 (2007)
[10] Jensen, C.S., Lin, D., Ooi, B.C.: Query and Update Efficient B+-Tree Based Indexing of Moving Objects. In: VLDB, pp. 768–779 (2004)
[11] Chen, S., Ooi, B.C., Tan, K.L., et al.: ST^2B-tree: A Self-Tunable Spatio-Temporal B+-tree Index for Moving Objects. In: ACM SIGMOD, pp. 29–42 (2008)
[12] Chen, N., Shou, L.D., Chen, G., et al.: B^s-tree: A Self-tuning Index of Moving Objects. In: Kitagawa, H., Ishikawa, Y., Li, Q., Watanabe, C. (eds.) DASFAA 2010. LNCS, vol. 5982, pp. 1–16. Springer, Heidelberg (2010)
[13] Pelanis, M., Saltenis, S., Jensen, C.S.: Indexing the Past, Present and Anticipated Future Positions of Moving Objects. In: ACM TODS, pp. 255–298 (2006)
[14] Raptopoulou, K., Vassilakopoulos, M., Manolopoulos, Y.: Efficient processing of past-future spatiotemporal queries. In: Proc. of the ACM Symposium on Applied Computing, pp. 68–72 (2006)
[15] `http://www.census.gov/geo/www/tigers`

Developing Rich Web GIS Applications for Visual Analytics

Michael Meyers[1] and Bruce A. Ralston[2]

[1] County Technical Assistance Service, Knoxville, TN 37996-0213 USA
mmeyers@tennessee.edu
[2] Department of Geography, University of Tennessee, Knoxville, TN 37996-0925 USA
bralston@utk.edu

Abstract. The development of Rich Internet Application tools and web mapping services, coupled with the dissemination of detailed, up-to-date socioeconomic data, present an opportunity to develop intuitive geospatial visual analytic tools. This paper presents our efforts at developing such tools that combine Google Maps, Flex, and data from the American Community Survey.

Keywords: Rich Internet Applications, Visual Analytics, American Community Survey, Google, Flex, KML.

1 Introduction

The world of internet GIS continues to evolve at an almost dizzying pace. Advances continue to be made in areas of development environments, web mapping services, and data availability. Indeed, data sources now include real time data from sensor networks [1], volunteered data, e.g., [2], and increasingly timely government data, such as the American Community Survey [3]. These tools and data advances give us the opportunity to develop new approaches for providing data analysis capabilities to literally anyone with internet access. Whether this is referred to as Web GIS or Geospatial Visual Analytics [4] these approaches allow us to develop web applications that let users retrieve data, visualize them in graphic, cartographic, and tabular formats, and interact with those formats in linked and synchronized ways. We now have the ability to develop Web GIS applications, some of which may employ statistical analysis methods, that support Exploratory Data Analysis as championed by Tukey [5]. Such applications allow users to draw inferences based on their ability to construct multiple data visualizations and interact with those visualizations in an intuitive manner. Such applications allow "…the human to directly interact with the underpinning data to gain insight, draw conclusions, and ultimately make better decisions" [4]. In short, we seek to build web applications that facilitate the user interaction with the data without the software getting in the way [6].

L. Wang et al. (Eds.): WAIM 2011 Workshops, LNCS 7142, pp. 40–50, 2012.

This paper presents our efforts to develop such tools for exploring data made available by the United States Census Bureau. This is particularly timely because the Census Bureau has begun releasing detailed data from American Community Survey (ACS), a data series that will supply yearly updates of socioeconomic and demographic data for communities across the United States. This marks "…the beginning of annual estimates for small communities and neighborhoods throughout the country" [7]. Prior to the development of the ACS, detailed socioeconomic data for the United States was only collected once per decade. Not only will the information released be more current, after a few years there will be time series data for a host of social and economic information for those communities. While these data are now coming online, the tools for exploring are waiting to be developed by the academic community, state and local government organizations, and the private sector. Such tools will give various user communities the ability to explore this more up-to-date information for making policy decisions.

2 Development Considerations

There are several decisions to be made when developing geospatial visual analytic tools. For example, what types of data representation should be developed? What underlying mapping service should be used? What types of data interaction should be supported? How much control should the user have in determining how the data are retrieved and presented? How much processing should be done on client and how much on the server?

In addition to these basic design questions, decisions on which development tools to use must be made. There is no one best set of tools, and choices often reflect the developer's experience and preferences. Current rich internet application (RIA) development is dominated by frameworks available in Flex, Silverlight, and to a lesser extent, JavaFX. Despite its capability, RIA drives few complex web applications compared to those developed with HTML, CSS, Javascript/AJAX, and, often, Flash or Silverlight plugins for video or simple animation. HTML5 extends standard HTML to provide a compliance standard for web graphics and video. Because it promises to be accessible in all major browsers and mobile hardware platforms, it is an important challenge to solutions using plugins and that consume higher client resources. The HTML5 video element is in full scale trial across the Internet, including as an opt-in trial on Google's YouTube. The less prevalent HTML5 canvas element provides rendering of 2d shapes and bitmaps (http://www.w3.org/TR/html5/the-canvas-element.html#the-canvas-element). MIT has developed an open source dynamic vector mapping framework [8] that includes an extension of CSS called geographic style sheeting (GSS). Performance is slow on the current implementation of the canvas element, and HTML5, even in its final specification, will not provide the rich set of user interface tools in the form of data grids, charting, and navigation that currently are available in Flex and Silverlight. Therefore the adoption of HTML5 for high performance geospatial visualization is probably some years in the future.

For our projects we are using Google's mapping service API for Adobe's Flex 4 development environment to develop the web tools presented to the user. Flex 4 is a RIA development environment that utilizes a sandbox in which to run powerful applications on client machines. By using an RIA, the tools developed can have many of the characteristics of desktop applications. Another advantage of RIA is that processing formerly done on the server can now be done on the client. An earlier effort at thematic map web mashups depended on server side construction of on-the-fly thematic overlays [9]. This required a round trip between the client and the server every time a new map option was selected. By using an RIA the need for round-tripping for each new client request, such as choosing a new mapping variable, is eliminated. This lessens server load and speeds response time. It does, however, use more of the client's resources so there is a tradeoff.

There are also design decisions concerning server side operations and data preparation. Microsoft SQL Server is the database used for processing user queries, and in-house tools convert and format spatial data for Web GIS. For report generation, an open source PDF library is used.

Developing data exploration tools for public use requires a sense of one's audience. Expert GIS users are rare, but even the average Internet user has experience using web mapping tools for driving directions, real estate search, and location based services using mashups based on Google Maps and similar products. In addition, RIA leverages the familiar vocabulary of dropdown menus, scrollbars, and other desktop interactions. The average user thus arrives at a website featuring Web GIS and associated visualization tools with the expectation that its functionality is intuitively apparent, or by a process of discovery lasting no more than a few mouse clicks.

2.1 Visualization Components

The Web GIS projects we have developed contain at least three data visualization components that combine and extend the concepts of enterprise dashboard mashups and data portal. In addition, there are controls that allow the user to select which data sets to explore. The data visualization components consist of a map view, a table view, and a chart view [Fig 1]. These visualization portals and their interactions are described below.

The map view consists of a Google Map service with custom data overlays that are generated on-the-fly in response to user choices. Within the map view are buttons for displaying a legend box and a mapping options box. The mapping options box allows the user to choose the method of classification, set the number of classes, specify the color ramp to use, and adjust the overlay transparency [Fig. 2]. The user can also choose to change the variable being mapped. The overlays are generated by creating polygon options for area features stored in "vanilla" KML format. The KMLs are passed from the server to the client as XML lists. By using an RIA, the functions required for thematic mapping changes are performed locally.

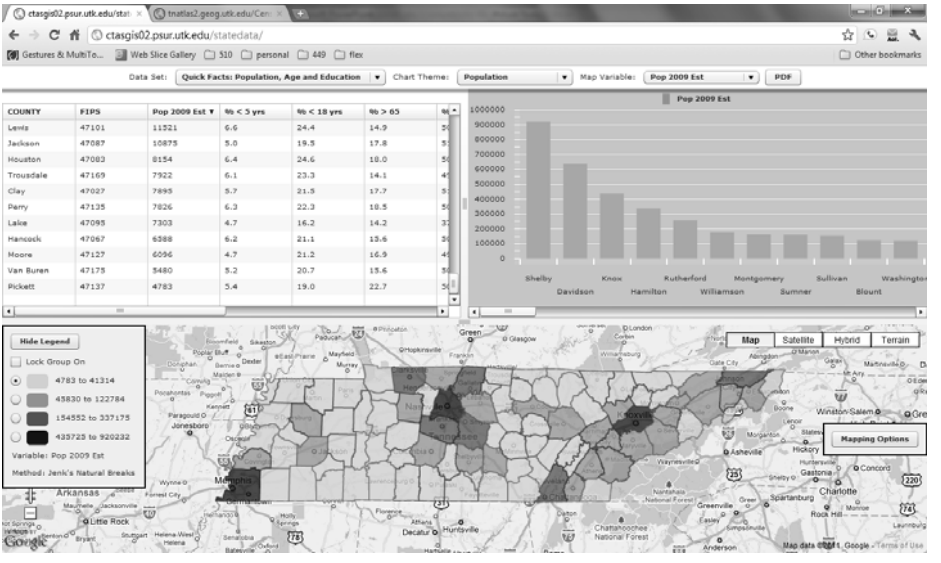


Fig. 1. A web page with the table (*top left*), chart (*top right*), and thematic map (*bottom*) visualization components

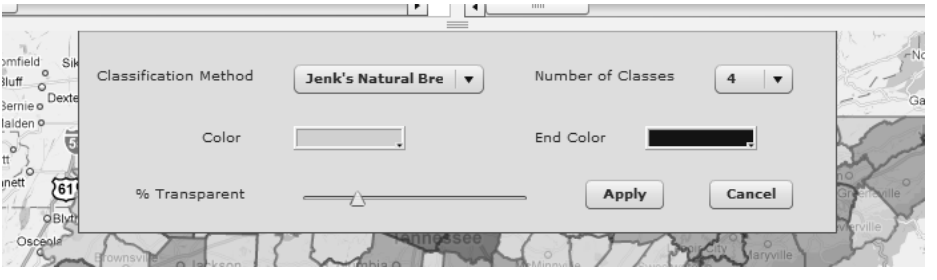


Fig. 2. The mapping options box (center) allows users to set the classification method, number of groups, color ramp and transparency

The table view of the data is accomplished using a standard data grid control. Standard table functions such as selecting and sorting are built into Flex data grids. The biggest challenge in using data grids in our applications is dynamic management of grids. Because we allow the user to change the data queries, the types and numbers of columns in a grid are not known until run time. This requires clearing the columns of any existing grid from memory and creating a new grid every time a new data query is made.

The final data portal in our applications is a charting view. In this type of view data can be presented in various forms, such as bar charts, pie charts, and plot charts. Since the data are loaded locally into the RIA sandbox, incorporating statistical functions into the application is possible. Fig.3 shows a plot chart that includes a regression line and its standard statistical measures (i.e., slope, intercept, and r^2).

For this example, the graph shows the relationship between education and income in Knox County, Tennessee. As in Figure 1, this chart will be accompanied by the corresponding map and data grid views.

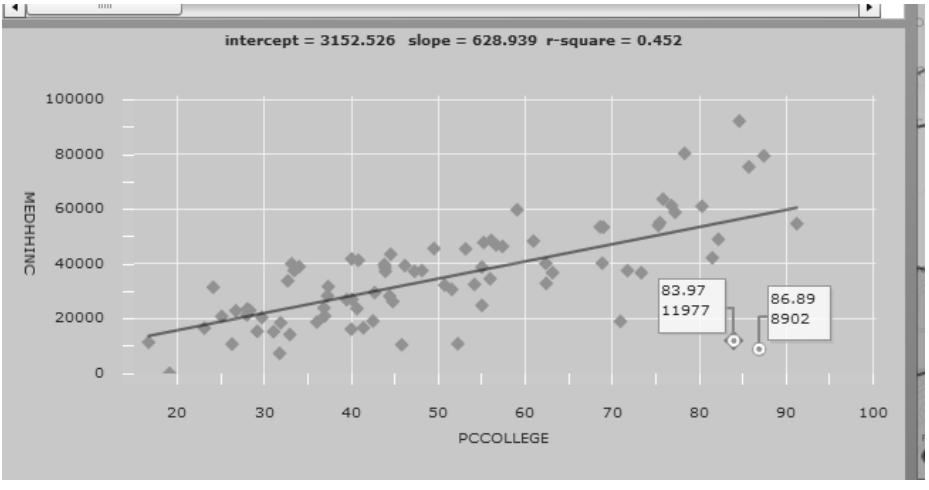


Fig. 3. An application that contains a regression analysis chart that shows the relationship between education and income in Knox County, Tennessee

2.2 Component Behavior

As well as providing multiple views of the same data, these components must respond to user interactions. The responses can be classified into two types: those that affect only the current component and those that require responses from all components simultaneously. The first type responses are the usual point-and-click responses. For example, pointing at an element in a chart will display data of the associated spatial unit. Similarly, clicking on a map feature will bring up the data corresponding to that feature. The second type of response requires that dynamic links be maintained between all three data views. For example, if a user rolls over an element in a one representation of the data, then the corresponding elements in the other representations should be highlighted. Care must be taken as the relationship between entities in the components is not always one-to-one. In a plot chart, for example, one point may correspond to several spatial features, or a mouse roll over may select more than one point, as in Figure 3.

This is useful for exploring relationships in a spatial and aspatial manner simultaneously. For example, there are two outliers in the lower right of the plot chart in Fig. 3. These are two areas that have high education levels but much lower than expected income levels. When these points are rolled over, the corresponding areas on the map are highlighted. These are neighborhoods with a high concentration of graduate student housing.

Another useful form of dynamic linking is to use the results in one data view to alter what is shown in the other views, such as filtering or re-ordering elements. Figure 4, for example, shows how rolling over a group entry in the map legend can filter all three windows to only show cases within that group. Similarly, sorting elements in the data grid will change the order in which values are presented in a chart. In Figure 1, for example, the elements in the data grid are sorted from highest to lowest on the same variable on which the chart is based. Thus, the chart displays values from highest to lowest. Other forms of filtering can include those based on spatial contiguity and areas within the same administrative districts or subject to the same policy rules.

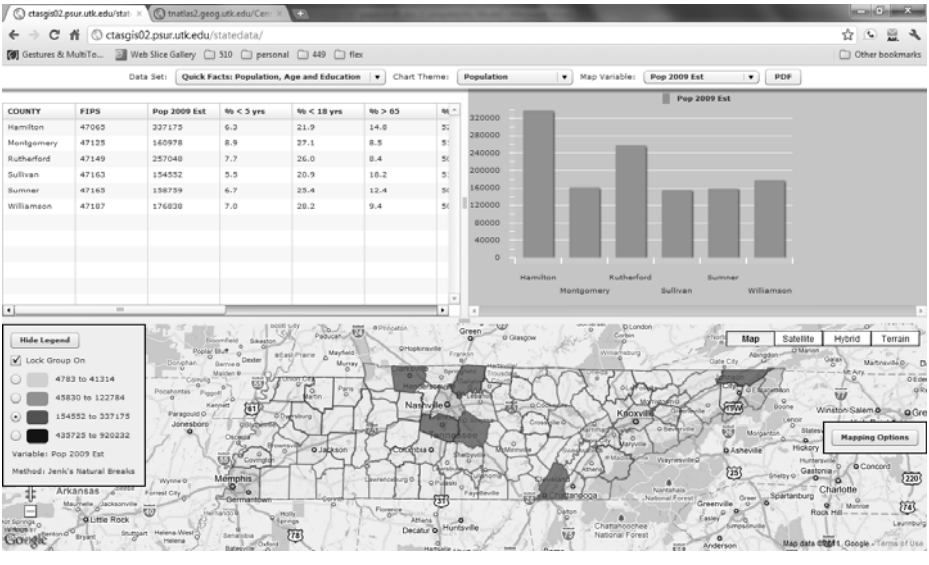


Fig. 4. Filtering data in all three views by choosing to view only the third group in the thematic map

While the default mode of operation is for all components to display one or more fields of the same topic, each view can be decoupled to show other data elements for additional analytic capability. Civilian unemployment percentages could be charted, for example, while housing values are mapped. However, the linked display of the spatial unit selected in any component continues to operate in the same way.

2.3 Component Layout

One task in designing web applications that present multiple views of information is management of the screen real estate. This is particularly challenging when the number of variables to be downloaded, the shape of areas to be rendered, and the

number of features to charted is not known until run time. We have considered a few approaches to dealing with these issues.

The most common approach is to use scrollbars. In this approach each data view container acts as a window that displays all or part of the information. A second approach is to allow the user to resize the data portal windows, often with scrollbars. This is what is shown in Figure 1. A third approach is to limit the number of features that are added to a component, e.g., only charting the top 5 and bottom 5 cases for a variable. It is also possible to swap the contents of the data portals to allow the user to give the most screen real estate to the data portal he chooses. By using transitions the movement of contents between windows can appear as smooth animation. A fourth approach is to support zooming and panning of the various data views. Zooming and panning of maps are quite common, but it is also possible to zoom and pan certain types of charts by extending Flex classes [10]. One could also change the layout dynamically based on the map extent of the area being studied or let the user choose from a set of layout templates at run time.

2.4 Reporting

If a Web GIS application is useful, then users will eventually want to generate reports. Our approach is to generate reports as PDF files. This is accomplished using AlivePDF, a free and open source library for creating PDFs from Flex websites [11]. Flex has an "addImage" method that will simply add the screen shot of a data portal to the report. However, this is unacceptable for the data grid and map. In the case of the grid, columns not on the screen are not reported. Further, the table is an image, not text. In the case of the map, Google Maps will not allow the image to be captured with this method. That is, using the addImage function on a portion of the screen that contains a Google map will result in a security violation.

To overcome the first of these problems we use the following strategy. Since the number of variables and features is not known until run time, the size of table to be reported must be calculated on the fly. By knowing the number of columns in the table it is possible to generate page breaks where necessary and to repeat identification fields on each page. An AlivePDF grid object is then added to each page. This prevents clipping of the table or compacting of fields so that they are unreadable. It also ensures that the identification field appears on each page. The code for accomplishing this is available at [12].

Trying to capture the map with the standard Flex AddImage function will result in a sandbox security error. The Google API function "getPrintableBitmap" must be used. However, this captures the map and its overlays. It does not capture any Flex generated controls in the map view portal, such as the legend. This results in the legend being printed separately from the map—not an ideal situation.

3 Data Preparation

The Census Bureau does not release information in ready-to-use GIS formats. In particular, the information on the geography of enumeration areas (e.g, counties, tracts, and metropolitan areas) is released in TIGER files. Developed by the Geography Division of the Census, these files do not contain socioeconomic information that one normally associates with the census. That information is released by other divisions through products like the 2010 Census of Population and Housing and the ACS. In order to support Web GIS of census related data, information from both these sources must be preprocessed and stored on the server.

3.1 Attribute Data Preparation

ACS data present a two significant challenges to the development of a visualization system. First, the number of variables is very large and values are updated annually, so we are building batch processing utilities that automate the tedious process of download and update. User demand and legislative issues are expected to drive which variables are selected for exploration. Second, the ACS data frequently must be recast to create meaningful demographic relationships that can be visually displayed. The Census Bureau tends to report counts of people and households. However, percentages or proportions of the population are often more useful. For example, the number of people below the poverty level in areas tends to reflect the total population size of an area. To get a measure of the concentration of poverty, the percentage of persons below the poverty level is a more useful measure. We have created tables based on the 2009 ACS five-year sample that contain socioeconomic measures that have been normalized by the appropriate universe variables. For example, the percentage of households that are owner occupied is derived from the number who are owner occupied scaled by the total number of households.

It is possible to simply store the count information in a database and create calculated fields on demand. Indeed, users could create variables through an on screen calculator dialog. The resulting variable could then be returned from the server as a calculated field or calculated on the client. This may be a reasonable strategy for sophisticated users. We believe that the standard public user prefers not to be burdened with specifying the details of variable creation. Creating commonly used variables ahead of time and storing them on the server removes this burden from the user and speeds the response to client side queries.

3.2 Geographic Data Preparation

Separating attribute information from geographic information has a long history, dating back to the GBF/DIME files of the 1970s. In 1988 the first of several versions of TIGER files were made available by the Census Bureau. With each release of

TIGER, the format of the files changed and one had to use translators. The second author developed such a translator application for converting TIGER files to KML format,the source code of which is available at [13].

With TIGER 2007, which was released in 2008, the Census Bureau started releasing TIGER in a widely used, open format: ESRI Shapefiles. In order to use these files with attribute data available from the American FactFinder, the Census Bureau's searchable database of population and economic data, in Google Earth and Google Maps, another application was developed. The current RIA projects described in this paper are outgrowths of these previous efforts. That is, in order to use Google Maps as the underlying mapping service, we have developed code to transform the TIGER shape files to Google KML files. These KMLs provide geometry for dynamic thematic mapping of data at every level of geography for which the Census tabulates data.

4 Server Side Functions

Flex cannot connect directly to SQL Server like ASP.NET or other server side languages. We developed a web service that provides the Flex client a secure connection to the ACS tables and other US Census data stored on our host. In addition, the web service constructs SQL queries dynamically based on user events in the Flex client. Query functions on the web service are necessarily general to accommodate all user events that filter the location, time period, or topic of interest by map, chart or table selections or menu tool interactions.

Data dictionaries including data types and field name aliases are maintained on the SQL Server. The web service returns data tables or datasets in a Web Services Description Language (WDSL) document, which includes data type information for each field in the dataset embedded in an XML Schema element [14]. While the WDSL document is heavier than a simple XML document, it is simpler to develop client side routines for representation when the data type is known.

A KML document providing the polygon geometry of the selected geographic area is also returned from server archives. The returned WDSL document is parsed in different ways to create interactive chart elements, map thematic overlays and table structure that may differ in design based on the topic of analysis or the number of records returned. Thus, data filtering tasks are split between the server and client, with the server providing data typing and the minimum-sized document required to supply client rendering processes for all graphic components related to a chosen topic in a single request.

The primary purpose of this architecture, which is illustrated in Figure 5, is to reduce server requests and provide fast client-side performance while the user manipulates thematic mapping breaks and color ramps, sorts tables and chart arrays, and resizes component windows. The speed of this interaction and its visual richness is essential to the intuitive quality of GIS and spatial exploration.

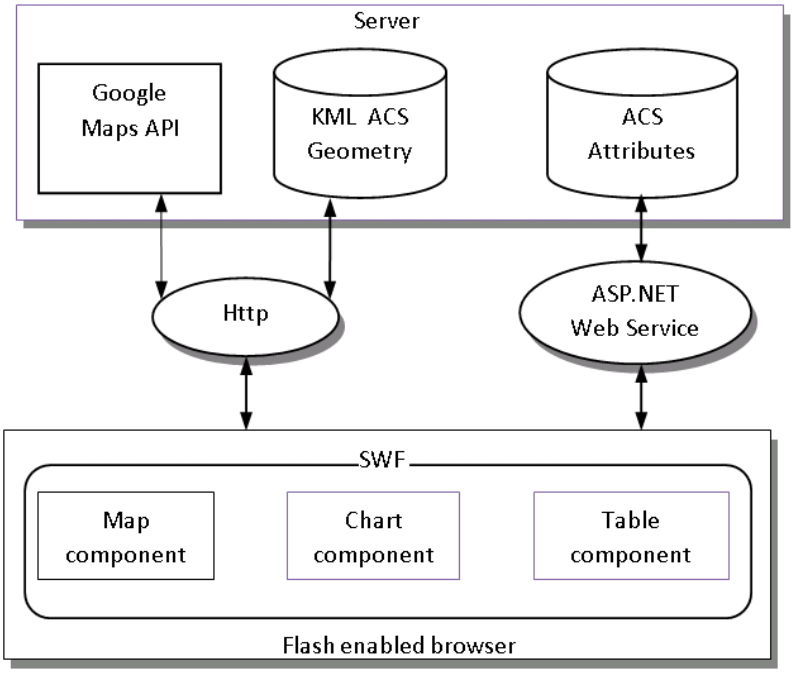


Fig. 5. After KML geometry and ACS data are retrieved from the server, they are processed for visualization components on the client machine in the SWF

5 Final Comments and Looking Ahead

There are many challenges remaining in developing rich internet GIS applications based on US Census Bureau products. These lie in at least three areas: working with time series of sample based data, the trade-off between simplicity and power, and the role of RIA on different hardware platforms.

The ACS is different from past Census products, not only in the timeliness of the data, but also in the sampling methodology. In particular, the ACS is based on smaller samples (1 in 40 households for the ACS compared to 1 in 6 for the previous "long form" results) that overlap in time. The sample size difference means that confidence limits will be wider under the ACS and great care must be taken when performing statistical tests. Because of the overlapping time windows (the recently released 2009 five year estimates are based on samples over the years 2005 to 2009) standard statistical tests between releases of the ACS whose sampling years overlap are not valid [15]. What is important is to get a sense of variability over time and to view trends in the time series data. There are many interesting, intuitive ways of viewing time series information, many of which employ animation techniques.

The trade-off between power and usability is a constant source of concern. Professional researchers often want powerful tools over which they have a great deal of decision making options. Decision makers and the general public may find such

tools and options overwhelming. Indeed, some find even simple statistics too technical for their liking. As stated previously the goal of visual analytics is to be intuitive and not have the software get in the way. Nonetheless, the increasing complexity of the information being processed (such as time series data, statistical variability, and the need for model building) often demands more complex tools. It is likely that we will develop a suite of web sites, each geared to a different level of user sophistication.

The growing use of RIAs coupled with REST services, such as Google Maps, Bing Maps, and ArcGISServer, has greatly improved the ease of application development and the response to user inputs. This change has taken place in only the past few years. Yet, the future of tools such as Flex and Silverlight is not clear. The rise of mobile computing platforms, such as smart phones and lightweight tablets, argues against pushing data and processing from the server to the client. It is likely that for these platforms the best place for manipulating data and performing analysis is on a cloud based server. Nonetheless, we believe RIAs running on the client have an important role to play, at least for the next few years.

Acknowledgments. The authors acknowledge the work of Jiuyuan Liu in data preparation. The support of the Tennessee Institute for Public Service is also acknowledged.

References

1. Culler, D., Estrin, D., Srivastava, M.: Overview of Sensor Networks. IEEE Computer 37(8), 41–49 (2004)
2. Open Street Map, http://www.openstreetmap.org/
3. United States Census Bureau, http://www.census.gov/acs/
4. Boulos, K., et al.: Web GIS in practice IX: a demonstration of geospatial visual analytics using Microsoft Live Labs Pivot technology and WHO mortality data. International Journal of Health Geographics 10, 19 (2011)
5. Tukey, J.W.: Exploratory Data Analysis. Addison-Wesley, Reading (1977)
6. Livnat Y., Samore M.: Visual Analytics in Surveillance and Epidemiology:Challenges and Opportunities (2010), http://www.cdc.gov/osels/ph_informatics_technology
7. Groves, R. M.: A Coming of Age for Local Social and Economic Statistical Information (2011), http://www.commerce.gov/blog/2011/04/21/coming-age-event-local-social-and-economic-statistical-information
8. Cartagen, http://cartagen.org/
9. Ralston, B., Streufert, J.: Efficient Generation of Area Thematic Maps in KML. In: 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems
10. Scrolling Flex Charts, http://www.connectedpixel.com/blog/scrollingcharts/
11. AlivePDF, http://alivepdf.bytearray.org/
12. Ralston, B.: Flex-Google Census Mapping Tutorial, http://tnatlas2.geog.utk.edu/tutorhome/
13. Ralston, B: TGR2KML, http://tnatlas.geog.utk.edu/tea/downloadfree.htm/
14. United States Census Bureau: Brave New World-Using the American Community Survey, http://www.lib.ncsu.edu/data/ACS.ppt/
15. Web Services Description Language, http://www.w3.org/TR/wsdl/

Single-Source Multi-Target A* Algorithm for POI Queries on Road Network

Htoo Htoo[1], Yutaka Ohsawa[1], and Noboru Sonehara[2]

[1] Graduate School of Science and Engineering, Saitama University
[2] National Institute of Informatics

Abstract. Searching for the shortest paths from a starting point to several target points on a road network is an essential operation for several kinds of queries in location based services. This search can be easily done using Dijkstra's algorithm. Although an A* algorithm is faster for finding the shortest path between two points, it is not so quick when several target points are given, because it must iterate pairwise searches. As the number of target points increases, the number of duplicated calculations for road network nodes also increases. This duplication degrades efficiency. A single-source multi-target A* (SSMTA*) algorithm is proposed to cope with this problem. It requires only one calculation per node and considerably outperforms Dijkstra's algorithm, especially when the target points are distributed with bias. An application with this algorithm for aggregate nearest neighbor search demonstrated its efficiency, especially when the number of target points is large.

1 Introduction

Route searching on a road network is an essential operation in location based services. A search for the shortest path between two points on the network is also necessary for various spatial query applications. Dijkstra's algorithm [1] and the A* algorithm [2] are representative algorithms used for such searches. Another frequent search is to find the shortest paths between starting point q and multiple target points $p(\in P)$. Such searches are needed, for example, in ANN (aggregate nearest neighbor) queries [3], skyline queries [4], k-NN queries [5], and several kinds of trip planning queries [6] [7].

While Dijkstra's algorithm can be directly applied to these queries, its performance is considerably degraded when the target points are distributed with bias or when some are distant from the starting point (see Figure 1(a)). A* algorithms can also solve these queries by iterating each pair-wise query $|P|$ times. Its performance is also degraded when $|P|$ is large, because neighboring road network paths to q are repeatedly processed (see Figure 1(b)). The inefficiency of this method increases with the size of $|P|$.

In this paper, we describe an algorithm that can efficiently find the individual distances from a single source to several target points. This single-source multi-target A* (SSMTA*) can find the shortest paths in ascending order of the distance from q, and it remains efficient even for very distant target points.

L. Wang et al. (Eds.): WAIM 2011 Workshops, LNCS 7142, pp. 51–62, 2012.

The basic operation in the shortest path search is to find neighboring nodes by using an adjacency list and to calculate the cost in terms of distance or time at each node. This operation is called *node expansion*. We explain *node expansion* detail in Section 3.2. SSMTA* expands a road network node at most once, the same as with Dijkstra's algorithm and in contrast to the conventional A* algorithm. Figure 1 shows the expanded node area for (a) Dijkstra's algorithm, (b) the A* algorithm, and (c) our SSMTA* algorithm.

The contributions of this paper are the followings.

- This is the first attempt of the A* algorithm to single-source multi-target situation.
- The SSMTA* algorithm was applied to an ANN query as an example. It can be adapted well to a wide range of location based service queries.
- Its efficiency was demonstrated experimentally.

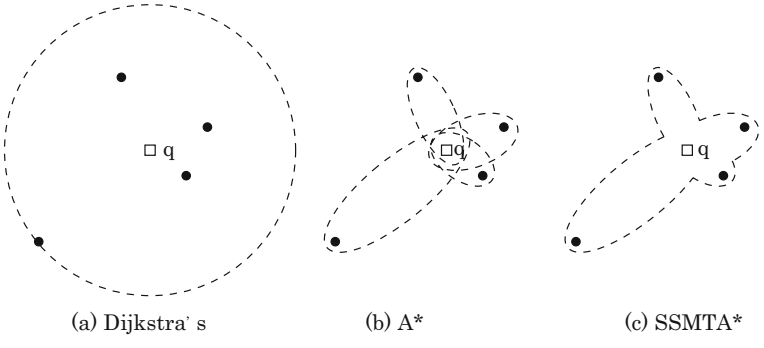


Fig. 1. Comparison among shortest path algorithms

The rest of the paper is organized as follows. Section 2 describes related work, and Section 3 describes the SSMTA* algorithm. Section 4 explains its application to ANN queries. Section 5 describes its experimental evaluation. Finally, Section 6 summarizes the key points and mentions future work.

2 Related Work

Compared with calculating the Euclidean distance, calculating the road network distance is computationally heavier. The calculation time for the shortest path between two points increases with the path length. As a result, there has been much research on finding shortest path query methods more efficient and reducing the computation times for road networks. The methods developed have generally taken one of two approaches: hypothesis verification or pre-processing. This section summarizes these methods. Related work on ANN queries is described in Section 4.

Papadias et al. [5] proposed two approaches to apply to several kinds of queries, including k-NN queries on a road network distance. They are incremental node expansion (INE) and incremental Euclidean restriction (IER). In the former, Dijkstra's method is simply used to search k-nearest neighbor points of interest (POIs). In the latter, the hypothesis verification is used. IER method finds k-NN points on the basis of the Euclidean distance before verifying the distance on the road network by using Dijkstra's algorithm. As described in Section 4, Yiu et al. adapted this approach to ANN queries [3].

In addition to pre-processing on a road network, several other types of methods have been proposed. Hu et al. [8] proposed a *distance signature* method which is a kind of materialization method on the road network distance. For a k-NN query, node information is added to an adjacency list which indicates the neighboring node leading to each POI. The shortest path from any node to a target POI on the road network can be found by simply tracing the path from one node to the next until the target POI is reached. Though, this method requires a data amount of $O(nm)$, where the n is number of nodes on the road network and the m is the number of POIs, it works very efficiently for finding the shortest path from a node to a POI. However, if points are added to or removed from the set of POIs, the re-construction to overall elements in the adjacency list is necessary.

Samet et al. [9] generalized this method into one for finding k-NN points in a best-first manner. This method is based on precomputation of the shortest paths between all possible nodes on the road network. The information for the next visited node is compressed using *shortest path quadtree*, resulting in $O(n^{1.5})$ storage cost.

Kolahdouzan et al. [10] proposed Voronoi based network nearest neighbor (VN^3) algorithm that searches for k-NN points using a network Voronoi diagram. While this method works well for bi-directional road networks, it can provide only an approximate shortest route if uni-directional roads are included. Zhu et al. [11] used the network Voronoi diagram for ANN queries.

Another precomputation method was proposed by Shaw et al. [12]. It uses an M-tree [13], a general-purpose data structure for metric space, like an R-tree for Euclidean space. This structure needs the calculated distance between two points during query processing, and this calculation can become very heavy for a road network because the shortest path calculation cost increases with the distance between two points. To overcome this problem, Zhu et al. proposed an approximation query method. Ioup et al. [14] used an M-tree for ANN queries in metric space.

3 Single-Source Multi-Target A* Algorithm

3.1 Preliminaries

In location based services, several types of POIs can be query targets. They can include nearby gas stations, hotels, and restaurants. Such facilities are POIs. Although a POI may not reside on a node in the road network, in the following

we assume that every POI is on a road network node. To resolve this restriction is straightforward [3].

Let set P contain k POIs, and let a starting point q be given. How to construct P from a large number of POIs is application dependent. We first consider the process for finding all paths from q to each $p(\in P)$ and calculating their distance on the road network. This can be solved using by Dijkstra's algorithm starting from q, or by using the A* algorithm in a repetitive manner between q and each p. In this section, we describe our more efficient method for doing this, the single-source multi-target A* (SSMTA*) algorithm.

First of all, we summarize the A* algorithm. The starting point(q) and target point (t) on the road network must be given in advance. The algorithm starts the search from q and the cost (c) in terms of distance or time is calculated at each node n encountered during the search.

$$c = d(q, n) + h(n, t)$$

The cost to move from q to n is represented by $d(q, n)$. The heuristics cost to move from n to t is represented by $h(n, t)$. The heuristics cost should not exceed the actual cost: i.e. $h(n, t)$ should be less than or equal to $d(n, t)$. When the distance on the road network is considered, the Euclidean distance between n and t satisfies this condition, and when the travel time is considered, the travel time with the fastest speed satisfies this condition.

Hereafter, the Euclidean distance is used as $h(n, t)$ and is denoted by $d_E(n, t)$, and $d_N(q, n)$ denotes the distance traveled between q and n on the road network..

3.2 Basic Algorithm

Here, we describe our SSMTA* algorithm searching for the shortest paths from a specified starting point to several target points on the road network. Table 1 summarizes the symbols used.

Table 1. Symbols

Symbol	Meaning
P	Target point set selected from POIs
q	Starting point, called query point in ANN query
Q	Query point set in ANN query
PQ	Priority queue
CS	Closed set
$d_E(x, y)$	Euclidean distance between x and y
$d_E^{min}(x, P)$	Minimum Euclidean distance between x and a point in P
$d_N(x, y)$	Road network distance between x and y
$RLink(x, y)$	Pointer to road segment with edges x and y

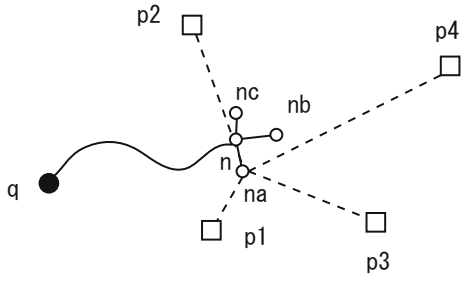


Fig. 2. SSMTA* Algorithm

In Figure 2, q is the starting point, and p_1 to p_4 are the target POIs in set P. The purpose of the query is to determine the shortest path from q to each point in P. Suppose that the path on the road network from q to n has already been determined, then n is currently noticed as a current node. n has three directly neighboring nodes (n_a to n_c). We first explain using n_a. The path length from q to n_a via n is $d_N(q,n) + d_N(n,n_a)$. Let $d_E^{min}(n_a, P)$ denote the distance between n_a and the point $p(\in P)$ that gives the minimum Euclidean distance. In the figure, p_1 is the closest POI to n_a, so $d_E^{min}(n_a, P) = d_E(n_a, p_1)$. Then, $Cost = d_N(q,n) + d_N(n,n_a) + d_E^{min}(n_a, P)$ is used to determine for the next expanded node. For each node neighboring n, this cost is calculated and then inserted into the priority queue (PQ), which controls the search order. We call this basic operation *node expansion*.

The records in the priority queue have the following format.

$$< Cost, N_C, N_P, d_N(q, N_C), RLink(N_P, N_C) > \tag{1}$$

The N_C represents the node of current interest (the *current node*). The N_P is the previous node of N_C on the path from q to N_C. The $d_N(q, N_C)$ is the road network distance from q to N_C, and $RLink(N_P, N_C)$ is the (pointer for the) road segment connecting N_P to N_C. For example, in Figure 2, N_C corresponds to n_a, N_P to n, and $d_N(q, N_C)$ to $d_N(q, n_a)$.

At the beginning of the SSMTA* algorithm, the record $< d_E^{min}(q, P), q, -, 0, - >$ is inserted into the PQ. The '$-$' of the third item shows the *null* value, because q does not have a previous node. In the same way, q does not have a previous link, so the *null* value is also assigned to the fifth item.

The steps in the SSMTA* algorithm are similar to those in the conventional A* algorithm. (1) Get the record having the minimum cost from the PQ; (2) check the current node (N_C) of the record to see whether it has been expanded (if the record has already been expanded, ignore it); (3) make a new record for each neighboring node and insert it into the PQ. Repeat (1) to (3) until all POIs have been visited. For the check in step (2), a once expanded node is inserted into the *closed set* (CS). The CS keeps all records that were obtained in step (1) indexed by the current node N_C. If the N_C of the record extracted in step (1) is already in the CS, the node has already been expanded, so the node is ignored.

Let the record obtained from the PQ be e. If $e.N_C$ matches $p(\in P)$, one POI has been found. The corresponding node is then removed from P. The number of points in P is reduced each time a POI is found. The CS is used to retrieve the route from the found POIs (target points) to q (the starting point) in a reverse manner by tracing the N_P, the previous node. It is the same way as with Dijkstra's algorithm or the A* algorithm. First, $e.N_C$ is searched in the CS, then the N_P of this record is searched in the CS. This searching is repeated until N_P meets q. When N_P meets q, the route can be obtained. Algorithm 1 shows the pseudo code for the SSMTA* algorithm.

Algorithm 1. SSMTA*

```
1.  R ← ∅
2.  d_min ← min(d_E(q, p_i), p_i ∈ P)
3.  enqueue(< d_min, q, −, 0, − >)
4.  loop
5.    e ← deleteMin()
6.    if CS.Contain(e.N_c) then
7.      CS.renew(e.d_N(q, e.N_P)); Continue;
8.    else
9.      CS.add(< e.N_C, e.N_P, e.d_N, e.RLink >)
10.   end if
11.   if e.N_C ∈ P then
12.     R ← R∪ < e.N_C, getPath(e.N_C) >
13.     P ← P − e.N_C
14.     if P = ∅ then
15.       return R
16.     end if
17.   end if
18.   for all nn ∈ neighbor(e.N_C) do
19.     decide p(p ∈ P) which gives d_E^min(nn, P)
20.     d_N ← d_N(q, e.N_C) + d_N(e.N_C, nn)
21.     enqueue(< d_N + d_E^min(N_C, P), nn, e.N_C, RLink(N_c, nn) >)
22.   end for
23. end loop
```

In Dijkstra's algorithm and the conventional A* algorithm, after a record has been obtained once from the PQ and inserted into the CS, the distance from q to current node N_C in the record is fixed. In contrast, in the SSMTA* algorithm, the value $(d_N(q, N_C))$ can be changed as illustrated in Figure 3. The numbers in parentheses show the order in which the records were obtained from the PQ.

Now, suppose node m has been obtained from the PQ (1), and that $Cost_a = d_N(q, a) + d_E(a, p_2)$ for node a, $Cost_b = d_N(q, b) + d_E(b, p_1)$ for node b, and $Cost_n = d_N(q, n) + d_E(n, p_1)$ for n have been calculated. This means that the nearest $p(\in P)$ for b and n is p_1, and for a is p_2. These three records are created and inserted into the PQ. If $Cost_a > Cost_n > Cost_b$, the record for b is obtained from the PQ first (2). b is expanded and somehow the search path reaches p_1, which is then removed from P. In this situation, P remains only p_2 in the example.

Next, n is obtained from the PQ (3) and expanded. Through this node expansion, records for a and o are created and n is inserted into the CS. This record contains the network distance for the $q \rightarrow m \rightarrow n$ route.

Suppose a is obtained from PQ (4) next. it is expanded and its record, $d_N(q,n) = d_N(q,a) + d_N(a,n)$, is inserted into the PQ. Later, the record for n is obtained from the PQ (5). However, n had already been placed in the CS with $d_N(q,n) = d_N(q,m) + d_N(m,n)$. Now, suppose that if $d_N(q,a) + d_N(a,n) < d_N(q,m) + d_N(m,n)$, the $d_N(q,n)$ in the CS should be replaced with a smaller value, $d_N(q,a) + d_N(a,n)$. This is why the distance for a record in the CS needs to be altered in the SSMTA* algorithm.

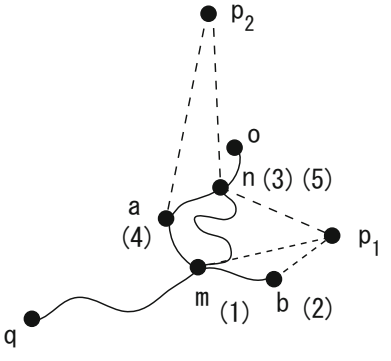


Fig. 3. Cost change in CS

3.3 Incremental Query

The SSMTA* algorithm requires that all elements of the POI set (P) be given in advance of starting the query. If a new POI (r) is added to P after the search has started, the path found from q to point r is not necessarily granted the shortest path. A node on the shortest path from q to r may not have been expanded, because the $Cost$ of a record in the PQ had already been determined when it was inserted, and thus was not changed even though a new point had been added.

After all the shortest paths to P have been found, each node in the CS has an exact shortest path distance from q. Then, when a distance q to any point in the CS is requested to retrieve, the correct distance can be easily obtained by referring to the $d_N(q,r)$. The shortest route from r to q can be obtained by using the basic algorithm described in 3.2.

Papadias et al.[5] proposed an incremental algorithm for k-NN queries, and Yiu et al.[3] proposed an incremental algorithm for ANN queries. In these algorithms, a new target point is incrementally added to set P. The SSMTA* algorithm can handle this situation in one of two ways.

The first way is to simply search for enough POI candidates using a Euclidean distance query and then to validate the road network distance. This process is repeated until the necessary number of POIs has been found. This approach, however, is problematic. The calculation of $d_E^{min}(n,P)$ becomes heavier as the

number of points in P increases because the number of Euclidean distance calculations is proportional to the number of points in P. An even bigger problem is deciding on a sufficient number of points in P before applying the SSMTA* algorithm. The node expansion must also be considered. However, the increase in the total number of node expansions is not proportional to the increase in the number of points in P, because POIs distant from q do not become target until the node expansion approaches the node.

The second way is to re-calculate the PQ. Suppose that the SSMTA* algorithm has already found all the points in P and that the PQ still contains several un-extracted records. If a new point (r) is added to P, $NewCost = d_N(q, N_C) + d_E(N_C, r)$ for all PQ entries is calculated, and $Cost$ is replaced with the $NewCost$ value. In Dijkstra's algorithm, the number of records in the PQ is proportional to the distance to the farthest point in P. In the SSMTA* algorithm, determining the size of the PQ is not so simple. However, the PQ still contains the nodes surrounding the expanded nodes (i.e. the nodes in the CS), so the number of records in the PQ should be proportional to the distance from q to the farthest POI in P. Additionally, the $NewCost$ calculation can be carried out in main memory, so disk access is not necessary. Moreover, this calculation does not affect the total calculation time very much. Besides the number of points to be added is not restricted, so one or more points can be added simultaneously.

4 Application to ANN Queries

In an ANN query, a set of query points Q is given and k POIs are found in ascending order by *evaluation value*, which is based on the distance from a set of query points Q to $p(\in P)$. The proposed distance evaluation functions include *sum*, *max*, and *min* [15]. When *sum* is used, the total distance from each query point in Q to p is calculated and used as the evaluation value.

The initial ANN query method proposed by Papadias et al. [16] was dubbed *group nearest neighbor query*. Since then, several other ANN query methods [15] have been proposed. Yiu et al. [3] proposed three methods for road network distance queries. Experimental evaluation showed that the IER (incremental Euclidean restriction) method outperformed the rest. The IER method is based on a three-step paradigm: 1) search for candidate ANN POIs on the basis of Euclidean distance, 2) evaluate the results on the basis of road network distance, and 3) repeat both steps until kANN POIs have been found, which can be efficiently done using a best-first query on the R-tree index.

Figure 4 shows a simple example application of an ANN query method using the *sum* function. The ANN query points are $q1$ and $q2$. The POIs are $p1$ to $p4$; they are indexed by an R-tree. Initially, the minimum bounding rectangles (MBRs) in the root node of the R-tree are placed in a priority queue (PQ). In this example, PQ=$\{< 6, R1 >, < 11, R2 >\}$. The first item of this record shows the sum of the supposed minimum distances (MINDISTs) from each query point to an MBR. For example, the MINDIST from $q1$ to R1 is 4 and from $q2$ to R1

is 2, so R1 has total cost of 6. Therefore, $< 6, R1 >$ is dequeued because its cost is lesser. Descending the R-tree one level results in two points $p1$ and $p2$ in a leaf node being enqueued so that the PQ=$\{< 10, p1 >,< 11, R2 >,< 15, p2 >\}$. The dequeuing of $p1$ means that one ANN POI has been found. The next step is verification in which the sum distance on the road network is calculated. The ANN POIs can thus be found incrementally on the basis of the Euclidean distance. These generation and verification steps are repeated until the minimum sum of the road distance fall below the n-th ANN on Euclidean distance. This algorithm can easily be expanded to handle kANN queries, meaning that it can be used to search up to the k-th minimum ANN. This algorithm can also be adapted simply to use the *min* and *max* functions [15].

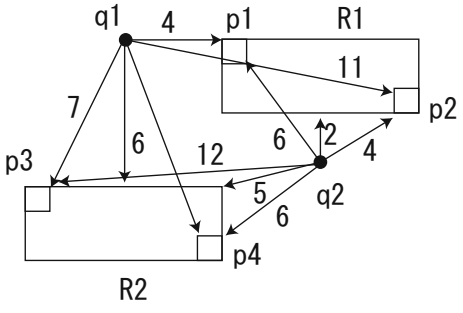


Fig. 4. ANN search on R tree

Yiu et al. [3] proposed evaluating the distance on a road network using an A* algorithm. They used a pair-wise A* algorithm, so $|Q|$ pairs of calculations were necessary. The SSMTA* algorithm can be adapted to do the same calculation and it should be able to do more quickly. Two types of methods using the SSMTA* algorithm can be used for ANN applications. One type is straightforwardly adopting for each candidate of ANN result by Euclidean distance as starting point, and each in Q as the target. The second type is adapting the SSMTA* algorithm so that one of the q ($\in Q$) is used as the starting point and the kANN result set is used as the destination.

Given the characteristics of the SSMTA* algorithm, we can say that the first method outperforms the second when $|Q|$ is large, and the second method is more suitable when k is large.

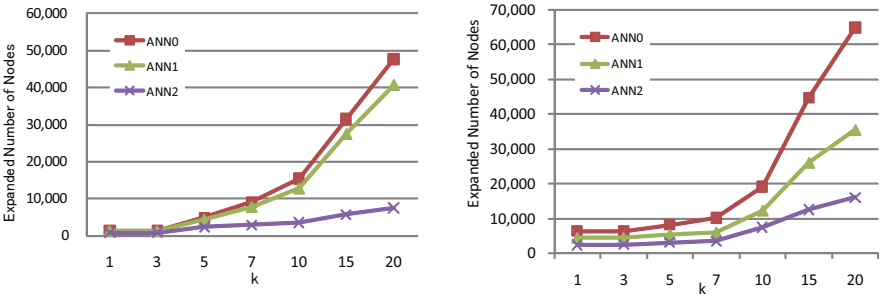
5 Performance Evaluation

We experimentally evaluate the efficiency of the two types of SSMTA* algorithm methods for ANN searches. We used a real road map and generated POIs by using pseudo-random sequences for a variety of existing probabilities *Prob* for a

road segment. For example, $Prob = 0.001$ means that 1000 road segments have one POI.

The ANN candidates were obtained using Yiu et al.'s method described above. The query cost was calculated using three methods. The first one used the conventional A* algorithm as is used in the IER method [3], hereafter called ANN0. The second method used the SSMTA* algorithm to calculate the distance from each ANN query result targeting all query points in Q, hereafter called ANN1. The third calculation method used the SSMTA* algorithm to calculate the distance from each query point to the result points found in a Euclidean kANN search, hereafter called ANN2.

In the Dijkstra, A*, and SSMTA* algorithms, four data sets are referred while searching: PQ, CS, adjacency list, and road segments. The PQ and CS are usually small enough to reside in memory, so they can be quickly accessed. In contrast, the adjacency list and road segments are usually large to reside in memory so they are stored on disk. Node expansion is thus inevitably needed to access them, meaning that node expansion time dominates processing time.



(a) $|Q| = 3$ (b) $|Q| = 7$

Fig. 5. Relation between k and expanded node number

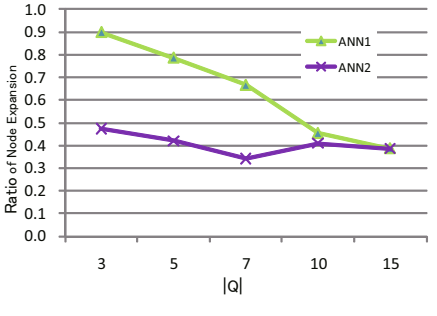


Fig. 6. Relation between $|Q|$ and expanded node number

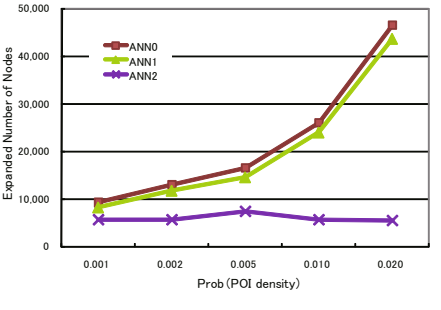


Fig. 7. Relation between $Prob$ and expanded node number

We thus evaluated the performance of the methods on the basis of the number of node expansions.

The first experiment was conducted with $|Q| = 3$ and $|Q| = 7$ and with the number of k in a kANN search variable from 1 to 20. As shown in Fig. 5, the ANN1 calculation method did not show much improvement compared with ANN0, especially for $Q = 3$. In contrast, the ANN2 method showed considerable improvement compared to the other two methods, especially when k was large. For example, when k was 20 and $|Q|$ was 3, the expanded node number was about 20% that of the other methods.

Figure 6 shows the result, when k was fixed at 5 and $|Q|$ was varied. The actual expanded node number for this query varies with the size of the Q distribution area on the map. The actual results we obtained experimentally were evaluated using two methods, each as a ratio to the ANN0 result. The ANN2 ratio was fairly stable because the searched k was the same for all Q. In contrast, the ANN1 ratio decreased with an increase in $|Q|$.

Figure 7 shows the results when *Prob* (the probability of a POI existing on each road segment) was varied and $|Q| = 3$ and k=5. With the ANN0 and ANN1 methods, the number of node expansions increased with the *Prob*, while, with the ANN2 method, the number was fairly stable and independent of *Prob*.

These results show that the two types of methods using the SSMTA* algorithm have two intrinsic characteristics: the efficiency of the ANN1 method depends on the number of k in a kANN query while that of the ANN2 method depends on $|Q|$. This is because the SSMTA* algorithm works more efficiently when the number of POIs to be searched for is large. Unlike the conventional A* algorithm, which suffers heavily from duplicated node expansions especially around the searched POI, the SSMTA* algorithm avoids duplicate node expansions.

6 Conclusion

We have described an algorithm for efficiently finding the shortest paths from one starting point to several targets on a road network, and have demonstrated that it outperforms the conventional A* algorithm when applied to a kANN query especially when the number of target points is large.

The SSMTA* algorithm can be adapted to several types of spatial queries based on road network distance including CkNN queries, spatial skyline queries, and trip planning queries. We plan to apply the SSMTA* algorithm to such queries as future work.

Acknowledgments. This study was partially supported by the Japanese Ministry of Education, Science, Sports and Culture (Grant-in-Aid for Scientific Research (C)), 21500093, and by Transdisciplinary Research Integration Center at the Research Organization of Information and Sciences, Japan.

References

1. Dijkstra, E.W.: A note on two problems in connection with graphs. Numeriche Mathematik 1, 269–271 (1959)
2. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions of Systems Science and Cybernetics SSC-4(2), 100–107 (1968)
3. Yiu, M.L., Mamoulis, N., Papadias, D.: Aggregate nearest neighbor queries in road networks. IEEE Transactions on Knowledge and Data Engineeing 17(6), 820–833 (2005)
4. Deng, K., Zhou, X., Shen, H.T.: Multi-source skyline query processing in road networks. In: Proceeding of IEEE 23rd International Conference on Data Engineering (2007)
5. Papadias, D., Zhang, J., Mamoulis, N., Tao, Y.: Query processing in spatial network databases. In: Proc. 29th VLDB, pp. 790–801 (2003)
6. Sharifzadeh, M., Kalahdouzan, M.R., Shahabi, C.: The optimal sequenced route query. Technical report, Computer Science Department, University of Southern Calfornia (2005)
7. Li, F., Cheng, D., Hadjieleftheriou, M., Kollios, G., Teng, S.H.: On Trip Planning Queries in Spatial Databases. In: Anshelevich, E., Egenhofer, M.J., Hwang, J. (eds.) SSTD 2005. LNCS, vol. 3633, pp. 273–290. Springer, Heidelberg (2005)
8. Hu, H., Lee, D.L., Lee, V.C.: Distance indexing on road networks. In: Poc. 32nd VLDB, pp. 894–905 (2006)
9. Samet, H., Sankaranarayanan, J., Alborzi, H.: Scalable network distance browsing in spatial databases. In: Proc. of the ACM SIGMOD Conference, pp. 43–54 (2008)
10. Kolahdouzan, M., Shahabi, C.: Voronoi-based K nearest neighbor search for spatial network databases. In: Proc. 30th VLDB, pp. 840–851 (2004)
11. Zhu, L., Sun, Y.J.W., Mao, D., Liu, P.: Voronoi-based aggregate nearest neighbor query processing in road networks. In: ACM GIS 2010 (2010)
12. Shaw, K., Ioup, E., Sample, J., Abdelguerfi, M., Tabone, O.: Efficient approximation of spatial network queries using the M-tree wirh road network embedding. In: 19th International Conference on Scientific and Statistical Database Management (2007)
13. Ciaccia, P., Patella, M., Zezula, P.: M-tree: An efficient access method for similarity search in metric spaces. In: Proceedings of the 23rd VLDB Conference, pp. 426–435 (1997)
14. Ioup, E., Shaw, K., Sample, J., Abdelguerfi, M.: Efficient AKNN spatial network queries using the M-tree. In: ACM GIS 2007 (2007)
15. Papadias, D., Tao, Y., Mouratidis, K., Hui, C.K.: Aggregate nearest neighbor queries in spatial databases. ACM Transactions on Database Systems 30(2), 529–576 (2005)
16. Papadias, D., Shen, Q., Tao, Y., Mouratidis, K.: Group nearest neighbor queries. In: Proceedings of the 20th International Conference on Data Engineering, pp. 301–312 (2004)

Combining Top-*k* Query in Road Networks

Weimo Liu, Yinan Jing, Kunjie Chen, and Weiwei Sun

School of Computer Science, Fudan University,
Shanghai, China
{liuweimo,jingyn,chenkunjie,wwsun}@fudan.edu.cn

Abstract. In location-based services, every query object usually has multiple attributes including its location in road networks. However, when making a decision to choose an object, there is probably no such an object that is best in every attribute. To have a balance among all the attributes, a monotone aggregation function can be used, in which every attribute is an independent variable of the function. To find *k* objects which have the minimal (maximal) values of the function is a typical combining top-*k* problem. In this paper, we address this problem in road network environment. To answer such a query more efficiently, we propose a novel algorithm called ATC (Access with Topology Changed). By making use of road networks' locality, the algorithm changes the networks' topology and reduces the number of data access. Extensive experiments show that our algorithm can obviously outperform existing algorithms that solve the combining top-*k* problem.

Keywords: Top-*k* Query, Spatial Index, Road Networks, Multi-Objective Decision.

1 Introduction

As mobile devices become increasingly widely used in recent years, the location based services develop rapidly. The demand for location based services is becoming higher and higher. In most of the problems of the former research, most existing service providers often consider only geographical information. For example, when receiving a user query for a hotel (Fig. 1(a)), the service provider usually returns the nearest one to the user, and does not consider other factors. However, there are many other factors should be taken into account for users in real life. For instance, when users want to find a hotel, they often consider the distance between the hotel and the query point, the price, the service quality and so on. When making such a decision, there are several methods to take these factors into consideration. The first method is to set a boundary for each attribute. That is users could usually filter their choices by restricting the value of each attribute within some limits. For example, a user will consider following conditions of a hotel: (1) its price is less than 1,000 dollars; (2) it is less than five kilometers away from the query point; (3) its service is not bad. But it is often difficult to set the boundary. If the boundary is too broad, there will be too many results. Conversely, there may be not enough results if it is too strict.

L. Wang et al. (Eds.): WAIM 2011 Workshops, LNCS 7142, pp. 63–75, 2012.

The second method is to select the best results according to a single attribute. But it is easy to lose some preferable results in this way, especially when the discrimination of the selected attribute is not great. For example, we suppose there are two hotels as shown in Fig. 1(b): hotel *j*'s price is 550 dollars, and the distance between *j* and the query point is 8 kilometers, while hotel *f*'s price is 560 dollars, and the distance between *f* and the query point is 1.4 kilometers. If the user prefers sorting hotels by their prices and then choose the cheapest one, the user will get the result that *j* is better. However, in this case, hotel *f* is obviously a better answer.

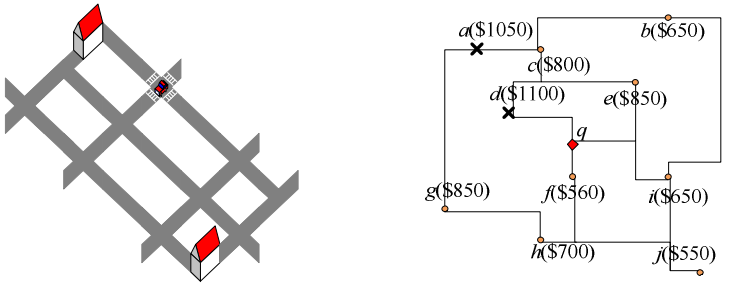


Fig. 1. Query for a hotel

To have a balance among all of the attributes, a monotone aggregation function *f* can be used, in which every attribute is an independent variable of the function. To find *k* objects which have the minimal (maximal) values of the function is a typical combining top-*k* query problem. In this paper, we address the location-dependent top-*k* query in road networks, in which the objects' location will be taken into account. For example, when the user selects hotels, two factors are taken into account, i.e. the price and the distance. Supposing that we use a monotone function *f*(*distance*, *price*) according to the user's demand, the hotel with the smallest function value will be the answer. When a user needs top-*k* results, the *k* hotels with the smallest value of *f* will be returned. However, due to complex distance computation in road networks, the traditional solution to this query problem is not efficient enough. In this paper, a novel algorithm called ATC (Access with Topology Changed), in which we use a strategy to reduce the number of data access by changing the network structure without losing critical information, is proposed to improve the query performance. Our contributions can be summarized as follows:

(1) We address top-*k* query in road networks to solve one certain query problem in real life which needs to consider multiple attributes including object's location. And to answer this type of query efficiently, we propose a novel algorithm.

(2) We also make many extensive experiments to evaluate our algorithm. Experiment results show that our algorithm significantly outperforms other algorithms such as NRA [1].

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 formally presents the concept of top-*k* query in road networks and related definitions and notations. Section 4 presents algorithms for processing top-*k* query in road networks and section 5 evaluate our algorithms by experiments. Finally, we draw the conclusion for this paper.

2 Related Work

The traditional location based services only consider geographical information. Nearest Neighbor Query is a common query in spatial database management system [2][3]. It appears in the scene that a user wants to find the nearest target of a certain type. For example, when a driver wants to find a gas station, the nearest one is usually the best choice. Methods to solve this type of problem have been deeply studied. The most widely used one is Incremental Nearest Neighbor algorithm which is presented by G. R. Hjaltason and H. Samet [4]. Then, the researchers focus on the road networks since these kinds of location based services are quite prevalent in the real world [5]. However, in real life we need to simultaneously consider several factors including both spatial and non-spatial attributes. To balance the factors, we need a method that can take several attributes into account at the same time.

Skyline query is to find out a set of points which is not dominated by any other points [6]. For example, a hotel is a candidate when there is no other hotel that is cheaper and closer to the beach than it. The skyline query was also applied in spatial database, namely the spatial skyline query [7][8], which regards the distances from different query points as multi-attributes. However, when the user needs a hotel which is cheaper and closer, either the general or spatial skyline query cannot meet the requirements because the distance between the hotel and the user is dynamic. To solve this problem, Zheng et al present the Location-Dependent Skyline Query [9]. The skyline query is a good way to solve the multi-objective optimal problem, but it has two drawbacks: (1) there may be too many results of the skyline query; (2) it cannot find out which is the second best choice.

Multi-objective decision is to simultaneously optimize several conflicting objectives subjected to certain constraints. Multi-objective decision appears in various fields: business, finance, industry, or wherever optimal decisions need to be made to have a balance among two or more conflicting objectives. Maximizing profit and minimizing the cost of a transaction; maximizing profit and minimizing the time of a loan; and minimizing cost of labor and freightage are examples of multi-objective optimal problems. Constructing a single Aggregate Objective Function (AOF) is the most widely-used approach to make multi-objective decision. The basic idea is to combine all the objective functions into a single functional form, called the AOF. A well-known method to combine the attributes is the weighted linear sum of the objectives. Each weight is for one objective to be optimized, and then combines all

the items into a single function. Obviously, the solution obtained will be determined by the values of the weights specified. Of course the weighted linear sum is not the only way to combine all the attributes. Any monotone aggregation function can be used. To find the k maximal (minimal) value of the AOF is top-k query. The threshold algorithm to solve this problem is first proposed by Fagin [1]. And he designed algorithms for the different situations respectively. When objectives are about spatial information in Euclidean space, it is a top-k spatial preference query [10][11] or location based top-k query [12]. M. L. Yiu et al. in [10] and H. Jung et al. in [12] proposed aR-Tree to prune the useless objects. If the spatial information is in form of road networks, those algorithms will not work.

In this paper, we address the top-k query in road networks to choose k results with the best value of one AOF. Compared with NN(Nearest Neighbor), it allows users to consider the non-spatial factors. And the difference from location-dependent skyline query is users can get several optimal results according to their purpose by the location dependent top-k query rather than have a large amount of candidates. We consider the properties of road networks and solve the top-k query in road networks more efficiently than traditional algorithms.

3 The Model

Assume that there is a set of spatial objects O in the database and each object $o \in O$ has spatial information and non-spatial attributes. Every non-spatial attribute can be represented by a scalar quantity. Then it is appropriate to note the i^{th} field of o with x_i, $x_i \in R$. Specify that x_1 represents the distance between the query point and o.

A Single Aggregate Objective Function (AOF). *Suppose there is a function $f(x_1, x_2, \ldots, x_n)$. x_i is a scalar quantity which represents an attribute of o.*

$$\forall x_i, \frac{\partial f(x_1, x_2, \ldots, x_n)}{\partial x_i} \geq 0 \left(\frac{\partial f(x_1, x_2, \ldots, x_n)}{\partial x_i} \leq 0 \right),$$

i.e. $f(x_1, x_2, \ldots, x_n)$ is a monotone function of every attribute. We define such a function as the single aggregate objective function (AOF).

Top-k Query

Situation 1: Suppose that $f(x_1, x_2, \ldots, x_n)$ is monotone increasing with every independent variable. If we want an object with a smaller value of x_i (i=1, 2, 3, … , n), the best choice is the object with the minimal value of function.

Situation 2: If the $f(x_1, x_2, \ldots, x_n)$ is monotone decreasing with every independent variable, the best choice is the maximum instead of minimum.

Situation 3: When $f(x_1, x_2, \ldots, x_n)$ is monotone but the monotonic is not the same with all the independent variables. In this case, we can change the sign of some independent variables and convert the function to situation 1 or situation 2.

The top query is to select the best choice in above situations. When we ask a top-k query, the k objects with the minimal value of function in situation 1 or the maximal value of function in situation 2 can meet the demand.

Top-*k* Query in Road Networks. *In top-k query, if* x_i (i = 1, 2, 3, … , n) , x_i *is the distance between the query point and the object o in the road networks, we called the query Top-k query in road networks.*

In this paper, we suppose that a monotone function $f(x_1, x_2, \dots, x_n)$ is ascending with $x_i(i=1, 2, \dots, n)$ and, consequently, the minimal value of the $f(x_1, x_2, \dots, x_n)$ is the best choice in the problem. If $f(x_1, x_2, \dots, x_n)$ does not satisfy the conditions, changing the sign of some $x_i(i=1, 2, \dots, n)$ can help. Specially, we assume that x_1 is the distance from the query point in road networks.

4 Algorithm

4.1 NRA (No Random Accesses) Algorithm

NRA (No Random Accesses) algorithm in this paper is a method to solve top-k query in road network. It is a variant of the threshold algorithm[1] in the situation that we can only get the objects' attribute by sequence of this attribute's sorted list, because the cost of computing the distance between two points in road networks is high.

To present the algorithm conveniently, we first define the following functions:

$S(O) = \{i_1, i_2, \dots, i_m\} \subset \{1, 2, \dots, n\}$ *is the known fields of the object O.* (*When we have already known the* i^{th} *field's value of O, it is a known field.*)

$W_S(O)$ *is the maximal value of the AOF* $f(x_1, x_2, \dots, x_n)$ *that can attain for object O. If f is monotone, f obtains the maximal value when* $x_i = \infty$, $i \in \{1, 2, \dots, n\} - S(O)$. *For example, if* $S(O) = \{1, 2, \dots, m\}$, $W_S(O) = f(x_1, x_2, \dots, x_m, \infty, \dots, \infty)$.

$B_S(O)$ *is the minimal value of the AOF* $f(x_1, x_2, \dots, x_n)$ *that can attain for object O. If f is monotone, f obtains the minimal value when* x_i *evaluates the bottom value* x_i', $i \in \{1, 2, \dots, n\} - S(O)$. *For example, if* $S(O) = \{1,2,\dots,m\}$, $B_S(O) = f(x_1,x_2,\dots,x_m,x_{m+1}',\dots,x_n')$. The concept of "bottom value" will be explained in the following algorithm.

The NRA algorithm first sorts the objects in a list L_i (i=1, 2, … , n) according to the value of the i^{th} attribute of the objects. So there are n sorted lists for the n attributes. Specially, L_1's sorted access is by Dijkstra's algorithm. The next step of NRA is to traverse the n sorted lists in a way called "sorted access", that is, retrieving the d^{th} element in each sorted list in the d^{th} iteration of this step. We use O_{id} to note the d^{th} object in the i^{th} sorted list. The bottom value x_i' is evaluated by the value of i^{th} attribute of O_{id} in the current iteration. After retrieving the n objects, the $S(O)$ of each accessed object in this iteration will be updated. Then the $W_s(O)$ and $B_s(O)$ will be computed again according to the new value of $S(O)$ and the new bottom values.

During the procedure of the sorted access, NRA uses a top k list $T_k^{(d)}$ to keep track of the k accessed objects with the smallest W value seen so far. If two objects have the same W value, the object with a smaller B value will be inserted into the top k list first. Let $M_k^{(d)}$ be the k^{th} lowest W value in $T_k^{(d)}$. NRA will end when no object outside $T_k^{(d)}$ has a B value smaller than $M_k^{(d)}$, that is, when $B(O) \geqq M_k$ for all $O \notin T_k$.

```
    NRA()
1   T_k ← NEWPRIORITYQUEUE()
        // T_k is a list which contains top k element of a priority queue sorted by W
    in ascending order, if W is the same, ties are broken using B, the lower B
    wins.
2   count ← 0
3   for d ← 1 to N do
4       O ← findmin()
5       x_1' ← O.x_1
6       M_k ← T_k [k]
7       if (O is distinct) do
8           count++
9       if (count >= k and f(x_1', x_2', … , x_n' ) >=M_k) do
10          halt
11      else do
12          ENQUEUE (T_k, O, f(O))
13      for i ← 2 to n do
14          O ← L_i(d)
15          x_i' ← O.x_i
16          M_k ← T_k [k]
17          if (O is distinct) do
18              count++
19          if(count >= k and f(x_1', x_2', … , x_n' ) >=M_k) do
20              halt
21          else
22              ENQUEUE (T_k, O, f(O))
23  Output(T_k)
```

Fig. 2. The NRA Algorithm

4.2 Access with Topology Changed (ATC) Algorithm.

Index Structure

(1) **Index the Data with R-Tree**

In Euclidean space, we can compute the distance between point $A(x_1, y_1)$ and point $B(x_2, y_2)$ by the below formula directly

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{1}$$

The distance in Euclidean can be used as the lower bound of that in road networks. So we first index the data with R-tree [13], in which the query point's distance from the MBR (Minimal Bound Rectangle) is the lower bound of the distance from all the points in the MBR.

(2) **Combing the Attributes with aR-Tree**

Now we consider to combine the indexes of different attributes into an integrate one. Since we already index the objects by R-tree based on their spatial information, the following step is to store the other attributes in the R-tree's leaf nodes. Then find out the minimal values of non-spatial attributes of every non-leaf node's objects and store them in the non-leaf nodes. We call the R-tree added the non-spatial attributes aR-Tree [12]. For the non-leaf node, x_1 equals the distance between the query point and the node's MBR. Let $x_i(i=2, \ldots, n)$ equals the minimum to make the $f(x_1, x_2, \ldots, x_n)$ get the minimal value. And this minimal value is the non-leaf node's value of $f(x_1, x_2, \ldots, x_n)$.

We take the aR-Tree in Fig. 3 as an example. Each object has one non-spatial attribute in the aR-Tree. R_1 is the MBR which include the object *A*, *B*, *C*, *D*. The non-spatial attributes of them is 500, 200, 600, 300. So R_1's non-spatial attribute is 200, the minimum of the non-spatial attributes of its objects. And for the same reason, *R*'s non-spatial attribute is 100, the minimum of the non-spatial attributes of R_1, R_2, R_3 and R_4.

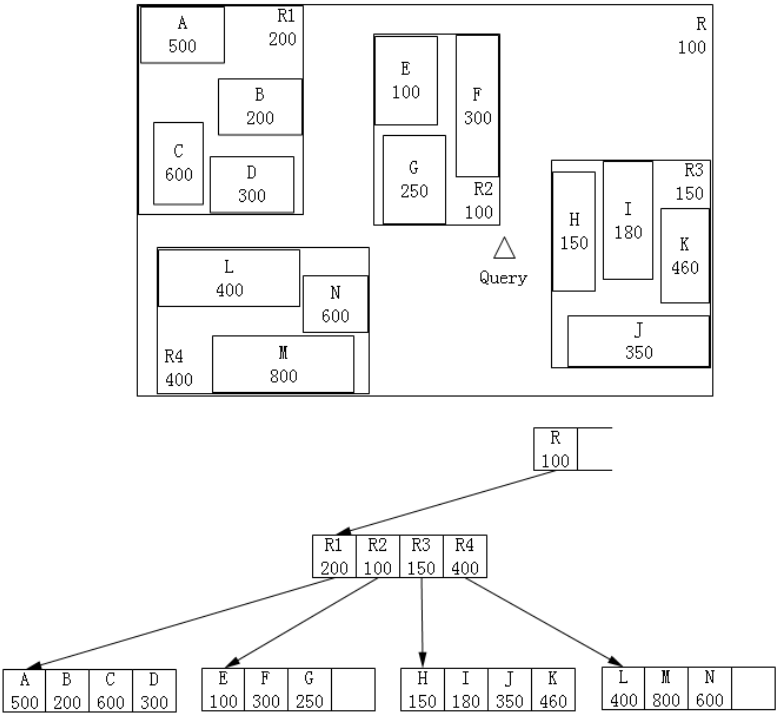


Fig. 3. The aR-Tree

Definition

d (*a, b*)*:* the network distance between point *a* and point *b*.

Inner point: the point which is not connected directly with points in the other disk pages.

Current page: the page in which the return of the findmin() function is when we execute the Dijkstra's algorithm.

We propose two methods to change the topology structure of the road networks.

ATC-I. Provided that there is an inner point *p* in the page, *S* is a set of the points which is connected directly with *p*. Choose any two points *a*, *b* in *S*. If there is no edge directly connecting *a* and *b*, join these two points with a new edge of which the weight equals the sum of *d*(*p*, *a*) and *d*(*p*, *b*). Otherwise compare the weight of the edge existed and the sum of *d*(*p*, *a*) and *d*(*p*, *b*). If the sum is smaller, replace the weight with the sum. Repeat this step to any two points in *S*. Then delete *p* and edges adjacent to *p*. For example as Fig. 4 shows, add edges *AG* and *DG* to (a), correspondingly *d*(*A*, *G*)=*d*(*A*, *F*)+*d*(*F*, *G*), *d*(*D*, *G*)=*d*(*D*, *F*)+*d*(*F*, *G*), then delete *AF*, *DF*, *FG* and (a) transforms to (b). Add *AC* and *CD*, let *d*(*A*, *C*)=*d*(*A*, *G*)+*d*(*G*, *C*), *d*(*C*, *D*)=*d*(*D*, *G*)+*d*(*G*, *C*). And (b) transforms to (c).Repeat the above operations with all inner points in this page.

ATC-II. Provided that there is an inner point in the page, compute all pairs' minimal distance in networks and store them in a matrix called Minimum Distance Matrix. Construct a complete graph corresponding to the Minimum Distance Matrix. Then replace the points in this page and the edges between them with the complete graph. Finally, delete the inner points and the edges connect with them. (Fig. 5)

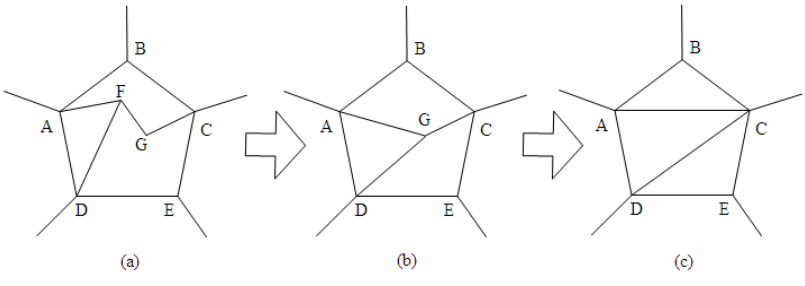


Fig. 4. Change Method I

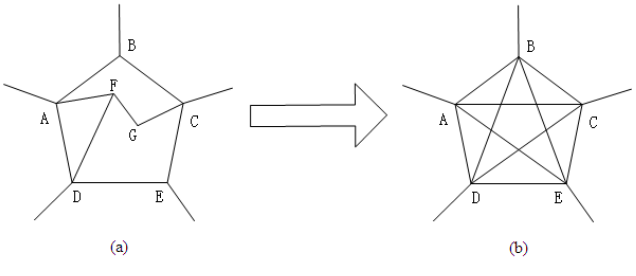


Fig. 5. Change Method II

The Algorithm

First sort the objects in a list L_i (i=1, 2, … , n) according to the value of the i^{th} attribute of the objects. So there are n sorted lists for the n attributes. Specially, L_1's sorted access is by Dijkstra's algorithm. When the current page's value of $f(x_1, x_2, \dots, x_n)$ is greater than the k^{th} smallest object, we use Method 1 to change the topology structure in ATC-I while Method 2 is used in ATC-II. The next step of NRA is to traverse the n sorted lists in a way called "sorted access", that is, retrieving the d^{th} element in each sorted list in the d^{th} iteration of this step. We use O_{id} to note the d^{th} object in the i^{th} sorted list. The bottom value x_i' is evaluated by the value of i^{th} attribute of O_{id} in the current iteration. After retrieving the n objects, the $S(O)$ of each accessed object in this iteration will be updated. Then the $W_s(O)$ and $B_s(O)$ will be computed again according the new value of $S(O)$ and the new bottom values. During the procedure of the sorted access, NRA uses a top k list $T_k^{(d)}$ to keep track of the k accessed objects with the smallest W value seen so far. If two objects have the same W value, the object with a smaller B value will be inserted into the top k list first. Let $M_k^{(d)}$ be the k^{th} lowest W value in $T_k^{(d)}$. NRA will end when no object outside $T_k^{(d)}$ has a B value smaller than $M_k^{(d)}$, that is, when $B(O) \geqq M_k$ for all $O \notin T_k$.

```
    ATC()
1   T_k ← NEWPRIORITYQUEUE()
        // T_k is a list which contains top k element of a priority queue sorted by
    W, if W is the same, ties are broken using B
2   count ← 0
3   for d ← 1 to N do
4       O ← findmin()
5       x_1' ← O.x_1
6       changeTopologyStructure(Method, O.pageId)
7       M_k ← T_k [k]
8       if (O is distinct) do
9           count++
10      if (count >= k and f(x_1', x_2', … , x_n' ) >=M_k) do
11          halt
12      else do
13          ENQUEUE (T_k, O, f(O))
14      for i ← 2 to n do
15          O ← L_i(d)
16          x_i' ← O.x_i
17          M_k ← T_k [k]
18          if (O is distinct) do
19              count++
20          if(count >= k and f(x_1', x_2', … , x_n' ) >=M_k) do
21              halt
22          else
23              ENQUEUE (T_k, O, f(O))
24  Output(T_k)
```

Fig. 6. The ATC Algorithm

5 Performance Evaluation

5.1 Experiment Settings

The experiment is on two data sets: California (Cal) and North America (NA) [14]. The coordinates of points and the networks are real while the non-spatial attributes are generated randomly (Table 1).The problem is to find k hotels which are both closer to the user and cheaper in Cal. For NA, it selects k possible locations for a branch.

To show the relationship of the variables clearly, as an example, we use the simple linear combination function of two variables as the AOF. Let the function be $f(x_1, x_2) = x_1+\lambda*x_2$. x_1 notes the distance between the object and the query point while x_2 represents the price of the hotel. Considering the price of gasoline and hotel in reality, λ is of the order of 0.01 in Cal. And for NA, x_2 represents daily operating cost and λ is in order of 10 based on the cost of transportation and daily operating. Of course any other multivariate monotone function is also valid.

We use R-tree as the basic index data structure [13][15]. [15] is an implementation of R-Tree. (The algorithms can also be applied to other hierarchy index data structures.)

Table 1. Summary of the Dataset

Dataset	Number of Points	Number of Edges	Page Size(B)	Attribute
California	20148	21693	4096	Coordinate, Price
North America	175813	179179	4096	Coordinate, Cost

5.2 Experiments on the Efficiency

We measure the performance of algorithms in road networks by disk I/O in dataset California and NA. The query is top-10 and the capacity of aR-tree is 10. Let λ in f(x1, x2) be distributed over a large extent. The Fig.7 shows that we have a universal better performance than the NRA algorithm in dataset California. The result is similar in dataset North America (Fig. 8). The curves indicate the disk I/O increases slowly with λ, because the non-spatial attributes weights more, the locality of the AOF's value in the aR-Tree is worse.

In dataset Cal, we computed that λ is about 0.01 based on the price of hotel in reality. In dataset NA, we get that the λ is 10. So the next step is to make λ in the order of 0.01 (Fig. 9) in dataset Cal and 10 (Fig. 10) in dataset NA. We can see that our algorithms perform better than the NRA algorithm all the time. Specially, the ratio of the ATC-II's and NRA's disk I/O is less than 1/2 in dataset California and much smaller in NA. This depends on capacity of the aR-tree and the character of the dataset's topological structure and coordinate information. Since the NRA's disk I/O is always more than twice ATC-II's. Greater is λ, larger is the difference of disk I/O.

Let us focus on how the *k* in query affects the performance of the algorithms. λ is 0.01 in Cal (Fig. 11) and 10 in NA (Fig.12). Since the greater *k* demands the more objects to check, the disk I/O is increasing with *k*. The disk I/O of NRA is also more than twice ATC-II's in different *k*. The ratio of NRA's and ATC-II's disk I/O is little relative to *k*.

Finally, we compare the performance of algorithms when selecting different aR-tree's capacity. Since the dataset is the same one and λ is 0.01 in Cal and 10 in NA and query is top-10, the only difference is aR-tree's capacity. The Fig. 13 and Fig. 14 show that NRA's performances are almost not affected by the change of capacity while the ATC-I and ATC-II's are improved by increasing the capacity of aR-tree. This is because the bigger capacity leads to the more inner points. When we could select the bigger capacity, the ATC-I and ATC-II perform better.

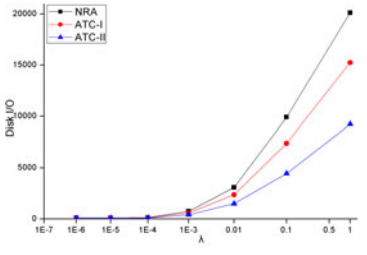


Fig. 7. Disk I/O for λ in large extent(Cal)

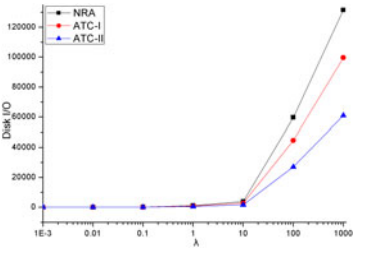


Fig. 8. Disk I/O for λ in large extent(NA)

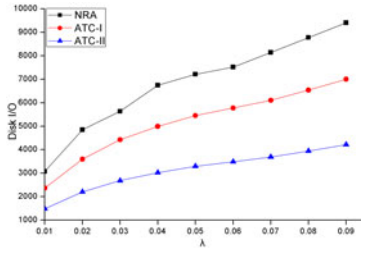


Fig. 9. Disk I/O for λ in order of 0.01(Cal)

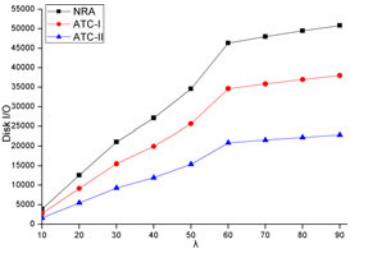


Fig. 10. Disk I/O for λ in order of 10(NA)

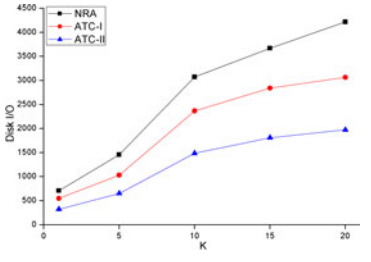


Fig. 11. Disk I/O for *k*(Cal)

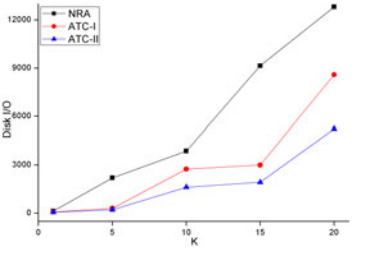


Fig. 12. Disk I/O for *k*(NA)

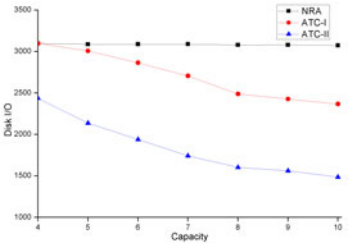


Fig. 13. Disk I/O for Capacity(Cal)

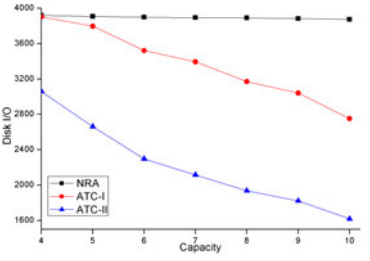


Fig. 14. Disk I/O for Capacity(NA)

6 Conclusion

In this paper, considering the wide applications, we propose a novel efficient algorithm for the top-*k* query in road networks In this algorithm, we combine the traditional R-tree index with the non-spatial attributes and transform the network topology structure without losing the useful information. Thus those pages in which there are no possible results can be pruned. Extensive experiment results show that this new algorithm can achieve better efficiency than the traditional multi-objective decision algorithm.

Acknowledgment. Thanks to FDUROP (Fudan's Undergraduate Research Opportunities Program). This research is also supported in part by the National Natural Science Foundation of China (NSFC) under grant 61073001.

References

1. Fagin, R.: Combining Fuzzy Information: an Overview. ACM SIGMOD Record 31(2), 109–118 (2002)
2. Roussopoulos, N., Kelley, S., Vincent, F.: Nearest Neighbor Queries. ACM SIGMOD Record (1995)
3. Cheung, K.L., Fu, A.W.: Enhanced Nearest Neighbour Search on the R-tree. ACM SIGMOD Record 27(3), 16–21 (1998)
4. Hjaltason, G.R., Samet, H.: Distance Browsing in Spatial Databases. TODS 24(2), 265–318 (1999)
5. Yoo, J.S., Shekhar, S.: In-route nearest neighbor queries. Geoinformatica (2005)
6. Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: Proc. IEEE Conf. on Data Engineering, Heidelberg, Germany (2001)
7. Sharifzadeh, M., Shahabi, C.: The spatial skyline queries. In: VLDB 2006: Proceedings of the 32nd International Conference on Very Large Data Bases, pp. 751–762. VLDB Endowment (2006)
8. Son, W., Lee, M.W., Ahn, H.K., Hwang, S.W.: Spatial Skyline Queries: An Efficient Geometric Algorithm. In: Mamoulis, N., Seidl, T., Pedersen, T.B., Torp, K., Assent, I. (eds.) SSTD 2009. LNCS, vol. 5644, pp. 247–264. Springer, Heidelberg (2009)
9. Zheng, B., Lee, K.C.K., Lee, W.C.: Location-Dependent Skyline Query. In: Proc. of MDM, pp. 148–155 (2008)

10. Yiu, M.L., Dai, X., Mamoulis, N., Vaitis, M.: Top-k spatial preference queries. In: ICDE (2007)
11. Rocha-Junior, J.B., Vlachou, A., Doulkeridis, C., Norvag, K.: Efficient processing of top-k spatial preference queries. In: VLDB (2010)
12. Jung, H., Cho, B., Chung, Y.D., Liu, L.: On processing location based top-k queries in the wireless broadcasting system. In: ACMSAC (2010)
13. Guttman, A.: R-trees: a Dynamic Index Structure for Spatial Searching. ACM SIGMOD Record, 47–57 (1984)
14. Real Datasets for Spatial Databases: Road Networks and Points of Interest – California Road Network and North America Road Network, `http://www.cs.fsu.edu/~lifeifei/SpatialDataset.htm`
15. Hadjieleftheriou, M., Hoel, E., Tsotras, V.J.: Sail: A spatial index library for efficient application integration. Geoinformatica 9(4) (2005)

Extracting Focused Locations for Web Pages

Qingqing Zhang, Peiquan Jin, Sheng Lin, and Lihua Yue

University of Science and Technology of China
jpq@ustc.edu.cn

Abstract. Most Web pages contain location information, which can be used to improve the effectiveness of search engines. In this paper, we concentrate on the focused locations, which refer to the most appropriate locations associated with Web pages. Current algorithms suffer from the ambiguities among locations, as many different locations share the same name (known as GEO/GEO ambiguity), and some locations have the same name with non-geographical entities such as person names (known as GEO/NON-GEO ambiguity). In this paper, we first propose a new algorithm named *GeoRank,* which employs a similar idea with *PageRank* to resolve the GEO/GEO ambiguity. We also introduce some heuristic rules to eliminate the GEO/NON-GEO ambiguity. After that, an algorithm with dynamic parameters to determine the focused locations is presented. We conduct experiments on two real datasets to evaluate the performance of our approach. The experimental results show that our algorithm outperforms the state-of-the-art methods in both disambiguation and focused locations determination.

Keywords: Web Search, Geographical information, GEO/GEO ambiguity, GEO/NON-GEO ambiguity, Focused locations.

1 Introduction

Web search engines such as Google and Bing have been an important part in people's life. However, existing search engines do not pay enough attention to the location information in Web pages. For example, it is difficult to express queries like "to find the retailer promotion about Nike in Beijing" in Google. On the other side, location, or in other words, the spatial dimension, is one of essential characteristics of information, and most Web pages are associated with certain locations, e.g., news report, retailer promotion and so on. A recent study in the literature [21] reported that among 2,500 queries, 18.6% of them contained a geographic predicates and 14.8% of them included a place name. Therefore, how to extract locations for Web pages and then use them in Web search process has been a hot and critical issue in current Web search.

As a Web page usually contains two or more location words, it is necessary to find the focused locations of the Web page. The focused locations represent the most appropriate locations associated with contents of a Web page. Generally, we assume that each Web page has several focused locations. The most difficult issue in determine focused locations is that there are GEO/GEO and GEO/NON-GEO

L. Wang et al. (Eds.): WAIM 2011 Workshops, LNCS 7142, pp. 76–89, 2012.

ambiguities existing in Web pages. The GEO/GEO ambiguity refers that many locations can share a single place name. For example, Washington can be 41 cities and communities in the United States and 11 locations outside [5]. The GEO/NON-GEO ambiguity refers that a location name can be used as other types of names, such as person names. For example, Washington can be regarded as a person name as George Washington and as a location name as Washington, D.C. Mark Sanderson's work [22] shows that 20%-30% extent of error rate in location names disambiguation was enough to worsen the performance of the information retrieval methods. Due to those ambiguities in Web pages, previous research failed to reach a satisfied performance in focused locations extraction.

On the other side, it is hard to resolve the GEO/GEO and GEO/NON-GEO ambiguities as well as to determine the focused locations of Web pages through the widely-studied named entity recognition (NER) approaches. Current NER tools in Web area aim at annotating named entities including place names from Web pages. However, although some of the GEO/NON-GEO ambiguities can be removed by NER tools, the GEO/GEO disambiguation is still a problem. Furthermore, NER tools have no consideration on the extraction of the focused locations of Web pages. Basically, the NER tools are able to extract place names from Web pages, which can be further processed to resolve the GEO/GEO ambiguities as well as the GEO/NON-GEO ones. Thus, in this paper we will not concentrate on the NER approaches but on the following disambiguation and focused locations determination. Those works differ a lot from traditional NER approaches.

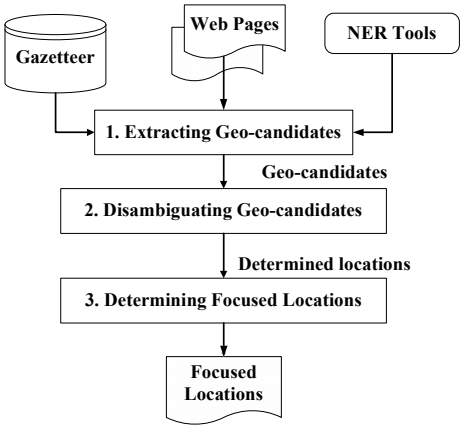


Fig. 1. The general process to extract focused locations from Web pages

Figure 1 shows the general process to extract focused locations from Web pages, in which we first extract geo-candidates based on Gazetteer and NER (named entity recognition) techniques. After this procedure, we get a set of geo-candidates. In this set, the relative order of candidates is the same as that in the text. Here, geo-candidates are just possible place names, e.g., “Washington”. Then, we run the

disambiguation procedure to assign a location for each GEO/GEO ambiguous geo-candidate and remove GEO/NON-GEO ambiguous geo-candidates. Location means a concrete geographical place in the world, e.g.: USA/ Washington, D.C. As a geo-candidate may refer to many locations in the world, the GEO/GEO disambiguation will decide which is the exact location that the geo-candidate refers to and, the GEO/NON-GEO disambiguation is going to determine whether it is a location or not. Finally, we present an effective algorithm to determine focused locations among the resolved locations.

The main contributions of the paper can be summarized as follows:

(1) We propose the *GeoRank* algorithm to resolve the GEO/GEO ambiguity and a heuristic approach to remove the GEO/NON-GEO ambiguity (Section 3). Particularly, the *GeoRank* algorithm uses a similar way as *PageRank* but focused on the determination of the exact location associated with a specific geo-candidate. And the experimental results demonstrate that *GeoRank* outperforms previous methods.

(2) We present an effective algorithm to determine focused locations for Web pages (Section 4), which uses dynamic parameters when computing other locations' contribution to a given location. Compared with the state-of-the-art algorithms with static parameters, our algorithm is more reasonable in computing the importance of locations and has better performance.

(3) We carry out experiments based on real datasets to evaluate the performance of our disambiguation algorithm as well as the algorithm to determine focused locations .

2 Related Work

Disambiguation is usually implemented by using some information in the text such as zip code, phone number and so on. Volz et al. [24] proposed a two-step method, which first used context information to narrow candidates and then ranked the left candidates primarily based on weights according to concepts. Rauch et al. [28] proposed a confidence-based approach. Silva et al. [18] used some classification rules and ontology-based classification such as feature type to disambiguate and with the help of relationships among different geographical concepts, then they used a variation of *PageRank* algorithm to get the focused locations. Place name patterns were studied in SASEIC [12], in which they first examine possible patterns in the Web page, and with the help of these patterns and hierarchical structure of places they get focus of the page. Ding et al. [13] used hyper-links to help decide the page focus. Markowetz et al. [17] and Sobhana et al. [23] made use of the best one of the biggest town first methods and co-occurrence models to remove geographical ambiguity. Andogah et al. [3] proposed a totally different way, with the help of geo-candidate frequency, place type and other features In MyMoSe [25], a K-partite graph for disambiguation was proposed, which used a score-based approach to determine focused locations. There are also other works that employed heuristics in disambiguation [15, 16].

There are also a lot of related works in locations detection [10, 20, 26, 27]. Web-a-where is a four-step heuristics algorithm to determine focused locations for Web

pages [10], in which all names were assigned a location with a confidence score. Based on those confidence scores, as well as other information such as frequency, location relationships and so on, the focused locations of a Web page are extracted. However, Web-a-where adopts fixed parameters and thresholds, which are not suitable for different kinds of Web pages. The evidence-based method is an effective algorithm for geo-candidates disambiguation [26], which makes use of metric relation, topological relation and typological relation between an ambiguous geo-candidate and other co-occurring geo-candidates in the context. Those co-occurring candidates are regarded as the evidences of a geo-candidate, which are fused by the Dempster-Shafer (D-S) theory. However, both of [10] and [26] did not consider the changing confidence that a geo-candidate impacts on other ones, which will lead to bad performance of disambiguation. As shown in our experimental results, the evidence-based method has a comparable performance with Web-a-where in resolving place names ambiguity.

3 Geo-Candidates Disambiguation

3.1 The *GeoRank* Algorithm for Resolving the GEO/GEO Ambiguity

3.1.1 Basic Idea

As Fig.1 shows, we have a set of geo-candidates at present before the disambiguation procedure. We first assume that all geo-candidates are associated with the locations in the Web page. Basically, we assume there are n geo-candidates and totally N locations that n geo-candidates can have in a Web page, the GEO/GEO disambiguation problem can be formalized as follows: *Given a specific geo-candidate G, determining the most appropriate location among its possible locations.*

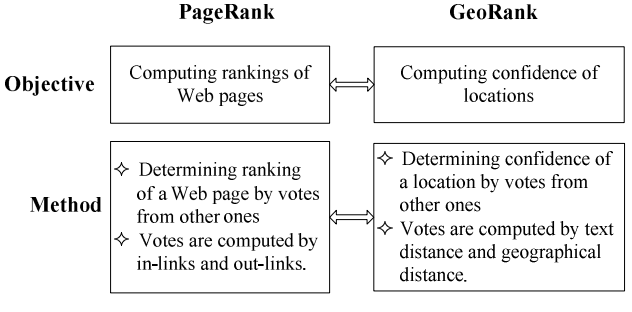


Fig. 2. PageRank vs. GeoRank

We use a general idea similar to *PageRank* to resolve the GEO/GEO ambiguity, which is named *GeoRank*. The *PageRank* algorithm introduced an iterated voting process to determine the ranking of a Web page. We also regard the GEO/GEO disambiguation in a Web page as a voting process. Figure 2 shows the similar problem definition between *PageRank* and our *GeoRank* algorithm. Specially, in

GeoRank, nodes are the locations corresponding to geo-candidates and the linkages are the evidence contributed by the locations each other. The higher score one location gets, the higher confidence it is the right location that the geo-candidate refers to.

In detail, as a geo-candidate can give more evidence to the one near to it in a Web page (text distance) and a location can give more evidence to the one near to it in the geographic context (geographical distance), we first construct a matrix involving all locations, whose values are scores of each location of each geo-candidate voted by other ones that belong to different geo-candidates.

3.1.2 Vote Computation

For simplification, all the possible locations associated with a certain geo-candidate have totally one vote that can be endowed to locations of other geo-candidates. A vote does not mean 1 in actual, it has something to do with the total number of geo-candidates, which we will discuss in section 3.1.3. Initially, all the locations of a geo-candidate have the same percentage of vote, namely 1/ *size* (*L*), L is the set of locations corresponding to this geo-candidate. For example, if a geo-candidate G has 10 possible locations; *size* (*L*) is 10, so their initial percentage of vote is 1/10.

Our *GeoRank* algorithm uses text distance and geographical distance to compute the percentage of vote that is contributed by locations of other geo-candidates to a certain location of a geo-candidate. In particular, the text distance is defined between two geo-candidates, while the geographical distance is defined between two locations of two different geo-candidates. The smaller those distances are, the more evidence they will give to each other.

Differing from traditional text distance that refers to the count of characters between two words, we use the term *relative text distance* to define the text distance between geo-candidates.

Definition 3.1: Relative Text Distance (RTD). Given two geo-candidates G_i *and* G_j in a Web page, their relative text distance is defined as *RTD* (G_i, G_j), which is determined by the count of sentences in the Web page between G_i *and* G_j.

Moreover, if two geo-candidates appear in the same sentence, the *RTD* value will be set to their distance in the geo-candidates list. It's reasonable that two geo-candidates appear in one sentence should have stronger evidence to each other. May be many geo-candidates appear more than once, we define *RTD* (G_i, G_j) as the smallest one among them.

Based on the definition of *RTD*, we can define the vote percentage of a geo-candidate to other geo-candidates.

Definition 3.2: Geo-candidate Vote. Given a geo-candidate G_i, and a set of ambiguous geo-candidates (G_1, G_2… G_m), G_i's vote to G_j is defined as *GV* (G_i, G_j):

$$GV(G_i, G_j) = \left(\frac{1}{RTD(G_i, G_j)}\right) / \sum_{k=1,k\neq i}^{m} \frac{1}{RTD(G_i, G_k)} \quad \blacksquare$$

This formula means that a geo-candidate have fixed percentage of vote, which it gives to other geo-candidates according to their RTD value. We make a normalization to assure that all of G_i's vote is spread to others.

For example, if there are three ambiguous geo-candidates G_1, G_2 and G_3 in a Web page, with $RTD(G_1, G_2)=1$ and $RTD(G_1, G_3)=2$, we can get that $GV(G_1, G_2)=2/3$ and $GV(G_1, G_3)=1/3$, which means G_2 will get 2/3 vote from G_1 while G_3 will get 1/3 vote. The definition of geo-candidate vote implies that a certain geo-candidate has more impact on its neighbors in the text, i.e., those geo-candidates with small *RTD* values.

As we want to finally compute the location-to-location vote, we need to divide the geo-candidate vote among the locations associated with the geo-candidate. So we introduce the geographical distance based method to deal with this issue. Since each geo-candidate appears in the gazetteer, which can be formally represented as a taxonomy tree, we can represent a location in the taxonomy tree as a structural sequence. For example, in the example in Fig.3, the location "*Peak Stone*" can be represented as "*USA/Massachuse/Peak Stone*". We then have the following observation that if two locations in the taxonomy tree have the same left prefix they tend to be located very closely. Therefore, we define the inverse geographical distance between two locations as follows.

Definition 3.3: Inverse Geographical Distance. Given two locations, *loc* of geo-candidate G_i and *loc'* of geo-candidate G_j, the geographical distance between *loc* and *loc'* is defined as *GD (loc, loc'),* and inverse geographical distance is defined as *IGD (loc, loc')*, which refers to the maximal count of the same left prefix between *loc* and *loc'*. A larger IGD value means two locations are nearer. ■

Definition 3.4: Location Vote. Given a location *loc* of geo-candidate G_i and an ambiguous geo-candidate G_j, L_j is the set of locations G_j corresponding to, and $loc' \in L_j$, the vote of *loc* to *loc'* is defined as *LV(loc, loc')*:

$$LV(loc, loc') = GV(G_i, G_j) * (IGD(loc, loc') / \sum_{loc'' \in L_j} IGD(loc, loc''))$$ ■

The location vote in Definition 3.4 indicates that the vote of *loc* to G_j is divided among all the possible locations associated with G_j according to their inverse geographical distances from *loc*. We use $GV(G_i, G_j)$ instead of each G_i's location vote, because each location's vote will be reflected when the confidence vector multiply the matrix, which we will discuss in the next section. This formula indicates that locations with larger the IGD value will get more percentage of vote from *loc*. In case that all the inverse geographical distances equal zero, we divide the $GV(G_i, G_j)$ among all the locations of G_j uniformly, i.e., $LV(loc, loc' \in L_j) = GV(G_i, G_j) / size(L_j)$. It indicates that *loc* cannot give any evidence to G_j, and we record this, which will be used in GEO/NON-GEO disambiguation.

3.1.3 The GeoRank Algorithm

GeoRank mainly consists of three stages (as shown in Fig.3):

(1) On the first stage (line 1 to 4), it computes the geo-candidate vote as well as the location vote, based on the relative text distance and inverse geographical distance which are defined in Section 3.1.2.

Algorithm *GeoRank*

Input: the set of geo-candidates $G = \{G_1, G_2 \ldots G_n\}$, the set of location sets $L = \{L_1, L_2, \ldots, L_n\}$, where L_i is the set of all the possible locations associated with G_i.

Output: the set of locations $D = \{D_1, D_2 \ldots D_n\}$, where D_i is the disambiguated location associated with G_i.

Preliminary: n is the count of geo-candidates, and N is the count of all possible locations associated with n geo-candidates.

/ Computing geo-candidate vote and location vote */*

1: **for** each $G_i \in G$ & $G_j \in \{G - G_i\}$ & G_j is ambiguous **{**
2: compute $RTD(G_i, G_j)$ and then $GV(G_i, G_j)$;}
3: **for** each $G_i \in G$ & $G_j \in \{G - G_i\}$ & G_j is ambiguous & $loc \in L_i$ & $loc' \in L_j$ **{**
4: compute $GD(loc, loc')$ and then $LV(loc, loc')$;}

/ Initializing the matrix for all the locations and the confidence vector */*

5: Initializing an $N \times N$ matrix M, with each location occupies one row and one column. The initial state of M is set by the following rule. {
6: **for** each location *loc* and *loc'* {
7: **if** *loc* = l*oc'* **then** $M[i, j] \leftarrow 0$
8: **else** $M[i, j] \leftarrow LV(loc, loc')$;}}
9: $M = (1-\alpha)M + \alpha S$; *// modify M according to Bryan et al.* [29]
10: Constructing the confidence vector $V = (v_1, v_2, \ldots, v_N)$, For each location loc_i, $v_i = 1/(n*count(G_k))$, where G_k is the geo-candidate loc_i is associated with, and $count(G_k)$ refers to the count of locations associated with G_k.

/ Determining the exact location of geo-candidate */*

11: **while** V does not converge **{**
12: $V = M * V$;
13: normalizing V so that $\sum_1^N v_i = 1$; }
14: Normalizing all the locations' vector values of each geo-candidate to make their sum to be 1.
15: **for** each $G_i \in G$ **{**
16: **for** each location $loc \in L_i$ **{**
17: **if** the *loc*'s vector value in $V > \delta$ **then** {
//i.e., δ *is* a predefined threshold
18: $D_i \leftarrow loc$; **exit for**; }
19: $D_i \leftarrow$ *use the server location and default meaning to help decide*;}}
20: **return** D;

End *GeoRank*

Fig. 3. The *GeoRank* Algorithm

(2) On the second stage (line 5 to 10), it initializes the matrix M for all the locations associated with each geo-candidate, as well as the initial confidence vector V. Generally, the vector represents each location's confidence of a geo-candidate. At first, we assume that each location of a geo-candidate has the same confidence. To make it adaptive to *PageRank*, the sum of all elements in V will be 1;

(3) Then on the third stage (line 11 to 20), we update the vector literately by introducing the influence of M into the confidence vector. The iteration process is similar to *PageRank*. According to Bryan et al. [29], the vector V will converge after several iterations, as we modify the matrix as $M = (1-\alpha)M + \alpha S$, S denotes an $N*N$ matrix with all entries $1/N$, M is column-stochastic and irreducible, according to *Perron–Frobenius theorem*, the vector V will finally converge and reach a stable state, which is not influenced by the initial values of the vector. In the experiment we set α as 0.1.

In the algorithm, we use a threshold δ, which is 0.6 in the implementation, to determine whether a location is the most relevant one for the given geo-candidate. A 0.6 threshold means the location has a confidence of 60% to be the location that the geo-candidate indicates. In case that all the locations associated with the given geo-candidate have vector values (confidence) less than the threshold, which implies that no location of the geo-candidate can be determined in the Web page, then we use the server location of the Web page as a filter and then the default sense to determine the real meaning of geo-candidates. We use the one that has the largest population as its default sense.

3.2 The Heuristic Algorithm for Resolving GEO/NON-GEO Ambiguity

Named entity recognition tools usually can remove some types of the GEO/NON-GEO ambiguities in a Web page. In order to get an improved performance, we propose two additional heuristics in the paper to further resolve GEO/NON-GEO ambiguities. Note these rules are based on the *GeoRank* algorithm we discussed in Section 3.1.

Rule 1: When constructing the matrix M (see Fig.3), if locations of a geo-candidate gets score averagely from all locations of other geo-candidates, it is considered not a location. It is reasonable that none of any possible location of any other geo-candidate can give evidence to locations of this geo-candidate; it is possibly not a location.

Rule 2: After removing the GEO/GEO ambiguity, if a non-country location does not have the same country with any other location; it is considered not a location. Here we get the rule from our observation that a Web page is unlikely to mention a non-country location that does not share a same country with any other locations.

4 Determining Focused Locations

In this stage, we calculate the scores of all the locations after disambiguation, and then return the focused ones for the Web page. We consider three aspects when computing the scores of a location, namely the term frequency, position and the contributions from locations geographically contained by the location. An example of the latter aspect is that if there are many states of USA in a Web page, the location USA will receive contributions from those states, as those states are all geographically contained in USA and mentioning states explicitly means mentioning USA implicitly.

As a result, we use an explicit score to represent the term frequency of a location name, and an implicit score for the geographical containment. The score of a location is its explicit score plus its implicit score.

For location D_i, its explicit score, denoted as ES (D_i), is defined as the term frequency of D_i in the Web page.

Then we use the following heuristics to modify ES (D_i):

(1) If D_i follows on the heels of the other location D_j and D_i has some relationship with D_j, suppose D_j is the son or grandson of D_i, then we think the appearance of D_i in the page aims at emphasizing or explaining D_j, so we take 0.5 away from D_i and add it to D_j, i.e., $ES(D_i) = ES(D_i) - 0.5$, $ES(D_j) = ES(D_j) + 0.5$.

(2) If D_i appears in the title of a Web page, then we add half of SUM to D_i to emphasize this appearance, where SUM is the sum of all the ES values, as defined in the formula 4.1.

$$SUM = \sum_{i=1}^{n} ES(D_i) \tag{4.1}$$

For the implicit scores, since many locations appear in one Web page usually have some geographical relationships, we take this feature when computing the implicit score of a location. In particular, we add some contributions from those locations contained by the given location into the score. Suppose a location D_i contains n sub-locations in the gazetteer: S_1, S_2,…, S_n, and the former m sub-locations appear along with D_i in the Web page, then those m sub-locations will contribute to D_i. The implicit score of D_i is defined in the formula 4.2 and 4.3.

$$IS(D_i) = \sum_{k=1}^{m} (ES(S_k) + IS(S_k)) * \frac{m}{n * diff} \tag{4.2}$$

$$diff = \frac{avg(S_1, S_2, ..., S_m)}{\max(S_1, S_2, ..., S_m)} \tag{4.3}$$

Here, $diff$ refers to the score difference among S_1, S_2,…, S_m, The average value of S_1, S_2,…, S_m must be less than or equal to the maximum value of them, so $diff <= 1$. If D_i contains no sub-locations, then $IS(D_i) = 0$.

Based on a Gazetteer, we can build a hierarchy geographical tree for locations. Then we start from the leaf nodes and compute the scores of all locations. Then we sort all locations according to their scores and partition locations into three groups by using a native clustering approach. The first group with highest scores is determined as the focused locations.

The difference between our algorithm and Web-a-where in [10] is that they employ a fix parameter when measuring the implicit score of a location, namely 0.7, while in our algorithm we use a dynamic parameter as $m/ (n*diff)$. m/n means the more sub-locations of a location appear, the more possibly it will be a focused location and $diff$ means the less difference of sub-locations' score, the more possibly it will be a focused location, this means that this Web page does not emphasis any sub-locations. Thus our algorithm is adaptive to the occurrence of locations that are geographically

related with the given location. Our experimental results demonstrate that our method has benefits by using the dynamic parameter.

5 Experiments

5.1 Datasets

We conduct experiments on real datasets to measure the performance of our algorithm in geo-candidate disambiguation and focused locations determination. Two real datasets are used in the experiments, an nj.gov dataset downloaded from http://www.nj.gov/ and a BBC dataset downloaded from http://www.bbc.co.uk/. For the geo-candidate disambiguation experiment, we choose Web-a-where [10] and the evidenced-based method [26] as the competitors of our *GeoRank* algorithm. For focused locations determination, we compare the performance between our approach and Web-a-where [10]. As surveyed in [14], Web-a-where [10] has the best performance in focused location extraction for Web pages compared with other competitor methods. Therefore, it is meaningful to conduct comparison experiment with Web-a-where.

5.2 Pre-processing

5.2.1 Gazetteer Construction

We first construct a gazetteer based on World Gazetteer [8]. Our gazetteer contains 320,707 place names and 56,665 alternate names. We store the following information about a location in Microsoft SQL Server 2008 database: *id, name, population, latitude, longitude* and *upper* (Here *upper* means its parent which is also represented as a taxonomy node).

5.2.2 Geo-Candidates Extraction

For geo-candidates extraction, we employ CCG (Cognitive Computation Group) [1] as the NER tool. After name entity tagging, we get a set of geo-candidates. Then we scan the set and check each element if it or its relatives appears in the gazetteer. The detailed process is as follows (suppose G_1 is a geo-candidate):

(1) Check G_1 if it appears in the gazetteer, if not found, go to (2);

(2) Remove phrase like "City of" or "City" and repeat the checking in the gazetteer. If G_1 is not found in the gazetteer, we delete it from the list.

5.3 Geo-Candidates Disambiguation

In this procedure, we run our algorithm, Web-a-where [10], and the evidence-based method [26] to resolve the ambiguity of geo-candidates. We first remove the unambiguous ones, i.e., those with only one entry in the gazetteer. Then we get 1990 ambiguous geo-candidates for the nj.gov dataset and 2488 for the BBC dataset.

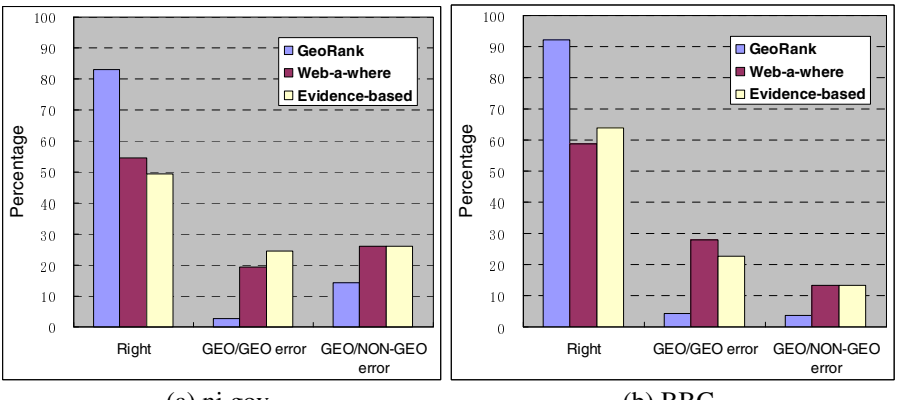
All the ambiguous geo-candidates are resolved by the three algorithms and the outputs are classified into three categories:

(1) *Right*: a geo-candidate is recognized rightly, it is assigned to a right location or it is not a location.

(2) *GEO/GEO error*: a geo-candidate with GEO/GEO ambiguity is not correctly resolved.

(3) *GEO/NON-GEO error*: a geo-candidate with GEO/NON-GEO ambiguity is not correctly resolved.

Figure 4 shows the percentages of the three categories of results for each algorithm (for simplification, *GeoRank* stands for both GEO/GEO and GEO/NON-GEO disambiguation), from which our *GeoRank* algorithm always has the best performance under two datasets and three metrics. In particular, GeoRank has a very low rate for the GEO/GEO error and GEO/NON-GEO error. This is because that *GeoRank* integrates into the disambiguation the confidence as well as its changing of all the locations for a geo-candidate. Another reason is due to its consideration on the text distance among all the geo-candidates appearing in a Web page. Furthermore, the heuristic rules used to reduce the GEO/NON-GEO ambiguity also contribute on the good performance.



(a) nj.gov (b) BBC

Fig. 4. Disambiguation results of *GeoRank*, Web-a-where and the evidenced-based method

5.4 Experiments on Determining Focused Locations

The results of focused locations determination are shown in Fig.5. As Fig.5 shows, we classify the results into four categories, namely *right*, *contain error*, *more or less error*, and *names error*. The definitions on those four metrics are as follows:

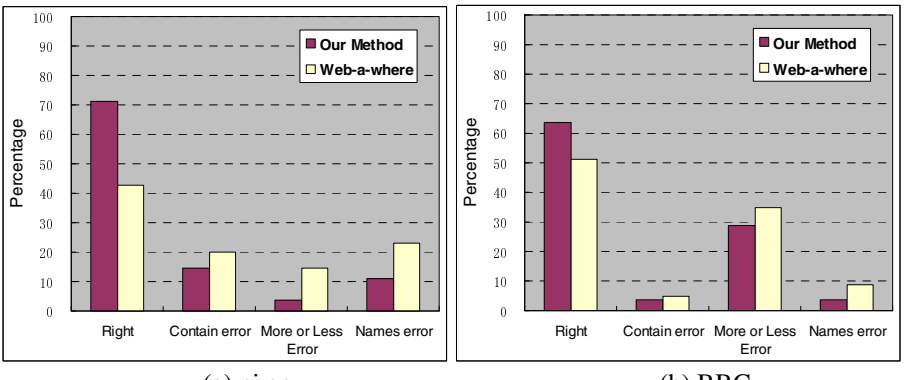
(1) *Right*: the focused locations are determined rightly.

(2) *Contain error*: the determined focused location has a larger or smaller geographical scope than the right one.

(3) *More or less error*: the number of focused locations is more or less than that of right ones.

(4) *Names error*: A wrong focused location is determined. This is mainly because of the former disambiguation error.

Here we only compare our algorithm with Web-a-where [10], as the evidence-based method does not have a procedure for focused locations determination. Figure 6 shows that our algorithm has not only better right rate but also lower error rate for all the three types of errors. According to our experimental results, "names error" is the most frequent error for Web-a-where [10], because of the error in disambiguation phrase. Web-a-where [10] also has a large number of "More or less errors", which are caused by their fixed parameter and thresholds. Differing from Web-a-where [10], we use dynamic parameter in our algorithm, which is demonstrated as a feasible approach to improving the performance of Web-a-where [10]. Another reason for the good performance of our algorithm is that we consider the positions of geo-candidates appearing in text into the computation of location scores.



(a) nj.gov (b) BBC

Fig. 5. Results of focused locations determination

6 Conclusions

In this paper, we concentrated on extracting focused locations from Web pages. In particular, we studied two issues, namely geo-candidates disambiguation and focused location extraction. We presented a new algorithm named *GeoRank* to resolve the GEO/GEO ambiguity and a framework to extract focused locations from Web pages. Experiments on different real datasets show that our approach has better performance than the state-of-the-art algorithms. We plan to make more comparisons by adjusting the parameter in our method and other approaches.

Acknowledgements. This work is supported by the National Science Foundation of China (no. 70803001), the Open Projects Program of National Laboratory of Pattern Recognition (20090029), the Key Laboratory of Advanced Information Science and Network Technology of Beijing (xdxx1005), and the USTC Youth Innovation Foundation.

References

1. Cognitive computation group, http://cogcomp.cs.illinois.edu/page/software (accessed in April 2011)
2. Gate, http://gate.ac.uk/ (accessed in April 2011)
3. Andogah, G., Bouma, G., Nerbonne, J., Koster, E.: Place name Ambiguity Resolution. In: Proc. of LREC, Marrakech Morocco, pp. 4–10 (2008)
4. Geonames, http://www.geonames.org (accessed in April 2011)
5. Washington, http://en.wikipedia.org/wiki/washington (accessed in April 2011)
6. United Nations department of economic and social affairs, http://unstats.un.org/unsd (accessed in April 2011)
7. Usgs geographic names information system (gnis), http://geonames.usgs.gov (accessed in April 2011)
8. World Gazetteer, http://www.world-gazetteer.com (accessed in April 2011)
9. Lingpipe, http://alias-i.com/lingpipe/ (accessed in April 2011)
10. Amitay, E., Har'El, N., Sivan, R., Soffer, A.: Web-a-where: geotagging Web content. In: Proc. of SIGIR, Sheffield, United Kingdom, pp. 273–280 (2004)
11. Anastacio, I., Martins, B., Calado, P.: A comparison of different approaches for assigning geographic scopes to documents. In: Proc. of the INForum 2009 (2009)
12. Chen, M., Lin, X., Zhang, Y., Wang, X., Yu, H.: Assigning geographical focus to documents. In: Proc. of Geoinformatics, Beijing, China, pp. 1–6 (2010)
13. Ding, J., Gravano, L., Shivakumar, N.: Computing geographical scopes of Web resources. In: Proc. of VLDB, Cairo, Egypt, pp. 545–556 (2000)
14. Gyle, A., Plaunt, C.: Gipsy: Automated geographic indexing of text documents. Journal of the American Society of Information Science 45(9), 645–655 (1994)
15. Leidner, J.L.: Toponym resolution in text: Annotation, evaluation and applications of spatial grounding of place names. PhD dissertation, University of Edinburgh (2007)
16. Leidner, J.L.: An evaluation dataset for the toponym resolution task. Computers Environment and Urban Systems 30(4), 400–417 (2006)
17. Markowetz, A., Chen, Y., Suel, T.: Design and implementation of a geographic search engine. In: Proc. of WebDB, Baltimore, Maryland, pp. 19–24 (2005)
18. Silva, M.J., Martins, B.: Adding Geographic Scopes to Web Resources. Computers Environment and Urban Systems 30(4), 378–399 (2006)
19. Martins, B., Silva, M.J.: A Graph-Ranking Algorithm for Geo-Referencing Documents. In: Proc. If ICDM, Houston, Texas, pp. 741–744 (2005)
20. Wang, C., Xie, X., Wang, L., Lu, Y., Ma, W.: Detecting Geographic Locations from Web Resources. In: Proc. of GIR, Bremen, Germany, pp. 17–249
21. Sanderson, M., Kohler, J.: Analyzing geographic queries. In: Proc. of GIR, Sheffield, UK (2004)
22. Sanderson, M.: Retrieving with good sense. Information Retrieval 2(1), 45–65 (2000)
23. Sobhana, N., Barua, A., Das, M., Mitra, P., Ghosh, S.: Co-occurrence Based Place Name Disambiguation and its Application to Retrieval of Geological Text. In: Meghanathan, N., Boumerdassi, S., Chaki, N., Nagamalai, D. (eds.) NeCoM 2010, Part III. CCIS, vol. 90, pp. 543–552. Springer, Heidelberg (2010)
24. Volz, R., Kleb, J., Mueller, W.: Towards ontology-based disambiguation of geographical identifiers. In: Proc. of WWW Workshop on Identity, Identifiers, Identifications (I3), Bandd, Alberta, Canada (2007)

25. Zubizarreta, A., de la Fuente, P., Cantera, J.M., Arias, M.: Extracting geographic context from the Web: georeferencing in mymose. In: Proc. of GIR, pp. 554–561 (2009)
26. Wang, X., Zhang, Y., Chen, M., Lin, X.: An Evidence-based Approach for Toponym Disambiguation. In: Proc. of Geoinformatics 2010, pp. 1–7 (2010)
27. Wang, L., Wang, C., Xie, X., Forman, J., Lu, Y., Ma, W., Li, Y.: Detecting Dominant Locations from Search Queries. In: Proc. of SIGIR, Salvador, Brazil, pp. 424–431 (2005)
28. Rauch, E., Bukatin, M., Baker, K.: A confidence-based framework for disambiguating geographic terms. In: Proc. of HLT-NAACL-GEOREF, pp. 50–54 (2003)
29. Bryan, K., Leise, T.: The $25,000,000,000 Eigenvector: The Linear Algebra Behind Google. Journal SIAM Review 40(3), 569–581 (2006)

Searching Similar Trajectories in Real Time: An Effectiveness and Efficiency Study*

Yuchi Ma, Chunyan Qu, Tingting Liu, Ning Yang**, and Changjie Tang

College of Computer Science, Sichuan University
610065 Chengdu, China
{crystalyc,quchunyaner}@163.com,
molly.liuting@gmail.com,
{yangning,cjtang}@scu.edu.cn

Abstract. Searching similar trajectories in real time has been a challenging task in a large variety of location-aware applications. This paper addresses its two key issues, i.e. evaluating the similarity between two trajectories reasonably and effectively, and providing efficient algorithms to support queries in real time. Firstly, a novel similarity measurement, called Global Temporal Similarity (GTS), is suggested, which is perturbation-free and effective since it takes into account both the evolution of the similarity over time and the spatial movements. Secondly, a new index structure with linear updated time, called Real Time Similar Trajectory Searching-tree (RTSTS-tree), is proposed to support the search of similar trajectories. Besides, to support k Nearest Neighbor (kNN) query of trajectories and Top k Similar Pairs query, two algorithms are proposed based on GTS and RTSTS-tree and are capable of searching similar trajectories by object and by location with the time complexity of $O(n)$ and $O(n^2)$ respectively. Finally, the results of the extensive experiments conducted on real and synthetic data set validate the effectiveness and the efficiency of the proposed similarity measurement, index structure and query algorithms.

Keywords: Trajectory similarity, trajectory index, trajectory query, kNN query, Top k query.

1 Introduction

Nowadays, with the rapid development of location-aware services, tremendous trajectories fill the databases in various applications, such as recommending system for travel routes, pursuing criminal, video monitoring, etc. The success of these applications depends on the effectiveness and efficiency of the queries of similar trajectories, especially the k Nearest Neighbor (kNN) query and the Top k Similar Pairs query.

* Supported by the Central University Fundamental Science Research Foundation of China under Grant No. 2010SCU11053.
** Corresponding author.

L. Wang et al. (Eds.): WAIM 2011 Workshops, LNCS 7142, pp. 90–102, 2012.

Although extensive works aiming at the problem of query over trajectory data have been proposed during the past few years, there are still two key issues far from their mature solutions. The first issue is how to measure the similarity between two trajectories. The existing works often lack for the consideration of the similarity evolving over time more or less and hence may lead to somewhat time-warped results. For instance, compare the best friend in 2003 without any communication after that with the best friend now. Who is the best friend indeed? Similarly, with the time goes by, may the two trajectories become far from each other, and the similarity also should be recalculated. The second issue is how to support the real-time query which requires the underlying index structures to be capable of timely being updated and the query algorithms to capture the time-evolving trajectories.

Aiming at the above two key issues, we first suggest a novel similarity measurement, referred to as Global Temporal Similarity (shortened as GTS) in this paper, which ensures the consistency between time and space and can evolves over time. GTS takes both the evolution of the similarity over time and the spatial movements into consideration, in which the global similarity of two trajectories is evaluated based on a sequence of local temporal similarity measures of them. The main superiority of GTS is perturbation-free. For example, given three trajectories A, B and C, the local similarity between B and A is occasionally far less than that between C and A (i.e. perturbation), but most of the time B is more close to A than C is, so intuitively B is the nearer trajectory of A in the whole time.

To support the real-time query, we then propose a new index structure, referred to as Real Time Similar Trajectory Searching Tree (shortened as RTSTS-tree) in this paper. RTSTS-tree has a good organization that combines the spatial and temporal features of the trajectories, and can be updated in linear time. Two query algorithms with capability of searching similar trajectories by object and by location are further proposed based on GTS and RTSTS-tree. Meanwhile, the proposed algorithms have the time complexity of $O(n)$ and $O(n^2)$ respectively, and therefore can efficiently support real-time query which is validated by the results of the extensive experiments.

In short, the contributions of this paper can be summarized as follows:

1) A new measurement of the similarity between two trajectories, GTS, is suggested in this paper, which takes into account both the similarity's evolution and the spatial movement.
2) A new trajectory index structure, called RTSTS-tree, is proposed with the updating time complexity of $O(n)$.
3) Two query algorithms with time complexity of $O(n)$ and $O(n^2)$ are proposed, which support searching the similar trajectories by object and by location respectively and can well satisfy the requirement of the real-time query.

The rest of this paper is organized as follows. Section 2 discusses the related works. Section 3 describes our measurement of the similarity between two trajectories. Section 4 explains the index structure of the trajectory data. Section 5 presents the algorithms for two kinds of queries, i.e. kNN query and Top k Similar Pairs query.

Section 6 demonstrates the effectiveness and efficiency of the proposed similarity measurement, index structure and algorithms through the experiments. Section 7 presents the conclusions of this paper.

2 Related Work

This section reviews two research directions most relevant to our work: similarity measurement and index structure of trajectories.

Similarity Measurement. Discrete Fourier Transformation (DFT) present by Agrawal et al. [2], evaluates the similarity of two trajectories by Euclidean distances between feature points. As a forerunner, DFT measures the similarity in a fuzzy way, and can be easily computed. DFT, however, does not use all the sequences, which is solved in the following work of Faloutsos et al. in [3]. Cai et al utilized the Chebyshev polynomials and Euclidean distances in [4] for approaching and indexing trajectories. Nevertheless, these methods cannot be applied on the trajectories with different lengths. Berndt and Clifford [5] brought out a method based on the Dynamic Time Warping (DTW) and worked that out. Yet, a shortcoming of DTW is that it would match all the points including noisy ones. This is improved by the Longest Common Sub Sequence (LCSS) adopted in [6]. To make more robust and overcome the LCSS's drawbacks, Chen et al. proposed a new similarity measure based on Edit Distance on Real sequences (EDR) in [7]. Recently, Pelekis et al. [8] proposed a classification of trajectory distance operators, which makes a contribution to the analysis of the trajectory databases. The related works mentioned above, however, often consider the space and time separately when evaluating the similarity. On the contrary, the GTS proposed in this paper conquers this problem in a spatial-temporal consistent manner.

Trajectory Index. The most popular index structures for spatial objects are R-tree and its variations [12-14]. With the development of the moving objects databases, the index structures for trajectory data are proposed, such as the TB-tree [15], 3DR-tree [16] and FNR-tree [11]. These index structures, however, are not suitable for the range query and the similar query of trajectories. To address this problem, Chang et al. proposed TMN-tree in [8], which is based on the MON-tree [9]. The index structure proposed by this paper is based on the TMN-tree and with two significant improvements: the first is the capability of searching by both object andlocation, which the works mentioned above have not; The second is with the input tree instead of the table, we make the updating operation faster, which is suitable for the real time query.

3 Global Temporal Similarity

This section presents our method to measure the similarity between trajectories and gives two examples to illustrate the method. In the following, we express a trajectory A as $A=(a_i)$, $i \in T_s$ where T_s is the time span and a_i is the location of the object at time instance i. Before we give the definition of the global similarity, we first define the local temporal similarity of two trajectories.

Definition 1. *Given two trajectories $A=(a_i)$, $B=(b_j)$, $i, j \in T_s$ where T_s is the time span of A and A is the referring trajectory. For any two time instances i,j at which $a_i=b_j$, the* **Local Temporal Similarity** *(shortened as LTS) at i,j between A and B is defined as:*

$$\boldsymbol{LTS_{i,j} = 2^{T_s-|j-i|-1}}.$$

Definition 1 gives the similarity when *B* is compared with *A*. Note that definition 1 captures the characteristic that the similarity between trajectories declines with the increase of the relative temporal distance (i.e. $|j-i|$). In other words, larger the distance between *i* and *j*, smaller the exponent $T_s-|j-i|-1$, and thereby smaller the $LTS_{i,j}$, which is consistent with people's intuition. Based on the LTS, we can define the global temporal similarity between two trajectories as the following:

Definition 2. *Given two trajectories $A=(a_i)$, $B=(b_j)$, $i, j \in T_s$ where T_s is the time span of A and A is the referring trajectory. The* **Global Temporal Similarity** *(shortend as GTS) between A and B, denoted by GTS(A,B), is defined as:*

$$GTS(A,B) = \frac{1}{M_S} \sum_{i,j,a_i=b_j} LTS_{i,j},$$

where

$$M_S = \sum_{j=1}^{T_S} \sum_{i=1}^{T_S} LTS_{i,j} = \sum_{j=1}^{T_S} \sum_{i=1}^{T_S} 2^{T_S-|j-i|-1} = (3T_S - 4)2^{T_S-1} + 2\,.$$

In definition 2, M_s is the theoretical maximum of the accumulative local temporal similarity over the whole time span. So $GTS(A,B)$ is actually a normalized similarity measurement since taking M_s as the denominator, which makes the measures under different time spans comparable to each others.

The attactive superiority of GTS is its capability to measure the similarity in the scenario with perturbations and the scenario of chase. To explain the scenario with perturbation, suppose there are three trajectories *A*, *B* and *C*, and *B* is more similar with *A* than with *C* in the most time although *A* has a sharp deviation from *B* at some instances which can be regarded as perturbations. The similarity $GTS(B, A)$ will be greater than $GTS(A,C)$ and $GTS(B,C)$ since GTS is based on the total of LTSs and normalized by M_s. In the scenario of chase, one object is chasing another one. For example, considering the trajectory of a camera car and the trajectories of two athletes who are racing in a marathon. The trajectories of the two athletes have the chasing relationship, i.e. in a certain time span, one of the three cases is true: (a) *A* always leads B, (b) *B* always leads *A*, and (c) *A* and *B* take turns to lead. In terms of definition 2, we will find out that $GTS(A,B)$ is greater than $GTS(A,C)$ and $GTS(B,C)$.

4 RTSTS-Tree

This section presents a new index structure, Real Time Similar Trajectory Searching Tree (shortened as RTSTS-tree), which is depicted in Fig.1.

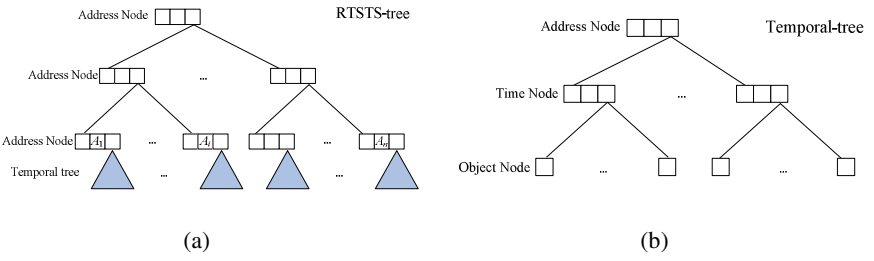


Fig. 1. RTSTS-tree

As shown in Fig.1(a), the RTSTR-tree is made up of two parts, one is the spatial 2D-tree and the other is the temporal-tree.

1) **Spatial 2D-Tree.** The spatial 2D-tree is created to preserve the spatial information in the address nodes. In the spatial 2D-tree, the whole 2D space is organized as a hierarchical structure, in which the area represented by a parent node covers the areas represented by its child nodes. Each node has the form of <*Address*, *Parent*, *ChildPtrList*>, where *Address* is a string that marks a meaningful address of an space area, and *Parent* refers to the parent node, and *ChildPtrList* has different meanings in inner nodes and leaf nodes. In an inner node, *ChildPtrList* saves the pointers to its children that correspond to the subareas, while in a leaf node *ChildPtrList* saves the pointers to the temporal nodes because every leaf node of the spatial 2D-tree is exactly the root of a temporal-tree, and represents an inseparable spatial unit.

2) **Temporal-Tree.** As shown in Fig.1(b), the temporal-tree has three levels including one address node in level 1 (i.e. root), several time nodes in level 2 and several object nodes in level 3. The address node of a temporal-tree is exactly a leaf node of the spatial 2D-tree. Each time node has the form of <*Time*, *Parent*, *ChildPtrList*>, where *Time* marks the sampling time instance, and *Parent* refers to its parent node and *ChildPtrList* points to the object nodes. An object node saves the ID of a moving object that appears in the area represented by the root node and at the time instance marked by the parent of the object node.

Note that the IDs of the moving objects whose trajectories are locally similar with each others are stored in the same temporal tree. This facilitates the computation of the LTS and GTS, and hence makes RTSTS-tree suitable for the search of similar trajectories. The updating process of RTSTS-tree is implemented in the following algorithm.

Algorithm Update_RTSTS (*R*,*D*)

input: *R* the spatial 2D-tree; *D* the new coming trajectory data

output: the updated RTSTS-tree

begin

1. **foreach** leaf node *n* **in** *R*
2. Let $S=\{x: x\in D$ and x falls into the corresponding area of $n\}$;
3. Let *T* be the set of different time instances appeared in *S*;
4. Add new time nodes according to *T* to the temporal tree rooted in *n*;
5. Add new object nodes according to *S* to the new time nodes;
6. **end for;**
7. **end;**

The main work of Update_RTSTS is to update the temporal trees according to the new trajectory data. Update_RTSTS has the time complexity of linear time, which is confirmed by the following proposition 1.

Proposition 1. *Let C_u be the time complexity of Update_RTSTS, then $C_u = O(N)$, where N is the number of points of the new trajectory data.*

Proof. At first, the executing time of statement 4 and statement 5 depends on $|T|$ and $|T|\times N_{mo}$ respectively, where $|T|$ is the number of different time instances in new coming data and N_{mo} is the number of moving objects. On the other hand, the number *m* of times of the loop in statement 1~6 is constant because it equals to the number of the leaf nodes of the spatial 2D-tree. So $C_u = m(|T|+|T|\times N_{mo}) = O(|T|\times N_{mo})$. At the same time, $N = |T|\times N_{mo}$, therefore $C_u = O(N)$. □

5 Query Algorithms

This section proposes two query algorithms based on GTS and RTSTS-tree to support two kinds of the similar trajectories search. One is the *k*NN query and the other is the Top *k* similar pairs query, where the former is the representative of the search by object and the latter is the representative of search by location.

5.1 *k*NN Query

The *k*NN Query is implemented in the following algorithm Search_*k*NN.

Algorithm Search_*k*NN(*Tr*, O_{id}, *R*)
input: *Tr* the given trajectory segment; O_{id} the given object; *R* the RTSTS-tree
output: *k*[] *k* nearest neighbors of *Tr*.
begin
1. T_s= the time span of *Tr*;
2. **foreach** instance *t* in T_s {
3. A_n = the leaf node of the spatial 2D-tree which cover the location of O_{id} at *t*;
4. *GTS*[O_{id}, *] = 0; // Initial the GTS between O_{id} and other objects
5. **foreach** child N_t of A_n // N_t is a time node pointed by An
6. **if** the time of N_t is in T_s **then**
7. **foreach** child N_o of N_t // N_o is an object node pointed by N_t
8. **if** (N_o != O_{id}) **then** // Compute GTS according to definition 2
9. $LTS = 2^{T_s-|N_t-t|-1}$;
10. *GTS*[O_{id}, N_o] = *GTS*[O_{id}, N_o] + *LTS*;
11. **endif**
12. **endfor**
13. **endif**
14. **endfor**
15. **endfor**
16. *k*[] = the N_{object} in the first *k* elements of descending sort on *GTS*[O_{id}, *]

Search_*k*NN outputs the *k* most similar trajectories of the given segment of the trajectory *Tr* of the given object O_{id}. The main work of Search_*k*NN is to compute the GTS between *Tr* and the other trajectories in terms of the definition 2 by traversing the given RTSTS-tree. In line 3, Search_*k*NN locates the address node A_n representing the area where the given object O_{id} appears at a specified time instance *t*. In line 5~12, Search_*k*NN traverses all the object nodes N_o belonging to A_n, and adds the *LTS* between N_o and O_{id} to *GTS*[O_{id}, N_o]. In Line 14, Search_*k*NN gets the *k* nearest neighbors of the trajectory of O_{id} from the descending sort of *GTS*[O_{id}, *]. The following proposition confirms the time complexity of Search_*k*NN is linear with the number of trajectories.

Proposition 2. *Let C_k be the time complexity of Search_kNN, then C_k=O(N), where N is the number of trajectories.*

Proof. Let *d* be the depth of the spatial 2D-tree, and *d* is constant since the space is fixed in advance. Then the executing time of statement 3 depends on *d*. According to the

algorithms Search_*k*NN, the loop from statement 2 to 15 will be executed T_s times, and the loop from statement 5 to 14 will be executed T_s times, and the loop from statement 7 to 12 will be executed N times in the worst case. So the total time of Search_*k*NN is $T_s(d + T_sN) = T_s^2N + T_sd$ which means C_k=$O(N)$. □

5.2 Top *k* Similar Pairs Query

The following algorithm implementes the query of the Top *k* similar pairs query in a given area and time span.

Algorithm Search_Top*k* (A, T_s, R)

input: A the given area; T_s the given time span; R the RTSTS-tree

output: the Top *k* pairs of similar trajectories

begin

1. N_a= the address node covering A;
2. **foreach** leaf node N_c of subtree rooted in A in spatial 2D-tree
3. **foreach** two childs $N_p \neq N_q$ of the temporal tree rooted in N_c
4. **if** N_p.*Parent.Time* and N_q.*Parent.Time* are in T_s **then**
5. Compute $GTS[N_p, N_q]$ according to definition 2;
6. **endif**;
7. **endfor;**
8. **endfor;**
9. Output the Top *k* pairs of similar trajectories from descending sorted $GTS[*,*]$;

Note that in statement 4, it is very convenient to examine whether the time instances of N_p and N_q fall into the given time span T_s because each object node in a RTSTS-tree has saved its parent reference (recall Section 4). The following proposition shows that the time complexity of Search_Top*k* is $O(N^2)$.

Proposition 3. Let C_t *be the time complexity of Search_Topk, then* C_t=$O(N^2)$*, where N is the number of trajectories.*

Proof. The loop times of statement 2 to 8 equals to the number of inseparable areas which is constant. So the executing time of Search_Top*k* depends on the loop times of statement 3 to 7 which is $C_N^2 = O(N^2)$ in the worst case. Hence the total complexity of Search_Top*k* is $O(N^2)$, i.e. C_t=$O(N^2)$. □

6 Experiments

In this section, we conduct the experiments on the real and synthetic data sets to validate the effectiveness and efficiency of the proposed similarity measurement, index structure and query algorithms in this paper. The algorithms are implemented in C# and executed on a Windows 7 platform with Intel Core 2 CPU (2.50GHz) and 2.0GB Memory.

6.1 Data Sets

The real and synthetic data sets are described respectively as follows:

Real Data Set. We use Beijing GPS trajectory data set collected by Microsoft GeoLife Project [17] to validate the effectiveness of the proposed similarity measurement, GTS. We randomly select 620,659 trajectories over the time span of 200×2 seconds. The whole space is divided into 38×37 net grids and each trajectory point is quantized with the coordinates of the grids.

Synthetic Data Set. The synthetic data set consists of 5,000,000 trajectories which are owned by 10,000 different moving objects respectively and in which the maximum time span has 500 time units.

6.2 Effectiveness of GTS

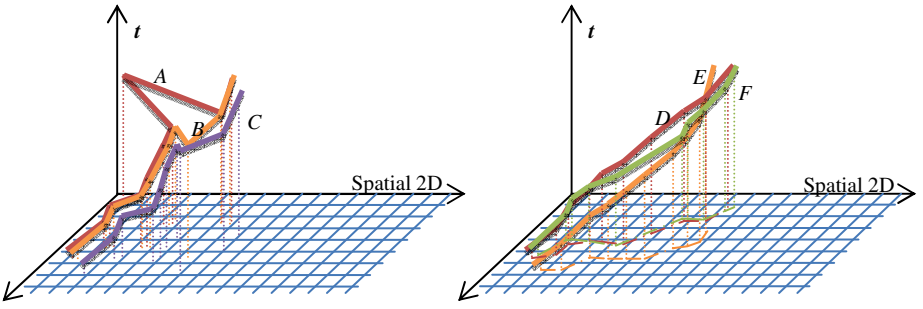
In this subsection we show the main superority of GTS, i.e. it is perturbation-free and can discover the similarity in the chasing scenario.

Table 1. Coordinates of *A*, *B*, *C*

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
A	(2,6)	(3,5)	(3,5)	(5,5)	(5,5)	(5,3)	(5,3)	(1,1)	(8,3)	(8,3)
B	(2,6)	(3,5)	(3,5)	(5,5)	(5,5)	(5,3)	(5,3)	(7,6)	(8,3)	(8,3)
C	(3,7)	(4,6)	(4,6)	(6,6)	(6,6)	(6,4)	(6,4)	(8,7)	(9,4)	(9,4)

Table 2. Coordinates of *D*, *E*, *F*

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
D	(2,6)	(3,6)	(3,5)	(4,5)	(5,5)	(6,5)	(7,4)	(8,3)	(9,3)	(9,2)
E	(2,6)	(3,5)	(3,5)	(4,5)	(6,5)	(8,3)	(8,3)	(9,3)	(9,2)	(9,2)
F	(3,7)	(4,7)	(4,6)	(5,6)	(6,6)	(7,6)	(8,5)	(9,4)	(9,4)	(9,3)



(a) Purterbation (b) Chase

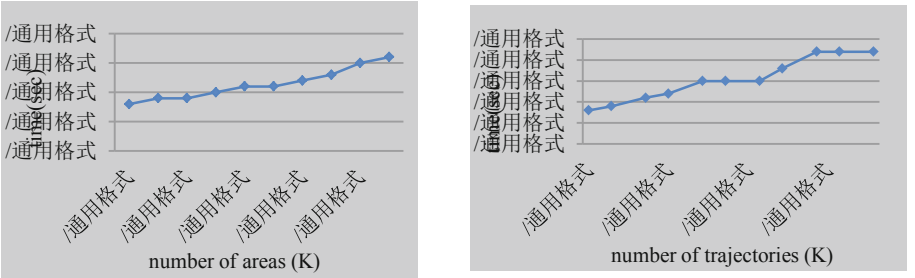
Fig. 2. The scenarios of perturbation and chase

Perturbation Scenario. The trajectory *A*, *B* and *C* with the time span of t_0 to t_9 are depicted in Fig.2(a) and their coordinates are listed in Table 1. It is obvious that *B* is more similar with *A* than with *C* in the most time although *A* has a sharp deviation from *B* at t_7 which can be regarded as a perturbation. Intuitively, the similarity between *B* and *A* is greater than that between *B* and *C* in the whole time span. The traditional similarity measurements based on average distance, however, will derive the nonsense result that B is more similar with *C* than with *A*, because the average suffers the influence of extremas caused by perturbation. On the contrary, using our GTS, we derived $GTS(B, A)=0.5 > GTS(B, C)=0.01$ since GTS is evaluated by summing up the local temporal similarity measures and is normalized by the theoretical maximum of similarity over the whole time span which dilutes the effect of perturbation. So this experiment shows that our GTS is perturbation free.

Chasing Scenario. The trajectory *D*, *E* and *F* with the time span of t_0 to t_9 are depicted in Fig.2(b) and their coordinates are listed in Table 2. As shown in Fig.2(b), *D* and *E* take turns to lead and *F* is concomitant, so the motion pattern of *D*, *E* and *F* is similar to the pattern of athletes and camera car described in Section 3. Here *D* and *E* are the trajectories of two athletes while *C* is the trajectory of camera car. Obviously, *D* and *E* are chasing after each others and the similarity between them should be greater intuitively. Again, the traditional measurements based on average are unable to handle this case since they cannot regonize the local temporal similarity at different time instances. On the contrary, measured by GTS we get the result of $GTS(D, E)= 0.26$, $GTS(D, F)=0.02$ and $GTS(E, F)=0.01$, which is reasonable and conforms to the expectation.

6.3 Updating Time of RTSTS-Tree

To examine the updating time of RTSTS-tree, the algorithm Update_RTSTS is executed on the RTSTS-tree indexing the trajectories in the real data set. The results are shown in Fig.3.



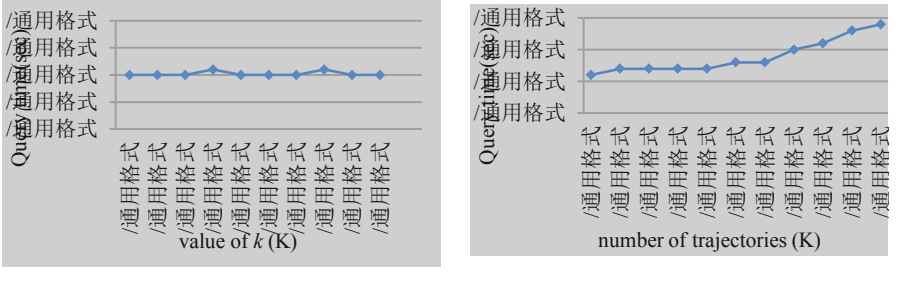
(a) (b)

Fig. 3. Updating time in different situations

Fig.3(a) and (b) indicate that the executing time increases linearly with the increase of the number of inseparable areas and with the increase of the number of the indexed trajectories respectively. This results show it is true that RTSTS-tree can be updated in linear time which is stated in Proposition 1.

6.4 Performance of Search_*k*NN

The algorithm Search_*k*NN is executed on the real data set and the query time is shown in Fig.4. Fig.4(a) shows that the query time almost keeps constant when the value of *k* increasing and Fig.4(b) shows that the query time linearly increases with the increase of the number of trajectories. As the analysis in Subsection 5.1, the reason is that the executing time of Search_*k*NN mainly depends on the scale of the trajectory data not the value of *k*.

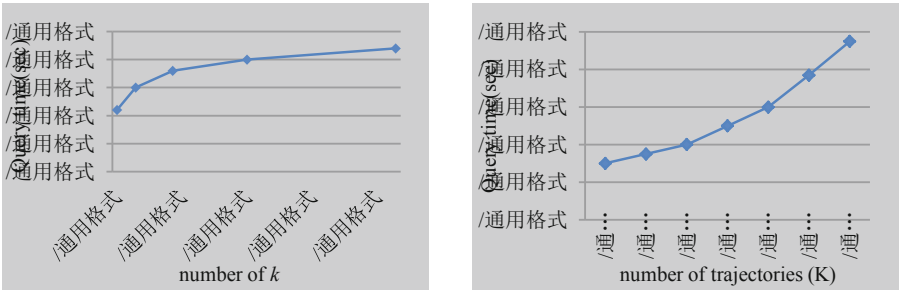


(a) (b)

Fig. 4. Performance of Search_*k*NN

6.5 Performance of Search_Top*k*

The algorithm Search_Top*k* is executed on the real data set and the query time is shown in Fig.5.



(a) (b)

Fig. 5. Performance of Search_Top*k*

As shown in the Fig.5(a) and (b), the curve of query time of Search_Top*k* is linear with the value of *k* and the number of trajectories, although Proposition 3 states that Search_Top*k* has the time complexity of $O(N^2)$. This is because in practice, once the search area is given, Search_Top*k* will compute the result restricted in only one or few leaf nodes of spatial 2D-tree under which the number of object nodes is far smaller than the total scale *N*. So the results show that Search_Top*k* performs well in practice although it is polynomial in the worst case.

7 Conclusion

This paper studies the problem of searching similar trajectories in real time. First, we propose a novel similarity measurement, GTS, to measure the time-evolving similarity between two trajectories, and we show from the theoretical and practical aspects that GTS is able to handle the scenarios with perturbation and chase. Second, we present a new index structure RTSTS-tree with the updating time of $O(n)$ for the fast evaluation of GTS. Based on GTS and RTSTS-tree, we further propose two algorithms for search by object and by location, i.e. the *k*NN query and the Top *k* similar pairs query, with the time complexity of $O(n)$ and $O(n^2)$ respectively. At last, the extensive experiments show that the proposed similarity measure is effective and reasonable and the proposed query algorithms are suitable for the real time queries in practice.

References

1. Frentzos, E., Gratsias, K., Theodoridis, Y.: Index-based Most Similar Trajectory Search. In: 23th IEEE International Conference on Data Engineering, ICDE (2007)
2. Agrawal, R., Faloutsos, C., Swami, A.: Efficient Similarity Search in Sequence Databases. In: Lomet, D.B. (ed.) FODO 1993. LNCS, vol. 730, pp. 69–84. Springer, Heidelberg (1993)
3. Faloutsos, C., Ranganathan, M., Manolopoulos, Y.: Fast subsequence matching in time-series databases. In: Proceedings of SIGMOD 1994, pp. 419–429 (1994)

4. Cai, Y., Ng, R.: Indexing spatio-temporaltrajectories with chebyshev polynomials. In: Proceedings of SIGMOD 2005, pp. 599–610 (2004)
5. Berndt, J., Clifford, J.: Finding patterns in time series: Adynamic programming approach. In: Advances in Knowledge Discovery and Data Mining, pp. 229–248. AAAI/MIT Press, Menlo Park, CA (1996)
6. Bollobas, B., Das, G., Gunopulos, D., Mannila, H.: Time-Series Similarity Problems and Well-Separated Geometric Sets. Nordic Journal of Computing (2001)
7. Chen, L., Özsu Tamer, M., Oria, V.: Robust and Fast Similarity Search for Moving Object Trajectories. In: Proceedings of SIGMOD 2005 (2005)
8. Chang, J., Song, M., Um, J.: TMN-tree - New Trajectory Index Structure for Moving Objects in Spatial Networks. In: 10th IEEE International Conference on Computer and Information Technology, CIT 2010 (2010)
9. Almeida, V., Güting, R.: Indexing the Trajectories of Moving Objects in Networks. Proceedings of GeoInformatica 9(1), 33–60 (2005)
10. Yu, C., Ooi, B., Tan, K., Jagadish, H.: Indexing the Distance: AnEfficient Method to KNN Processing. In: Proceedings of VLDB 2001, pp. 421–430 (September 2001)
11. Frentzos, E.: Indexing Objects Moving on Fixed Networks. In: Hadzilacos, T., Manolopoulos, Y., Roddick, J., Theodoridis, Y. (eds.) SSTD 2003. LNCS, vol. 2750, pp. 289–305. Springer, Heidelberg (2003)
12. Guttman, A.: R-Trees: A Dynamic Index Structure for Spatial Searching. In: Proceedings of ACM SIGMOD, pp. 47–57 (1984)
13. Sellis, T., Roussopoulos, N., Faloutsos, C.: The R^+-Tree: A Dynamic Index for Multi-Dimensional Objects. In: Proceedings of VLDB 1987, pp. 507–518 (1987)
14. Beckmann, N., Kriegel, H., Schneider, R.: The R*-tree: an efficient and robust access method for points and rectangles. In: SIGMOD 1999, pp. 322–331 (1999)
15. Pfoser, D., Jensen, C., Theodoridis, Y.: Novel Approach to theIndexing of Moving Object Trajectories. In: Proceedings of VLDB 2000, pp. 395–406 (2000)
16. Vazirgiannis, M., Theodoridis, Y., Sellis, T.: Spatio-temporal Indexing for Large Multimedia Applications. In: Proceedings of the IEEE Conference on Multimedia Computing and Systems, vol. 6(4), pp. 284–298 (1998)
17. http://research.microsoft.com/en-us/projects/geolife/

Multidimensional Implementation of Stream ADT*

Filip Křižka, Michal Krátký, Radim Bača, and Peter Chovanec

Department of Computer Science
VšB-Technical University of Ostrava, Czech Republic
{filip.krizka,michal.kratky,radim.baca,peter.chovanec}@vsb.cz

Abstract. Holistic approaches are considered as the most robust solution for processing of twig pattern queries requiring no complicated query optimization. Holistic approaches use an abstract data type called a stream which is an ordered set of XML nodes with the same schema node. A straightforward implementation of a stream is a paged array. In this article, we introduce a multidimensional implementation of the stream for path labeling schemes. We also show that this implementation can be extended in such a way that it supports fast searching of nodes with a content. Although many multidimensional data structures have been introduced in recent years, we show that it is necessary to combine two variants of the R-tree (Ordered R-tree and Signature R-tree) for an efficient implementation the stream ADT.

Keywords: Indexing XML data, stream, stream ADT, path labeling scheme, R-tree, R*-tree, Ordered R-tree, Signature R-tree.

1 Introduction

Twig pattern query (TPQ) is considered as a core operation of query languages, such as XPath or XQuery [27] which are de facto standards among XML query languages.

The TPQ processing have been studied extensively and works such as [28,1,26] have outlined basic principles of structural joins. The main disadvantage of the structural join is that the intermediate result can be significantly larger than the result. In works [4,12,5], we can find approaches based on holistic joins. Contrary to the structural join, the intermediate results are comparable with the final result size. In [23], we can find a comparison of different approaches to TPQ processing based on structural joins, holistic joins, and sequence searching. Holistic approaches were considered as the most robust solution requiring no complicated query optimization.

Holistic approaches use an abstract data type called a *stream* which is an ordered set of XML nodes with the same schema node, e.g. tag, tag+level or

* Work is partially supported by Grants of GACR No. GAP202/10/0573 and GA201/09/0990.

L. Wang et al. (Eds.): WAIM 2011 Workshops, LNCS 7142, pp. 103–112, 2012.

labeled path [6]. A cursor pointing to the first XML node is assigned to each required stream at the beginning of the query processing. Given a stream T, let us define the following operations: (1) `head(T)` which returns the node label at the cursor's position, (2) `eof(T)` which returns true if the cursor is at the end of T, and (3) `advance(T)` which moves the cursor to the next node label in T. The most simple implementation of the stream ADT is an inverted list or a simple array.

The input streams can contain many XML nodes. For example, the average length of streams of the XMARK collection[1] with scaling factor 45 is 168,977.2 if the prefix path streaming scheme is utilized [6]. When we consider the page size 2,048 B and approximately 50 labels per a page then a stream includes 3,379.5 pages in average. Sequential reading and comparing of the XML nodes from the stream can take significant amount of time. This time can be reduced especially when the input stream contains a lot of XML nodes which are not a part of the result (unnecessary XML nodes). Holistic algorithms can be extended in order to use additional two stream's operations: (1) fwdToAncOf(T, n) which forwards a stream cursor on the first label which does not precede the XML node n [12], (2) fwdToDescOf(T, n) which forwards a stream cursor on the first label which has a higher document order then the XML node n [9]. If we index the stream using the XB-tree [4] or XR-tree [12] we can skip many unnecessary nodes when these forward methods are invoked and, as a result, speed up the query processing.

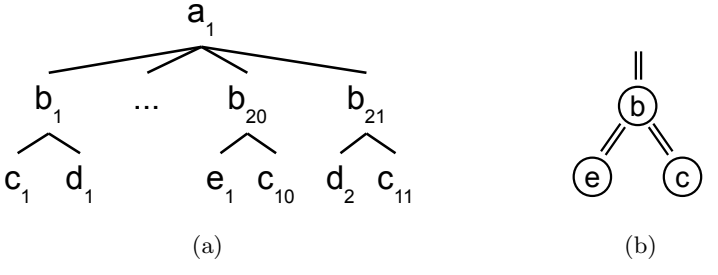


Fig. 1. (a) XML tree (b) TPQ Q1

For example, let us have the XML tree in Figure 1(a) and the TPQ Q1 in the Figure 1(b). We work with three streams T_b, T_e, and T_c (a tag streaming scheme is used) where head(T_b) = b_1, head(T_e) = e_1, and head(T_c) = c_1 at the beginning of query processing. The head(T_b) should be an ancestor of head(T_e) and head(T_c); therefore, an algorithm calls the fwdToAncOf(T_b, e_1) method forwarding the cursor of T_b on b_{20}. In the next step, the algorithm calls the fwdToDescOf(T_c, b_{20}) method since the head(T_c) can not precede its ancestor query node. As a result, the T_c's cursor is forwarded on c_{10} and the cursors form one occurrence of the Q1 in the XML tree. Consequently, we do not have to

[1] http://monetdb.cwi.nl/xml/

access the XML nodes between b_1 and b_{20} and c_1 and c_{10} and all useless nodes are skip in this way.

An XML node is represented by a label which allows us to resolve basic XPath relationships such as parent-child, ancestor-descendant, or preceding-following. There are two types of labeling schemes which assign a label to an XML node: (1) *element labeling scheme* (e.g., containment labeling scheme [28] or Dietz's labeling scheme [8]) and (2) *path labeling scheme* (e.g., Dewey Order [26] or OrdPath [24]). Both above depicted stream indexing techniques (XB-tree and XR-tree) are designed for the element labeling schemes, which makes them unusable when we want to use the holistic algorithms such as TJFast [21] or TreeMatch [22] which utilizes the path labeling scheme.

In this article, we introduce a multidimensional implementation of the stream ADT for path labeling schemes. Multidimensional indices have been utilized to indexing XML data in works [10,15,2,14], however there was no a relation to holistic joins. In this article, we introduce the utilization of multidimensional indices in a relation with holistic joins. We show that the stream index can be extended in such a way that it supports fast searching of nodes with a content which is important for an efficient query processing of XPath queries containing a value condition. Since the R-tree is a well-known multidimensional data structure, in this article, we utilize this data structure and its variants. A comprehensive description of multi-dimensional data structures is given in [25].

The outline of the paper is as follows: In Section 2, we describe why it is appropriate to implement a stream using a multidimensional data structure. In Section 3, R*-tree, Ordered R-tree, and Signature R-tree are briefly described and we explain why it is necessary to combine the Ordered and Signature R-trees for this purpose. In Section 4, we put forward preliminary experimental results. Finally, we outline possible areas of our future work and conclude the paper.

2 Why Multidimensional Implementation?

Let us consider the following node from the XMARK collection labeled by Dewey order: 0,3,126807,4,3. When a labeled path streaming scheme is used [6], this label is included in a stream with id 493. It means, the labeled path related to the node has id 493. This node contains a content; a string with id 7475. Let us suppose the following tuple: (lp_{id}, *label*, $term_{id}$). Consequently, the tuple for the proposed label is: $(493, 0, 3, 126807, 4, 3, 7475)$.

Whereas a range query of the B-tree returns all tuples of a range defined by a prefix, the multidimensional range query returns all tuples of a multidimensional space $\Omega = D_1 \times D_2 \times \ldots D_n$ in a query rectangle defined by two tuples QL and QH [11,25]. In this article, we use a notation of the range query (493,*,...*,475) instead of $QL = (493, min, ..., min, 475) : QH = (493, max, ..., max, 475)$, where min and max are minimal and maximal values of the domain D. An example of the B-tree range query is 493*, it means that all tuples starting by the value 493 are retrieved.

Now, we consider various types of queries and we analyze if we need a multidimensional index or we can use the common B-tree index with the compound key.

1. Query: "Get all labels of the stream 493":
   This query is processed by the following queries in the data structures, we suppose only queries with at least logarithmic complexity of finding the first item, since in the case of a sequence scan in a complete data structure, we rather use an array implementation of the stream:
   - B-tree, Range Query: 493*
   - Multidimensional Index, Range Query: (493,*,...,*)

   This query is not utilized in the case of holistic joins, we use it in the case of structural joins. However, this query can return only the first item and a context record which enables us to continue with the next label during the scan of a stream. We suppose this technique for the following cases as well.
2. Query: "Get all labels with the term id 475 of the stream 493":
   - B-tree: It is not possible to define a query, since the label is not defined, in other words the second part of the query tuple is not defined. It means, the sequential scan in all nodes related to the stream 493 is processed. If we want to scan only nodes for the stream and the term, we can switch $term_{id}$ and *label* of the tuple. Since we need tuples ordered according to the label, it is not possible to make this switch.
   - Multidimensional Index, Range Query: (493,*,...*,475)
3. Query: "Get all labels which do not precede the label 0,3,126807,4,3 of the stream 493"
   This query represents the operation fwdToAncOf() described in the first section. Range Queries:
   - B-tree: Range Query: 493|0|3|126807|4|4*
   - Multidimensional Index, Range Query: $(493, 0, 3, 126807, 4, 4, *)$:(493, max, max, ..., max, $*$) with respect to the length of the labeled path 493.

   We must note that | means a concatenation of two binary (often 32bit) numbers to a longer (64bit) number.
4. Query: "Get all labels of the stream 493 matching the label 0,3,126807,*,..."
   - B-tree, Range Query: 493|0|3|126807*
   - Multidimensional Index, Range Query: (493,0,3,126807,*,...,*)

The multidimensional implementation is appropriate in the case when the complete label is not defined and a content filter is defined. We see that the multidimensional implementation of a stream is more general than the B-tree, however we require the additional properties of a multidimensional data structure used:

- Nodes must be ordered according to labels.
- The data structure must support a special type of range queries: the narrow range query [17]. The narrow range query includes the same values for some identical dimensions in both tuples defining the query rectangle.

3 Multidimensional Index for XML

In this section, we describe the multidimensional data structure fitting the above described properties; it is a combination of Ordered and Signature R-trees. In [19,18], we introduced an R-tree variant to indexing of multidimensional ordered data called Ordered R-tree. In [17], a signature extension of the R-tree for processing narrow range queries has been introduced.

In the R-tree [11], tuples are clustered in tree's pages when *MBRs* (*Minimal Bounding Rectangles*) are built. R-tree supports various types of queries, e.g. point and range queries. A general structure of the R-tree is shown in Figure 2.

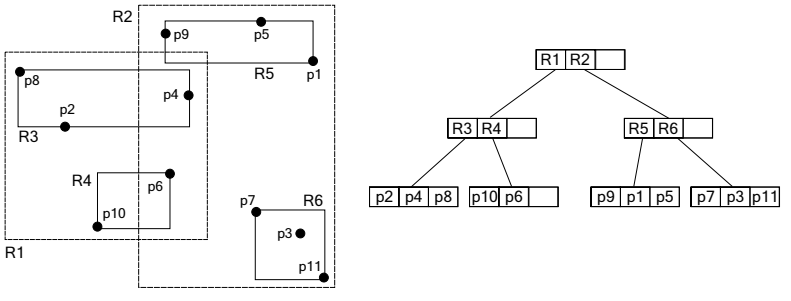


Fig. 2. A general structure of the R-tree

It is a hierarchical data structure representing spatial data by the set of nested n-dimensional *minimum bounding rectangles* (MBR). Each MBR is defined by two tuples QL and QH, where $QL^i \leq QH^i$, $1 \leq i \leq n$. If $\mathcal{N}$ is an inner node, it contains pairs (R_i, P_i), where P_i is a pointer to a child of the node $\mathcal{N}$. If R is the inner node MBR, then the boxes R_i corresponding to the children $\mathcal{N}_i$ of $\mathcal{N}$ are contained in R. Boxes at the same tree level may overlap. If $\mathcal{N}$ is a leaf node, it contains pairs (R_i, O_i), so called *index records*, where R_i contains a spatial object O_i. Each node of the R-tree contains between m and M entries unless it is the root and corresponds to a disk page.

A range query is processed by a depth-first search algorithm. If the query rectangle intersects an MBR then the node related to the MBR is retrieved and searched. In general, there is no order; therefore, tuples of a result set are sorted in the same order in which these tuples were inserted in leaf nodes and new MBRs have been inserted in inner nodes.

There are many variants of the R-tree, the following data structures utilize the R*-tree [3]. In [19,18], Ordered R-tree allowing to index ordered multidimensional tuples has been introduced. A new concept, the *First Tuple* (FT), to each MBR was introduced to support the ordering of tuples in the R-tree. All nodes are then ordered according to their FTs which are utilized during the tree building. In [17], Signature R-tree was introduced for more efficient processing of

narrow range queries. Signature R-tree is a variant of the R-tree including multidimensional signatures for a more efficient filtration of irrelevant tree nodes. In this case, the irrelevant node does not contain any tuple of the query rectangle. The multidimensional signature contains a signature of tuples in one node for each dimension.

Evidently, we need a combination of the Ordered and Signature R-trees for a multidimensional implementation of the Stream ADT. In our preliminary results, we show the performance of this data structure.

4 Preliminary Results

In our experiments[2], we used the XMARK collection[3] of scaling factor 45 with 92 mil. nodes and 547 streams using the prefix path streaming scheme and Dewey Order labeling scheme. Basic characteristics of streams are shown in Table 1. The page size is 2kB for all data structures used. If we consider 40 B per one label in average and the 50% page utilization, the average number of labels in one page is 25. Consequently, the average count of pages of one stream is 6,759.1. Although we can increase the page utilization by a bulk-load algorithm up-to 100% [13], the average count of pages in one stream is high again (approximately 3,379.5 in this case). It is clear that reading and searching in the long streams is a time consuming operation.

Table 1. Stream Statistics of XMARK

	All Labels	**Label Length**	
		1–7	**8–14**
XML file size [GB]	5.0	–	–
#Labels	92,430,540	59,374,127	33,056,414
#Streams	547	148	399
Minimum count of labels in one stream	1	1	1
Maximum count of labels in one stream	2,695,696	2,695,696	2,695,696
Average count of labels in one stream	168,977.2	401,176.5	82,848.2

This collection has been indexed by the R*-tree, Ordered R*-tree, and the combination of Ordered R*-tree and Signature R*-tree introduced in the previous section. In Table 2, we see properties of the created R-trees[4]. We see that the page utilization is by 7% higher in the case of the R*-tree than in the case of the ordered variants. This is caused by inserting of labels exported from the streams. It means, all labels of one stream were inserted before all labels of another stream. In the case of the ordered variants, labels are always inserted into the last leaf

[2] The experiments were executed on an Intel Xeon X5670 2.93Ghz, 12.0 MB L2 cache; 4GB of DDR333; RAID5, SATA, 7200RPM, 1TB; Windows Server 2008 R2.

[3] http://monetdb.cwi.nl/xml/

[4] All data structures have been implemented in C++.

node, therefore the page utlization is 50%. Evidenly, if labels are inserted as an XML document is read or a rebuild algorithm is used, the utilization will be higher. The lower size of the R*-tree is given by the utilization. As we see, the overhead of the Ordered R*-tree compared to the R-tree is only 1.5%; the size of the First Tuple array is 0.05GB. The overhead of the Signature R*-tree is higher: the size of arrays with multidimensional signatures for two lowest levels of the tree is 2×0.59 GB. It means, the overhead is 31.3% compared to the R*-tree and Ordered R*-tree, however the results show that the Signature R*-tree is the most efficient from the query processing time point of view. In this case, we used the following configuration of multidimensional signatures (for detail see [17]): the signature length for the leaf nodes and super-leaf nodes was 128 and 2048 for each dimension, respectively, the number of hashing functions was 3, and one bit was set for each tuple value.

Table 2. R*-trees Statistics

	R*-tree	**Ord. R*-tree**	**Ord-Sig. R*-tree**
Dimension		7	
Paged Size [B]		2,048	
Leaf Node Capacity [items]		63	
#Leaf Nodes	1,624,012	1,855,441	1,855,441
#Inner Nodes	96,350	109,142	109,142
#Tuples		59,374,127	
Avg. #Tuples in Leaf Nodes	36.56	32	32
Page Utilization [%]	57.9	50.9	50.9
Total Tree Size [GB]	3.28	3.36	4.94
R-tree Size [GB]	3.28	3.31	3.75
First Tuple Array [GB]	–	0.05	0.05
Signatures level 0 [GB]	–	–	0.59
Signatures level 1 [GB]	–	–	0.59
Avg. count of pages of one stream	10,973.1	12,536.8	12,536.8

Labels are variable-length tuples with the maximal length for an XML collection. The maximal length is 14 for this collection. We used an approach introduced in paper [16] where the multidimensional forest indexing variable-length tuples has been introduced. In this case, two multidimensional spaces have been created: the first one includes labels of the length 1–7 and the second one includes labels of the length 8–14. Since our queries retrieved only labels up-to the length 7, we used only the first multidimensional space in our experiments (and in Tables 2–4). In Table 1, we see statistics of streams for labels of the length 1–7 as well. This tree includes 59 mil. labels (see also Table 2) and the average count of labels in one stream is higher than in the whole collection (401,176.5 labels).

In our experiments, we tested three groups of queries covering queries over a stream depicted in Section 2. The first group includes queries of the type 1, for example $(292, *, *, *, *, *, *)$ or $(18, *, *, *, *, *, max)$. The second group includes queries of the type 2, for example $(297, *, *, *, *, *, 860099805)$ or $(18, *, *, *, *,$

1718073587, max). The third group includes queries of the type 4, for example $(295, 0, 5, 2462, *, *, 860712262)$ or $(18, 0, 4, 134143, *, 1718073587, max)$. The DAC (Disk Access Cost) [20] for these queries is shown in Table 3. We used c_Q (so called *relevance*) introduced in [17] as a quality ratio of range query processing: $c_Q = N_r/N_p$, where N_r is the number of relevant leaf nodes, it means nodes including tuples in the query rectangle, N_p is the number of all accessed leaf nodes. Evidently, if only relevant nodes are accessed during a range query then $c_Q = 1$.

Table 3. c_Q and DAC for Tested Queries

Query Group	Result Size	R*-tree		Ordered R*-tree		Ord-Sig. R*-tree	
		c_Q	DAC	c_Q	DAC	c_Q	DAC
1	630,366.8	0.85	71,182.5	1.00	20,863.5	1.00	20,863.5
2	7.3	0.03	58,951.8	0.25	11,157.5	0.83	69.0
3	1	1.00	8.5	0.75	7.5	1.00	7.5

We see that for the Query Group 2, c_Q is rather low for the R*-tree. In the case of the Ordered R*-tree, c_Q is higher (0.25), but it is close to 0 for the most of queries. Whereas c_Q is low for some query groups in the case of the R*-tree and Ordered R*-tree, c_Q is 1 or close to 1 for the Ordered-Signature R*-tree. In Table 4, we see the query processing time for tested queries. We must note that we read all pages from the disk and we did not use any method to efficient reading of pages during the range query processing (see [7]), since, in the case of real XML query processing, pages are not read together.

Table 4. Query Processing Time for Tested Queries [s]

Query Group	Result Size	R*-tree	Ordered R*-tree	Ord-Sig. R*-tree
1	630,366.8	16.44	4.99	4.99
2	7.3	8.43	1.79	0.03
3	1	0.00	0.00	0.00

We must note that the result of a range query is not sorted in the case of the R*-tree, therefore labels of the result must be sorted and it means an additional overhead. Another problem of the R*-tree is that leaf nodes include labels of various streams, whereas ordered variants include labels of one stream in the same leaf nodes. Consequently, it is necessary to consider the combination of Ordered and Signature R*-trees to indexing XML data. Moreover, we see that the average count of pages accessed (measured by DAC) << the average count of pages of one stream (12,536.8 in this case) for query groups 2 and 3. In the case of the Query group 1, we read all labels in a stream and DAC is approximately equal to the count of pages in the streams.

5 Conclusion

In this article, we introduced a multidimensional implementation of the stream index for the path labeling schemes related to holistic join approaches. We also show that this stream index can be extended in such a way that it supports fast searching in XML nodes with a content. Our preliminary results showed that it is necessary to combine two variants of the R*-tree: Ordered and Signature R*-trees. In our future work, we should compare the performance of our solution with other stream implementations for real XML queries. We must also solve the storage of variable-length labels since the solution using the multidimensional forest is not optimal.

References

1. Al-Khalifa, S., Jagadish, H.V., Koudas, N.: Structural Joins: A Primitive for Efficient XML Query Pattern Matching. In: Proceedings of the 18th International Conference on Data Engineering, ICDE 2002 (2002)
2. Bauer, M.G., Ramsak, F., Bayer, R.: Multidimensional mapping and indexing of xml. In: BTW. LNI, vol. 26, pp. 305–323. GI (2003)
3. Beckmann, N., Kriegel, H.-P., Schneider, R., Seeger, B.: The R*-tree: An Efficient and Robust Access Method for Points and Rectangles. In: Proceedings of the 9th ACM International Conference on Management of Data (SIGMOD 1990) (1990)
4. Bruno, N., Srivastava, D., Koudas, N.: Holistic Twig Joins: Optimal XML Pattern Matching. In: Proceedings of SIGMOD 2002, pp. 310–321. ACM (2002)
5. Chen, S., Li, H.-G., Tatemura, J., Hsiung, W.-P., Agrawal, D., Candan, K.S.: Twig2stack: bottom-up processing of generalized-tree-pattern queries over xml documents. In: Proceedings of VLDB 2006, pp. 283–294. VLDB Endowment (2006)
6. Chen, T., Lu, J., Ling, T.W.: On Boosting Holism in XML Twig Pattern Matching Using Structural Indexing Techniques. In: Proceedings of SIGMOD 2005, pp. 455–466. ACM Press (2005)
7. Chovanec, P., Krátký, M., Bača, R.: Optimization of Disk Accesses for Multidimensional Range Queries. In: Bringas, P.G., Hameurlain, A., Quirchmayr, G. (eds.) DEXA 2010. LNCS, vol. 6261, pp. 358–367. Springer, Heidelberg (2010)
8. Dietz, P.F.: Maintaining Order in a Linked List. In: Proceedings of 14th Annual ACM Symposium on Theory of Computing (STOC 1982), pp. 122–127 (1982)
9. Grimsmo, N., Bjorklund, T.A., Hetland, M.L.: Fast Optimal Twig Joins. In: Proceedings of VLDB 2010. VLDB Endowment (2010)
10. Grüst, T.: Accelerating XPath Location Steps. In: Proceedings of SIGMOD 2002, pp. 109–120. ACM Press (2002)
11. Guttman, A.: R-Trees: A Dynamic Index Structure for Spatial Searching. In: Proceedings SIGMOD 1984, pp. 47–57. ACM Press (1984)
12. Jiang, H., Lu, H., Wang, W., Ooi, B.: XR-Tree: Indexing XML Data for Efficient Structural Join. In: Proceedings of ICDE 2003. IEEE, India (2003)
13. Kamel, I., Faloutsos, C.: On packing R-trees. In: Proceedings of the Second International Conference on Information and Knowledge Management (CIKM 1993), pp. 490–499. ACM Press (1993)
14. Krátký, M., Bača, R., Snášel, V.: On the Efficient Processing Regular Path Expressions of an Enormous Volume of XML Data. In: Wagner, R., Revell, N., Pernul, G. (eds.) DEXA 2007. LNCS, vol. 4653, pp. 1–12. Springer, Heidelberg (2007)

15. Krátký, M., Pokorný, J., Snášel, V.: Implementation of XPath Axes in the Multidimensional Approach to Indexing XML Data. In: Lindner, W., Fischer, F., Türker, C., Tzitzikas, Y., Vakali, A.I. (eds.) EDBT 2004. LNCS, vol. 3268, pp. 219–229. Springer, Heidelberg (2004)
16. Krátký, M., Skopal, T., Snášel, V.: Multidimensional Term Indexing for Efficient Processing of Complex Queries. Kybernetika Journal 40(3), 381–396 (2004)
17. Krátký, M., Snášel, V., Zezula, P., Pokorný, J.: Efficient Processing of Narrow Range Queries in the R-Tree. In: Proceedings of the 10th International Database Engineering and Applications Symposium (IDEAS 2006), pp. 69–79. IEEE (2006)
18. Křižka, F., Krátký, M.: On the Efficient Indexing of Ordered Multidimensional Tuples. In: 5th International Conference for Internet Technology and Secured Transactions (ICITST 2010). IEEE, London (2010)
19. Křižka, F., Krátký, M., Bača, R.: On Support of Ordering in Multidimensional Data Structures. In: Catania, B., Ivanović, M., Thalheim, B. (eds.) ADBIS 2010. LNCS, vol. 6295, pp. 575–578. Springer, Heidelberg (2010)
20. Lightstone, S.S., Teorey, T.J., Nadeau, T.: Physical Database Design: the Database Professional's Guide. Morgan Kaufmann (2007)
21. Lu, J., Ling, T.W., Chan, C.Y., Chen, T.: From Region Encoding to Extended Dewey: on Efficient Processing of XML Twig Pattern Matching. In: Proceedings of VLDB 2005, pp. 193–204 (2005)
22. Lu, J., Ling, T.W., Bao, Z., Wang, C.: Extended XML Tree Pattern Matching: Theories and Algorithms. IEEE Transactions on Knowledge and Data Engineering (TKDE) 23 (2011)
23. Moro, M.M., Vagena, Z., Tsotras, V.J.: Tree-pattern Queries on a Lightweight XML Processor. In: Proceedings of VLDB 2005, pp. 205–216 (2005)
24. O'Neil, P., O'Neil, E., Pal, S., Cseri, I., Schaller, G., Westbury, N.: ORDPATHs: Insert-friendly XML Node Labels. In: Proceedings of SIGMOD 2004 (2004)
25. Samet, H.: Foundations of Multidimensional and Metric Data Structures. Morgan Kaufmann (2006)
26. Tatarinov, I., et al.: Storing and Querying Ordered XML Using a Relational Database System. In: Proceedings of SIGMOD 2002, pp. 204–215. ACM Press, New York (2002)
27. W3 Consortium. XQuery 1.0: An XML Query Language, W3C Working Draft (November 12, 2003), http://www.w3.org/TR/xquery/
28. Zhang, C., Naughton, J., DeWitt, D., Luo, Q., Lohman, G.: On Supporting Containment Queries in Relational Database Management Systems. In: Proceedings of SIGMOD 2001, pp. 425–436. ACM Press, New York (2001)

Measuring XML Structured-ness with Entropy

Ruiming Tang[1], Huayu Wu[1], and Stéphane Bressan[2]

[1]School of Computing, National University of Singapore
{tangruiming,wuhuayu}@comp.nus.edu.sg
[2]Center for Maritime Studies
steph@nus.edu.sg

Abstract. XML is semi-structured. It can be used to annotate unstructured data, to represent structured data and almost anything in-between. Yet, it is unclear how to formally characterize, yet to quantify, structured-ness of XML. In this paper we propose and evaluate entropy-based metrics for XML structured-ness. The metrics measure the structural uniformity of path and subtrees, respectively. We empirically study the correlation of these metrics with real and synthetic data sets.

1 Introduction

XML is commonly qualified as a semi-structured data model. Indeed, the hierarchical and optional nature of element organization in XML make it a good choice for representing data ranging from annotated unstructured text to relational tables with fixed and prescribed schemata.

Being able to characterize and possibly quantify the absolute or relative "structured-ness" of an XML document or collection can be critical in many applications and at various stages of their life cycle (from design to tuning). For the sake of conciseness let us illustrate the potential benefits of a characterization of XML structured-ness with two example cases among numerous possible ones.

Case 1: There are two types of XML databases, i.e., XML-enabled relational databases and native XML databases. It has been studied that the efficiency of managing and querying XML data using the two types of XML databases highly depend on how structured an XML document is [13]. Generally, the more structured an XML document is, the more efficiently it can be managed and queried with an XML-enabled relational database; while the more irregular a document is, the more advantages emerged for a native XML database. Measuring the structured-ness of an XML document can help to determine which types of database should be used to store the document.

Case 2: Measuring similarity between XML documents is an essential building block for XML comparison, alignment, clustering and classification. However, computing the similarity among a set of XML documents is normally expensive. Using the degree of the structured-ness of each document, we can easily tell one document is far different from another, and thus avoid a lot of unnecessary similarity computations in many applications.

L. Wang et al. (Eds.): WAIM 2011 Workshops, LNCS 7142, pp. 113–123, 2012.

Surprisingly, the early enthusiasm for the promises brought by XML as a candidate standard for data management and interchange has not encouraged attempts to formally and quantitatively define structured-ness. [11] and [2] address the issue in specific contexts for XML collections and DTDs, respectively.

In this paper we understand structured-ness as uniformity of the data representation (schema). Information theoretic entropy suggests itself as a measure of such uniformity, We devise, implement and compare two candidate categories of metrics evaluating uniformity of paths and uniformity of subtrees, respectively, and their variants. We empirically study the correlation of these metrics with the real and synthetic data sets. The structured-ness of the real data sets is manually annotated. The structured-ness of the synthetic data sets is given by construction.

The remainder of this paper is organized as follows: Section 2 presents the background and reviews related work. Section 3 develops our algorithms. Section 4 sets up the experiments to evaluate our algorithms. Section 5 concludes this paper.

2 Background and Related Work

2.1 Background

XML Data Model. An XML document can be modeled as an ordered labeled tree, without considering the ID references. In an XML document tree, each node represents an element, an attribute or a value in the corresponding XML document, and its label is the corresponding tag name, attribute name or value text. Each edge of an XML document tree represents a hierarchical relationship between the element and sub-element, element and attribute, element and value or attribute and value. Since in this paper we are only interested in the structure of an XML document, we ignore the value nodes which appear as leave of the document tree.

Structured-ness of XML Data. XML is a semi-structured data format, which contains both structured components that can be defined by schemas, and the flexibility to freely organize structured components and unstructured components in a hierarchy. The structured-ness of an XML document measures how flexible the document is, i.e., whether the document is regular with many structured components or irregular with a large amount of unstructured tags. The importance of knowing the structured-ness of an XML document is illustrated by the two examples in Section 1.

Shannon Entropy. In the year of 1948, Claude E.Shannon introduced entropy to solve the problem of quantifying information. Shannon entropy is used to quantify the information contained in a random variable; in other words, it is a measure of information we lose if we do not know the value of the random variable. Assume there is a discrete random variable X that can take on possible values $x_1, x_2, ..., x_n$ and the Shannon entropy is: $H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$,

where I(X) is the information content of X, and $p(x_i) = Pr(X = x_i)$ is the probability mass function of X.

There are two views of entropy. The first one is that entropy captures uncertainty in data, i.e., higher entropy indicates more unpredictability. The second view is that entropy captures information content, i.e., higher entropy indicates more information. Thus we can conclude that given two distributions, the one that is closer to the uniform distribution has a higher entropy; if two distributions are both uniform distributions, the one that is more diverse has a higher entropy.

2.2 Related Work

We have not identified any existing work on measuring the structured-ness of XML data. Theoretically, the lot of algorithms measuring similarity between XML documents can be extended for structured-ness measurement. In particular, an XML tree becomes a forest of trees if the root node and the edges between the root node and its children are removed. Then if all trees in the forest are quite similar to each other, the original XML document is considered with a high degree of structured-ness. Hereby, we review the existing work in XML similarity measurement in this section.

[10,5,12] measure the structural similarity between XML documents or XML document/DTD using tree edit distance. [10] aims to partition the XML document collection into several sets based on tree edit distance. [5] detects changes between several versions of an original XML document using tree edit distance. [5] matches XML documents and XML grammars for document classification using tree edit distance. [6,7] measure the similarity between an XML document and an XML query using IR techniques. Traditional information retrieval(TF-IDF) deals with flat textual data. These works add some more information and modify the data structures in order to handle XML data which is in hierarchy. [3,8]uses other techniques to measure similarity between XML documents. For instance, the authors of [3] describe the structure of an XML document as a set of paths, and in [8], they view the structure of an XML document as edges. Moreover, there exist some works which measures similarity using entropy. [11] measures the heterogeneity of XML collections using entropy while [2] proposes an entropy metric to measure the similarity between DTDs.

Our work focuses on measuring the structured-ness of a given XML document as a whole, instead of decomposing it into sub-documents and computing the similarity between these sub-documents.

3 Structured-ness Measurement

We use entropy to measure the structured-ness of an XML document. If the XML document is more structured, it has a low entropy value; otherwise, a high entropy value should be assign to the document. We propose two types of entropy for structured-ness measurement: path-based entropy and subtree-based entropy.

3.1 Path-Based Entropy

The distribution of paths will affect the degree of structured-ness of an XML document. In particular, the more diverse the paths are, the less structured the document is. We define path entropy to measure the diversity of paths. When all the paths[1] in an XML document are identical, i.e., the document is very structured, then the path diversity is of the lowest value, and so does the path entropy. When all the paths in the document are different, the diversity and the path entropy are in the highest values.

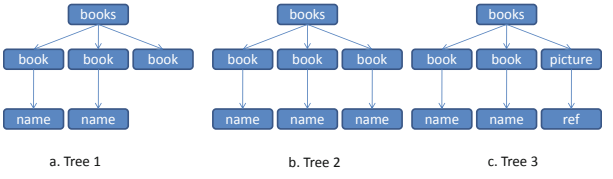


Fig. 1. Samples of three XML documents

The formula is given in Equation 1.

$$Path\_entropy = -\sum_{i=1}^{n} p(X = x_i) \log_2 p(X = x_i) \tag{1}$$

where n represents the number of unique paths. $p(X = x_i) = \frac{n(X=x_i)}{N}$, where N is the number of paths in the document, and $n(X = x_i)$ is the number of paths which are the same as the unique path x_i. For instance, in Tree 1 in Fig. 1.a, there are three paths and only two unique paths. In this example, $p(X = x_1) = \frac{2}{3}$, $p(X = x_2) = \frac{1}{3}$, hence the path entropy of Tree 1 is $\frac{2}{3} * log_2\frac{3}{2} + \frac{1}{3} * log_2\frac{3}{1}$=0.92. Similarly, we can get the path entropy of Tree 2 in Figure 1.b is 0, and the path entropy of Tree 3 in Fig. 1.c has the same value as Tree 1.

3.2 Subtree-Based Entropy

In Fig. 2, we can see some drawbacks of the path entropy. For instance, the document in Fig. 2.a is more structured than the one in Fig. 2.b, so we expect that the path entropy of the document in Fig. 2.a is lower. Unfortunately it is higher.

The reason for the contradictions is that path-based entropy is only suitable for the case that each structured component is a path, instead of a subtree under the document root. However, this case is rare in practice. In this section, we propose to use subtree-based entropy to handle general cases.

[1] This paper focuses on the structure of XML data, so when we mention *path* or *substree* the leaf values are not considered.

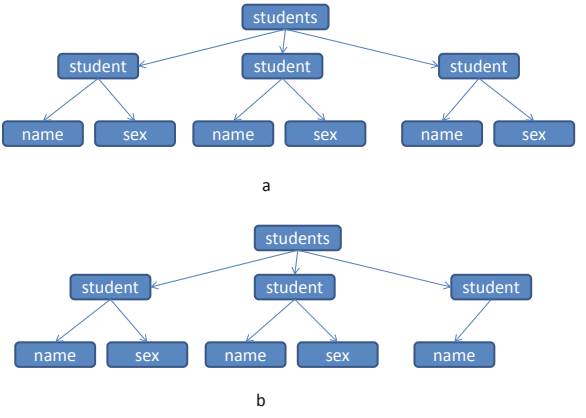


Fig. 2. Samples of two XML documents

Exact Subtrees Entropy. The exact subtrees entropy measures the diversity of subtrees in the document. The basic idea is that the less diverse the subtrees are, the more structured the document is. We need to extract all the subtrees, and then compute the entropy based on their distribution. The formula is given in Equation 2.

$$exact\_subtrees\_entropy = -\sum_{i=1}^{n} p(i) \log_2 p(i) \tag{2}$$

where n represents the number of categories. $p(i) = \frac{N_i}{N}$, where N_i is the number of $subtree_i$ appearing in the document, N is the total number of subtrees. In Fig. 2.a, there are 4 different kinds of subtrees in total while there are 5 in Figure 2.b. For the document in Figure 2.a, $p(1) = 0.3, p(2) = 0.3, p(3) = 0.3, p(4) = 0.1$, and entropy=$3 * 0.3 * log_2 \frac{10}{3} + 0.1 * log_2 \frac{10}{1}$=1.89546. For the document in 2.b, $p(1) = \frac{1}{3}, p(2) = \frac{2}{9}, p(3) = \frac{2}{9}, p(4) = \frac{1}{9}, p(5) = \frac{1}{9}$, and entropy=2.1964. This result reflects the fact that the document in Fig. 2.a is more structured than the one in Fig. 2.b.

Unfortunately, exact subtree entropy also has its drawbacks. This kind of entropy can only tell that these two subtrees are different, but cannot tell how much different they are.

Similar Subtrees Entropy. The basic idea of similar subtrees entropy is that the difference between subtrees is better modeled as a numerical value rather than a Boolean value (yes/no). We use tree edit distance to measure this difference, after which we partition the subtrees into clusters based on their tree edit distances. We get the number of clusters and the size of each cluster. Finally, we compute similar subtrees entropy as the entropy of clusters

Consider an XML document with n substrees. These subtrees are clustered into k clusters, i.e., T=$\{t_1, t_2, ..., t_k\}$. Assume the cluster t_i contains n_i subtrees, and $p_i = \frac{n_i}{n}$ denotes the ratio of the size of t_i over the total size of the document,

in terms of number of subtrees. Then the entropy of T is the entropy of the cluster size distribution. The formula is given in Equation 3.

$$Similar\_subtrees\_entropy = -\sum_{i=1}^{n} p(i) \log_2 p(i) \tag{3}$$

In the similar subtree entropy, there is one important concept, tree edit distance. In our similar subtrees entropy, we choose two different kinds of tree edit distances: the first one is from [14], the second one is from [4]. In [14], for the first time, they introduce a non-exponential algorithm to compute the edit distance between ordered labeled trees, allowing insertion, deletion and updating of internal/leaf nodes. In [4], the authors restrict insertion and deletion to leaf nodes, and allow updating of nodes anywhere (in our version, we replace updating by one deletion and one insertion).

Clustering is a method to partition objects into groups to maximize the similarity within groups and the dissimilarity between groups. We choose the agglomerative clustering algorithm which is a bottom-up hierarchical clustering algorithm in [9].

There are three different metrics for measuring the distance between two clusters A and B: (1) max {d(x,y):$x \in A, y \in B$}; (2) min {d(x,y):$x \in A, y \in B$}; (3) $\frac{1}{|A||B|} \sum_{x \in A} \sum_{y \in B} d(x, y)$. We will use all of them for comparison.

The complete algorithm of similar subtree entropy is given in Algorithm 1.

Algorithm 1: similar subtree entropy Algorithm

Data: *file* : an XML document
Result: The entropy for the document E

Parse file, and get all the subtrees;
$N \leftarrow$ *the number of subtrees*;
for $i = 1; i <= N; i++$ **do**
  for $j = 1; j <= N; j++$ **do**
    compute tree editing distance between $subtree_i$ *and* $subtree_j$*, and store it in the distance matrix*
  end
end
min $\leftarrow$ *minimum value in the distance matrix*;
while *min* <= *threshold* **do**
  merge two closest clusters into one;
  update the distance matrix;
  min $\leftarrow$ *minimum value in the distance matrix*;
end
compute entropy based on distribution of fraction of objects in each cluster

4 Performance Experiment

4.1 Experiment Setup

We generate our synthetic data sets using ToXGene([1]). During synthetic data generation, we use "?" in DTDs to control the structured-ness of each XML document. "?" in the DTDs means that this element either appears once or does not appear. In the same depth, the XML generated by this DTD will be more structured-ness if there are less "?". It will affect the structured-ness of the document more seriously if "?" is in the level closer to the root. Based on the observations above, we can design several DTDs which can guarantee that the XML documents generated by them are ranked from the most structured to least structured. The DTDs are presented in Fig. 3. We can see that Fig. 3.a is the most structured case, while Fig. 3.j is the least structured case. From Fig. 3.a to Figure 3.j, the structured-ness of each document decreases. The "*" beside the element "student" means that the number of students can be changed.

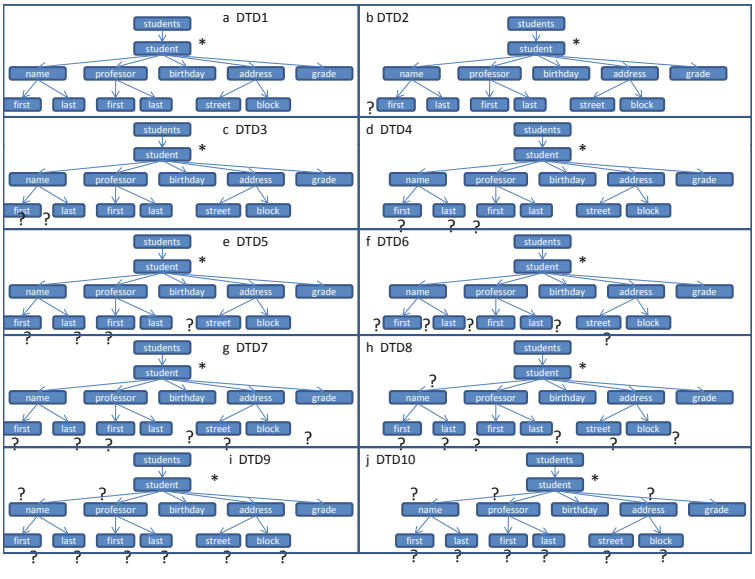
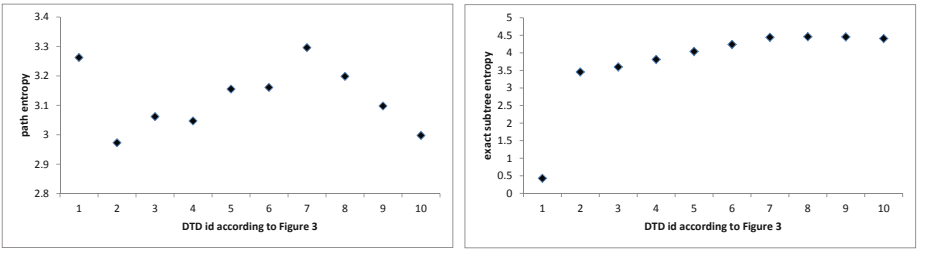


Fig. 3. DTDs used for generating synthetic data

From our manual judgment, the entropy of each group of documents generated by DTDs in Fig. 3 should keep increasing from Fig. 3.a to Fig. 3.j due to the increase of the number of "?" in these DTDs. We calculate all these kinds of entropy when the number of students is 50.

4.2 Experimental Evaluation on Synthetic Data

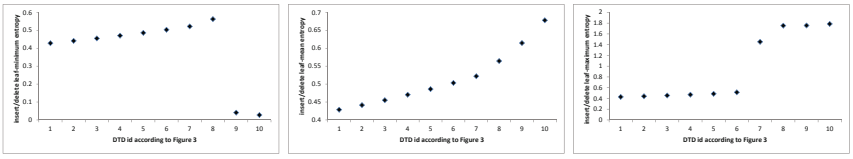
The experimental evaluation results are presented in Fig. 4 and 5. The x axis represents the corresponding DTD ID according to Fig. 3, the y axis is for the average entropy value. Each point in Fig. 4 and Fig. 5 is average entropy value of 1000 documents generated by the corresponding DTD. Fig. 4.a and Fig. 4.b are path entropy and exact subtree entropy when fixing the number of students to 50 and varying DTDs respectively. Fig. 5 is the similar subtree entropy. Fig. 5.a to Fig. 5.c are results for using the first kind of tree edit distance in similar subtree entropy, and Fig. 5.d to Fig. 5.f are results for using the second kind of tree edit distance(these two kinds of tree edit distance are introduced in the section of "similar subtree entropy"). In Fig. 5.a to Fig. 5.c, they are three different metrics we use(minimum value, mean value and maximum value) when we do the clustering(mentioned in the section"similar subtree entropy"), and in Fig. 5.d to Fig. 5.f, the situation is the same.



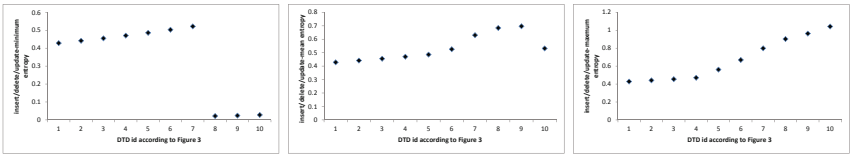
(a) path entropy (b) exact subtree entropy

Fig. 4. Path-based entropy and exact subtree entropy

From Fig. 4.a, we can see that the path entropy does not work well. The key reason is that path entropy defines structured based on paths, but now the synthetic documents are ranked from structured to unstructured based on subtrees. In Fig. 4.b and Figure 5, we can see that the kinds of subtree-based entropy works better, except for the two cases that use "minimum distance" as clustering parameter. In 5.a and 5.d, the entropy value suddenly goes down when there comes the most unstructured documents. The reason is that when we construct synthetic documents, we put more and more "?" in DTDs which causes the nodes of unstructured documents less than structured ones. Thus in the more unstructured case, the tree edit distance between subtrees becomes smaller since the size of subtrees shrinks. To make things worse, the "minimum distance" is more likely to merge different groups compared to "maximum distance" and "mean distance". In this case, the algorithm will generate less clusters than it is supposed to do. However, the exact subtree entropy and other cases of the similar subtree entropy works acceptable since they almost fit to our manual judgement.



(a) insert/delete leaf-minimum entropy (b) insert/delete leaf-mean entropy (c) insert/delete leaf-maximum entropy



(d) insert/delete/update-minimum entropy (e) insert/delete/update-mean entropy (f) insert/delete/update-maximum entropy

Fig. 5. Similar subtree entropy

We also do the experiments varying the number of students from 10 to 100. We find that the results are almost the same as the case of 50. Thus we will not show the results due to the space limitation.

4.3 Experiment Evaluation on Real Data

We get ten different XML documents from INEX and XML repository, whose sizes are from 1K to 334K. INEX: 427.xml(334K), 536.xml(166K), 2008-topics.xml(142K); XML repository: 321gone.xml(24K), ebay.xml(35K), nation.xml(5K), region.xml(1K), supplier.xml(29K), ubid.xml(20K), yahoo.xml(25K). We measure Pearson correlation coefficient within kinds of subtree-based entropy. Table 1 is the Pearson correlation coefficient within subtree based entropy.

As shown in Table 1, the Pearson's correlation coefficient is almost 1 between any two kinds of entropy within the same tree edit distance method(one is insert/delete/update, the other one is insert/delete leaf). When looking into the entropy values, we can see the entropy values of insert/delete/update-minimum, maximum, and mean are almost the same for most of the documents. The same thing happens for entropy of insert/delete leaf. We can conclude that, sometimes when computing entropy for real XML data, it does not matter choosing which parameter to do the clustering(minimum distance, maximum distance, or mean distance).

Table 1. correlation coefficient between subtree-based entropy for real data

entropy x	entropy y	Pearson's correlation
exact subtree	insert/delete/update-minimum	0.217305
exact subtree	insert/delete/update-maximum	0.185191
exact subtree	insert/delete/update-mean	0.192933
exact subtree	insert/delete leaf-minimum	-0.045959
exact subtree	insert/delete leaf-maximum	0.309342
exact subtree	insert/delete leaf-mean	-0.066557
insert/delete/update-minimum	insert/delete/update-maximum	0.998775
insert/delete/update-minimum	insert/delete/update-mean	0.999257
insert/delete/update-minimum	insert/delete leaf-minimum	0.498007
insert/delete/update-minimum	insert/delete leaf-maximum	0.491377
insert/delete/update-minimum	insert/delete leaf-mean	0.487187
insert/delete/update-maximum	insert/delete/update-mean	0.999936
insert/delete/update-maximum	insert/delete leaf-minimum	0.476003
insert/delete/update-maximum	insert/delete leaf-maximum	0.458947
insert/delete/update-maximum	insert/delete leaf-mean	0.465916
insert/delete/update-mean	insert/delete leaf-minimum	0.481393
insert/delete/update-mean	insert/delete leaf-maximum	0.466889
insert/delete/update-mean	insert/delete leaf-mean	0.471148
insert/delete leaf-minimum	insert/delete leaf-maximum	0.929183
insert/delete leaf-minimum	insert/delete leaf-mean	0.999720
insert/delete leaf-maximum	insert/delete leaf-mean	0.922643

5 Conclusion and Future Work

In this paper, we introduce a method measuring XML structured-ness within a single XML document using entropy. To the best of our knowledge, this is the first work on measuring the structured-ness of XML data. We propose two different groups of entropy, i.e., the path-based entropy and the subtree-based entropy. The experiment results show that the subtree-based entropy work better than path entropy for more general XML data.

As we know, computing tree edit distance is a very expensive operation. In our future work, we will try to find another metric to measure the distance between subtrees instead of tree edit distance.

References

1. Barbosa, D., Mendelzon, A.O., Keenleyside, J., Lyons, K.A.: ToXgene: An Extensible Template-Based Data Generator for XML. In: WebDB (2002)
2. Basci, D., Misra, S.: Entropy metric for xml dtd documents. ACM SIGSOFT Software Engineering Notes 33(4) (2008)
3. Buttler, D.: A Short Survey of Document Structure Similarity Algorithms. In: International Conference on Internet Computing, pp. 3–9 (2004)
4. Chawathe, S.S.: Comparing Hierarchical Data in External Memory. In: VLDB (1999)

5. Cobena, G., Abiteboul, S., Marian, A.: Detecting changes in xml documents. In: ICDE, pp. 41–52 (2002)
6. Fuhr, N., Großjohann, K.: XIRQL: An XML Query Language Based on Information Retrieval Concepts. ACM Trans. Inf. Syst. 22(2), 313–356 (2004)
7. Grabs, T., Schek, H.-J.: Generating Vector Spaces On-the-fly for Flexible XML Retrieval. In: XML and Information Retrieval Workshop at SIGIR (2002)
8. Kriegel, H.-P., Schönauer, S.: Similarity Search in Structured Data. In: Kambayashi, Y., Mohania, M., Wöß, W. (eds.) DaWaK 2003. LNCS, vol. 2737, pp. 309–319. Springer, Heidelberg (2003)
9. Kurita, T.: An efficient agglomerative clustering algorithm using a heap. Pattern Recognition 24(3), 205–209 (1991)
10. Nierman, A., Jagadish, H.V.: Evaluating structural similarity in xml documents. In: WebDB, pp. 61–66 (2002)
11. Sanz, I., Mesiti, M., Guerrini, G., Llavori, R.B.: An Entropy-Based Characterization of the Heterogeneity of XML Collections. In: DEXA Workshops (2008)
12. Tekli, J., Chbeir, R., Yétongnon, K.: Structural Similarity Evaluation between XML Documents and Dtds. In: Benatallah, B., Casati, F., Georgakopoulos, D., Bartolini, C., Sadiq, W., Godart, C. (eds.) WISE 2007. LNCS, vol. 4831, pp. 196–211. Springer, Heidelberg (2007)
13. Wu, H., Ling, T.W., Chen, B., Xu, L.: Twigtable: Using Semantics in XML Twig Pattern Query Processing. In: Spaccapietra, S. (ed.) Journal on Data Semantics XV. LNCS, vol. 6720, pp. 102–129. Springer, Heidelberg (2011)
14. Zhang, K., Shasha, D.: Simple fast algorithms for the editing distance between trees and related problems. SIAM J. Comput. 18(6), 1245–1262 (1989)

Similarity Join on XML Based on *k*-Generation Set Distance*

Yue Wang, Hongzhi Wang, Yang Wang, and Hong Gao

The School of Computer Science and Technology, Harbin Institute of Technology
{hitwangyue,magicpaul007}@gmail.com, {wangzh,honggao}@hit.edu.cn

Abstract. Similarity join is applied very widely nowadays since data items representing the same real-world objects may be different due to various conventions. Another reason for similarity join is that the efficiency of traditional methods is really low. Therefore, a method with both high efficiency and high join quality is in need. In the paper, we put forward two new edit operations (reversing and mapping) together with related algorithms concerning similarity join based on the new defined measure. In our method, computing tree edit distance is replaced by computing *k*-generation set distance between trees. The join process is simplified largely by applying the new method. The time complexity of our method is $O(n^2)$, where n is the tree size. We have proved that our method owns some advantages over others. And it can be scaled to large data sets as well.

Keywords: Similarity join, XML, new edit operations, *k*-generation set distance.

1 Introduction

XML documents are widely used in data storage on the web due to the ability to represent data from various sources. Integrating XML documents from autonomous databases has become common practice. One of the most universal methods to evaluate the similarity between different XML documents is computing tree edit distance [2].

We focus on hierarchical data, where certain relationship exists between adjacent levels of data items. Hierarchical data can be described as ordered labeled rooted trees. Then data can be matched based on similarities of their corresponding trees [1]. To date, the fastest algorithm to compute tree edit distance still costs $O(n^3)$ run-time, where n denotes the number of nodes. This illustrates that current operations are limited and can merely be applied to traditional edit distance.

Therefore, more edit operations are essential in practical use. Under this circumstance, we put forward two new edit operations: reversing and mapping. And extended tree edit distance is defined on the new edit operations in our work. In this paper, we

* This research is partially supported by National Science Foundation of China under Grant No. 61003046, No. 60831160525, No. 61111130189. Key Program of the National Natural Science Foundation of China under Grant No. 60933001, National Postdoctoral Foundation of China under Grant No. 20090450126, No. 201003447, Doctoral Fund of Ministry of Education of China under Grant No. 20102302120054.

L. Wang et al. (Eds.): WAIM 2011 Workshops, LNCS 7142, pp. 124–135, 2012.

came up with an idea called *k*-generation set distance algorithm. Trees are transformed into *k*-generation sets from each node and the distance between corresponding *k*-generation sets of two trees is utilized as an approximation of distance between them. This technique is very suitable for hierarchical data structures.

To sum up, the following goals have been achieved in the paper:

—New edit operations are applied in the process of similarity joins. And extended tree edit distance is defined correspondingly.
—We propose the idea of *k*-generation set distance based on the new defined measure. *k*-generation set distance is used to evaluate the distance between trees.
—Experiments based on both real and synthetic data confirm our analytical results. Our method is sensitive to structural changes, which is really indispensable in matching hierarchical data.

The rest of the paper is arranged as follows. In section 2, related work is discussed and analyzed. In section 3, we introduce some preliminary information. We propose the *k*-generation set distance method and detailed information in section 4. Section 5 is about performance experiments. In section 6, the conclusion is displayed together with future works.

2 Related Work

In the area of computing tree edit distance, the performances are really affluent. The first algorithm was created by Tai and it held a run-time of $O(n^6)$. Zhang and Shasha improved the results and gained a run-time of $O(n^4)$. Then Klein further improved the algorithm and finished the matching process in $O(n^3 logn)$ run-time. After that, Demaine contributed to improve the algorithm with a $O(n^3)$ run-time in the worst case.

Other previous works try to transform trees into other data structures to simplify the computation process. In the method of *pq*-gram, trees are extended and then transformed into a series of specific sub-trees called *pq*-grams [1]. Later, improvements have been applied to *pq*-gram and the theory of *pq*-hash came into being [3]. Another transformation was using *g*-string distance as an alternative of distance between two trees [4].

To make the matching process of hierarchical data more flexible and effective, additional edit operations including reversing two nodes and mapping a path to an edge are applied and the notion of extended tree edit distance is raised. The edit distance defined on the new operation collection is called extended tree edit distance. Although the matching process of XML data items has been improved by extended tree edit distance, we can still put forward the idea of *k*-generation set distance to further improve the efficiency and time complexity.

3 Preliminary

In this section, some preliminary information and notations that will be used throughout the paper are introduced.

3.1 New Edit Operations and Extended Tree Edit Distance

Trees are hierarchical data structures, which contain two kinds of information: label information and structure information [4]. Traditional edit operations are too limited for join since they neglect the relationship between adjacent levels. In order to place more emphasis on hierarchical structure, two new edit operations are proposed.

Reverse. Reverse two nodes that are one non-root node and its parent node. That is, these two nodes are exchanged with each other while all the other nodes stay unchanged.

Map. Map a path to an edge. Regard a path with more than two nodes as an edge that consists of only the first and last node of the path.

Extended Tree Edit Distance. Given two trees T_1 and T_2, a series of edit operations are applied to transform T_1 into T_2. The operations include inserting, deleting, relabeling, reversing and mapping. The minimum total cost is extended tree edit distance, denoted by $\boldsymbol{\delta(T_1, T_2)}$.

Example 1. In Figure1, Fig. 1 (a) is the original tree while Fig. 1 (b) and Fig. 1(c) are modified trees after applying the two new edit operations respectively.

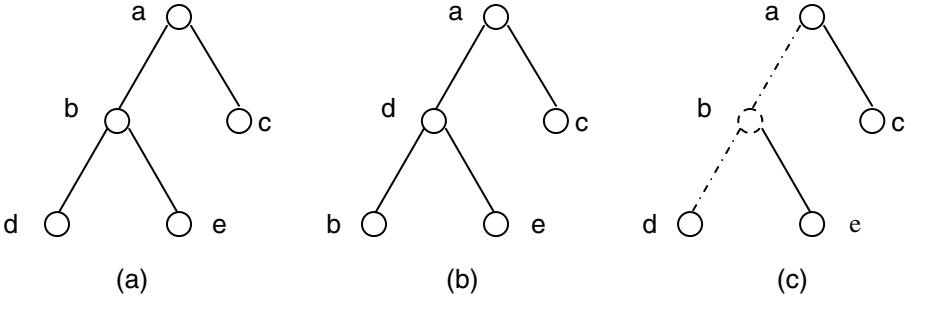
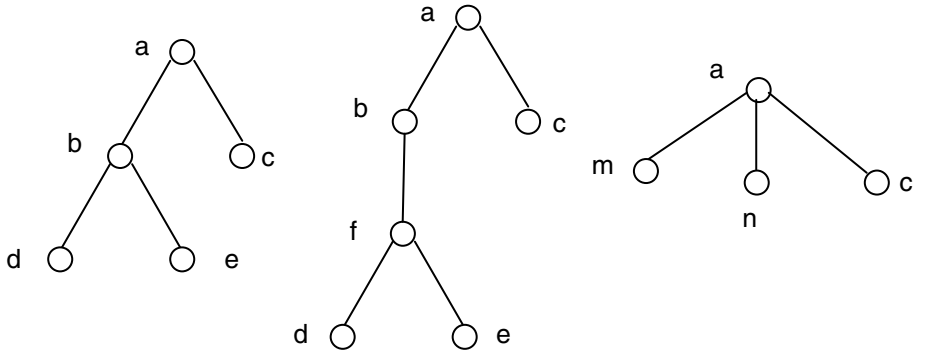


Fig. 1. Two new edit operations (a) Original tree (b) Reverse the two nodes *b* and *d*. (c) Map the path *a-b-d* to an edge of *a-d* (*shown as the dotted line*).

3.2 Similarity Join on XML

Definition 1 (Similarity Join on XML). Given two XML documents F_1 and F_2, the similarity join on XML between F_1 and F_2 is the set $\{(\boldsymbol{Ti, Tj})|(\boldsymbol{Ti, T_j}) \in F_1 \times F_2$ and $\boldsymbol{\delta(T_i, T_j) \leq \theta}\}$ [4], where $\boldsymbol{\delta(T_i, T_j)}$ is computed by pre-defined measure and $\boldsymbol{\theta}$ is a pre-defined threshold.

Example 2. Figure 2 shows an example of similarity join on two XML documents $F_1$and F_2. Let $\theta = 2$. $\delta(T_1, T_2) = 1$ and $\delta(T_1, T_3) = 3$. So the join result is $\{(T_1, T_2)\}$.



(a)T_1 (b) T_2 and T_3

Fig. 2. (a) XML document F_1 (b) XML document F_2

4 Similarity Join Based on *k*-Generation Set

A similarity join returns all pairs of objects (from each set respectively) from two sets of objects based on a similarity function such that their similarity value satisfies a given criterion.

4.1 *k*-Generation Set and *k*-Generation Set Distance

In the paper, trees are divided into sub-trees from each node according to certain rules. Each sub-tree relates to a *k*-generation set. *k*-generation set distance between two trees is computed as a criterion for similarity join.

Definition 2 (*k*-generation sub-tree). Let *V(T)* be the set of the nodes of tree *T*. ∀ $v \in V(T)$, the *k*-generation sub-tree rooted at *v* is defined as follows: Node *v* is the root of the sub-tree. Each time take one node from the path of tree *T* which begins at *v* and ends at a leaf node until the total number reaches *k* (including *v*). If the number of nodes along the path is less than *k*, then some dummy nodes (*) are added to complement the vacancy. The sub-tree rooted at *v* along with *k-1* other nodes is a *k*-generation sub-tree of *T*, denoted by $\boldsymbol{T_k(v)}$.

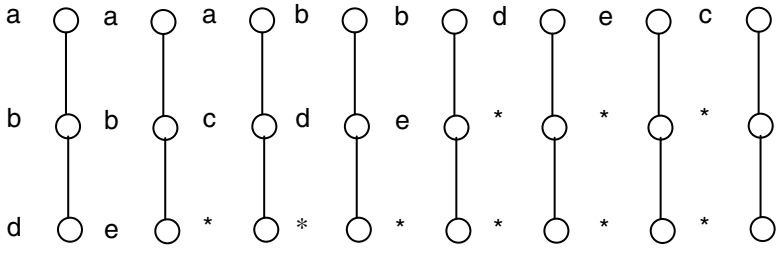


Fig. 3. 3-generation sub-trees of *T*'s nodes, *T* is shown is Fig. 2.(a)

Definition 3 (*k*-generation string). Let $V(T)$ be the set of the nodes of tree T. $\forall v \in V(T)$, the k-generation string of v is the collection of these nodes that are possessed by $T_k(v)$. For each node v, there may be more than one k-generation sub-tree rooted at v. We mark all the k-generation string of v as $S_k(v)$.

Example 3. Figure 2(a) shows a tree T of three levels. The sequence of pre-order traversal of T is a, b, d, e, c. The *3*-generation sub-trees of these nodes are showed below in Figure 3. Accordingly, the *3*-generation string of node a is {(a, b, d), (a, b, e), (a, c, *)}.

Definition 4 (*k*-generation set). Given a tree T, $\overline{V(T)}$ is the set of the non-leaf nodes of T. For each node $v \in \overline{V(T)}$, the set of $S_k(v)$ is called the k-generation set of T, denoted by $\pi_k(T)$.

Non-leaf nodes always have more impact in the join process than leaf nodes [8]. In [8], the notion of influence degree is introduced. Through calculation, we can confirm that the influence degree of non-leaf nodes is higher than that of leaf nodes. Since the impact of internal nodes is more evident, we decide to emphasize more on internal nodes during the search. Consequently, when we define k-generation set of a tree, leaf nodes are not taken into consideration. This method makes the join process less complicated and more effective.

In addition, we'd like to choose the value of k as 3 on account of adapting to the new edit operations. Mapping involves more than two levels of a tree, so k should be larger than 2. However, a much bigger k also brings side effects to the join process. This overlooks the local structure information of the trees, which leads to poor join quality. To compromise between the two conditions, we recommend that k is chosen as 3 generally.

Theorem 1. Given an ordered labeled tree T with root v. k-*2* dummy nodes are added to each leaf nodes of T respectively when computing k-generation set. The number of k-generation sub-trees of node v is no less than that of all the other nodes.

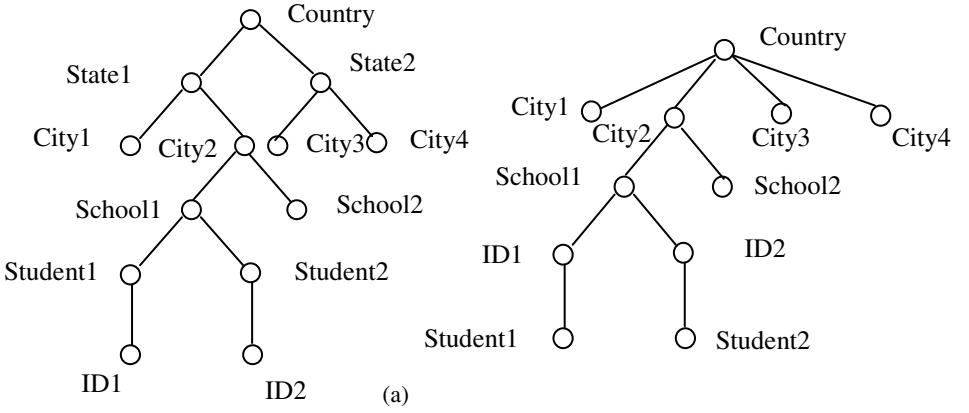
Definition 5 (*k*-string distance). Given two tuples with k elements in each of them, the elements in the tuples are not ordered. Suppose u pairs of elements match with each other in the tuples, then k-string distance between the two tuples is k-u. Particularly, dummy node matches with any element and dummy node matching with dummy node owns priority.

Example 4. Let (a, b, d) and (a, e, b) be two tuples. Since the two nodes with label a and label b match respectively, *3*-string distance between them is 1.

When computing k-string distance, we don't consider the order of the elements in each tuple. At the same time, dummy nodes are added to complement vacancy. These ideas are adopted in order to meet the needs of the two new edit operations. If we neglect the order of the elements, then reversing operation is simulated. We use dummy nodes, which imitates the method of mapping operation since dummy node matches with any element.

Definition 6 (*k*-generation set distance). Let T_1 and T_2 be two labeled trees. For $k > 1$, the *k*-generation set distance $\Delta_k(\boldsymbol{T_1}, \boldsymbol{T_2})$ between the two trees is defined as follows: For each tuple in $\boldsymbol{\pi_k(T_1)}$, find a tuple in $\boldsymbol{\pi_k(T_2)}$ to reach the minimal *k*-string distance. The sum of these *k*-string distances is marked as $Dist(T_1,T_2)$. Then

$$\Delta_k(T_1, T_2) = \frac{\mathrm{Dist}(T_1,T_2)}{|\pi_k(T_1)|+|\pi_k(T_2)|} . \quad (1)$$



$\boldsymbol{\pi_3(T_1)}$:(country,state1,city1), (country,state1,city2), (country,state2,city3), (country,state2,city4), (state1,city1,*,) (state1,city2,school1), (state1,city2,school2), (city2,school1,student1),(city2,school1,student2),(city2,school2,*),(school1,student1,ID1), (school1,student2,ID2), (student1,ID1,*), (student2,ID2,*), (state2,city3,*), (state2,city4,*) $\boldsymbol{\pi_3(T_2)}$:(country,city1,*),(country,city2,school1),(country,city2,school2),(country,city3,*), (country,city4,*), (city2,shool1,ID1), (city2,school1,ID2), (city2,school2,*), (school1,ID1,student1), (school1,ID2,student2), (ID1,student1,*), (ID2,student2,*) (b)

Fig. 4. (a) Two representations of hierarchical data items (b) *3*-generation set of T_1 and T_2

Example 5. Two trees concerning hierarchical data items are shown in Figure 4. They are used to store information about students. The labels of the nodes from adjacent levels have close relationship, as we can see in the figure. If we match $\pi_3(T_1)$ in $\pi_3(T_2)$, then $Dist(T_1,T_2)$=6. $\Delta_3(T_1, T_2)$ =6/(16+12)=0.214. If we match $\pi_3(T_2)$ in $\pi_3(T_1)$, then $Dist(T_2,T_1)$=4. $\Delta_3(T_2, T_1)$=4/(16+12)=0.143.

Theorem 2. Generally speaking, if $\boldsymbol{\delta(T_i, T_j) > \delta(T_m, T_n)}$, then $\boldsymbol{\Delta_k(T_i, T_j) > \Delta_k(T_m, T_n)}$ and vice versa.

Proof. The method of *k*-generation set distance is proposed on the basis of extended tree edit operations. The order of the elements in a set is neglected to adapt to the

operation of reversing. Complementing dummy nodes is applied to adapt to the operation of mapping. Therefore, extended tree edit distance indicated by tree edit operations associates with *k*-generation set distance. Less operations to unify two trees means small extended tree edit distance as well as less structure diversity. Less structure diversity denotes small *k*-generation set distance. That is to say, extended tree edit distance and *k*-generation set distance are consistent to compare two tree structures. One example can be given here. In Fig. 2, $\delta(T_1, T_2) = 1$ and $\delta(T_1, T_3)$=3. $\Delta_3(T_1, T_2)$ =0.182 and $\Delta_3(T_1, T_3)$ =0.25. Obviously, $\delta(T_1, T_2) < \delta(T_1, T_3)$ and $\Delta_3(T_1, T_2) < \Delta_3(T_1, T_3)$.

4.2 Algorithms Concerning the *k*-Generation Set Distance

We show the main algorithms concerning the *k*-generation set distance here. A threshold τ is pre-defined to adapt to certain data items. If the data items for similarity join share many common elements and they look identical in structure, a small τ is suggested. On the contrary, a slightly bigger τ should be selected. In example 5, let τ=0.25. Since $\Delta_3(T_2, T_1)$=0.143<0.25, thus they can be joined with each other.

Algorithm 1. Pre-order Traversal.

```
Input: T, root(T); Output: a sequence of T's nodes de-
noted by P(T)
r= root(T)
if T=∅, then
    return
else
    add r to the sequence P(T)
if r is a non-leaf node then
    for all children c (from left to right) of r do
      P(T)=Pre-order Traversal(sub-tree of T, c)
return P(T)
```

Algorithm 2. Generate *k*-Generation Set.

```
Input: T, k, P(T); Output: π_k(T)
for each node v∈P(T) do
   if v is a non-leaf node then
       add S_k(v) to π_k(T)
return π_k(T)
```

Algorithm 3. Compute *k*-Generation Set Distance.

```
Input: T_1, π_k(T_1), T_2, π_k(T_2), k; Output: Δ_k(T_1,T_2)
if |π_k(T_1)| > |π_k(T_2)| then
   exchange T_1 with T_2
```

```
for each tuple (i) in π_k(T_1) do
     suppose that the k-string distance between tuple i
     in π_k(T_1) and tuple 0 in π_k(T_2) is minimum
   for each tuple (j) in π_k(T_2) do
        if that distance between tuple i in π_k(T_1) and
       tuple j in π_k(T_2) is smaller than previous one then
         use tuple j instead
   record the minimal k-string distance dist
   Dist(T_1,T_2)= Dist(T_1,T_2)+dist
Δ_k(T_1,T_2) = Dist(T_1,T_2)/(|π_k(T_1)| + |π_k(T_2)|)
return Δ_k(T_1,T_2)
```

Algorithm 4. Similarity Join Based on *k*-Generation Set Distance.

```
Input: F_1, F_2, k, τ; Output: join_result
for all trees T_i in F_j do
    P(T_i)=Pre-order Traversal(T_i, root(T_i))
     π_k(T_i)=Generation k-generation set(T_i, k, P(T_i))
     count_ji=|π_k(T_i)| where j denotes the sign of a forest
for all count_ji do
   if |count_1m-count_2n|<(count_1m+count_2n)/k then
      tree_num1←the tree corresponds to π_k(T_m) in F_1
      tree_num2←the tree corresponds to π_k(T_n) in F_2
       Couple=Couple ∪ (tree_num1,tree_num2)
for each tree pairs in Couple do
    Compute k-generation set distance Δ_k(tree_num1,tree_num2)
    if Δ_k(tree_num1,tree_num2) < τ then
       join_result=join_result ∪ (tree_num1,tree_num2)
return join_result
```

After providing the algorithms, a detailed join process is given in Table 1.

Table 1. 3-generation set of T_1, 3-generation set of T_2 and the join process of them

3-generation set of T_1	*3*-generation set of T_2
1.(country,state1,city1)	a.(country,city1,*)
2.(country,state1,city2)	b.(country,city2,school1)
3.(country,state2,city3)	c.(country,city2,school2)
4.(country,state2,city4)	d.(country,city3,*)
5.(state1,city1,*)	e.(country,city4,*)
6.(state1,city2,school1)	f.(city2,shool1,ID1)
7.(state1,city2,school2)	g.(city2,school1,ID2)
8.(city2,school1,student1)	h.(city2,school2,*)
9.(city2,school1,student2)	i.(school1,ID1,student1)
10.(city2,school2,*)	j.(school1,ID2,student2)
11.(school1,student1,ID1)	k.(ID1,student1,*)
12.(school1,student2,ID2)	l.(ID2,student2,*)
13.(student1,ID1,*)	
14.(student2,ID2,*)	
15.(state2,city3,*)	
16.(state2,city4,*)	

T_2	T_1	*dist*
a	1	0
b	6	1
c	7	1
d	3	0
e	4	0
f	8	1
g	9	1
h	10	0
i	11	0
j	12	0
k	13	0
l	14	0

In the table, T_1 and T_2 are from example 5. *3*-generation sets of T_1 and T_2 are given respectively in the first chart. Since $|T_1|>|T_2|$, so *3*-generation set distance between T_1 and T_2 is $\Delta_3(T_2, T_1)$. The second chart in Table 1 shows the relevant tuples and their distances. Consequently, $\Delta_3(T_2, T_1)$ =4/(16+12)=0.143.

4.3 Characteristics of *k*-Generation Set Distance

k-generation set distance emphasizes more on structure, which surpasses other methods when dealing with hierarchical data. Our method is sensitive to local structure information of trees, meanwhile the label information of trees is not overemphasized too much [4].

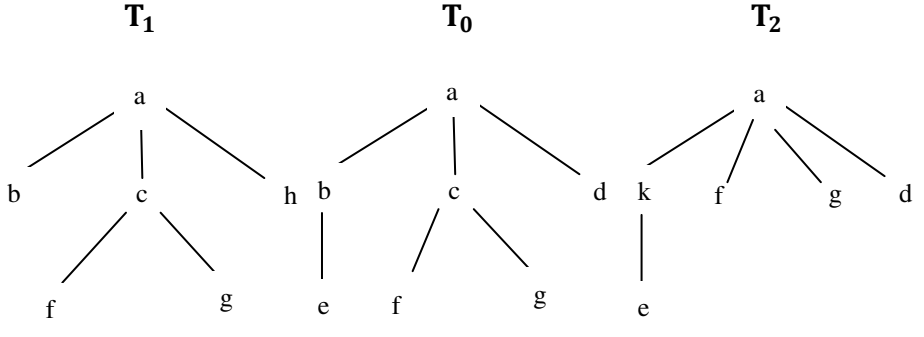


Fig. 5. T_0 is the original tree. T_1 is generated by deleting a leaf node and relabeling a leaf node while T_2 is generated by deleting a non-leaf node and relabeling a non-leaf node.

Example 6. Figure 5 shows a tree T_0 and two modified trees by applying deleting and relabeling operations. $\delta(T_0, T_1) = \delta(T_0, T_2) = 2$. However, T_0 and T_1 seem more similar. $\Delta_3(T_0, T_1)$=0.077 and $\Delta_3(T_0, T_2)$=0.15. Obviously, $\Delta_3(T_0, T_2) > \Delta_3(T_0, T_1)$. It illustrates that T_0 and T_1 share more similarity than T_0 and T_2, which is consistent with the facts.

4.4 Time Complexity Analysis

We summarize the procedure of our method into the following steps. Suppose the two trees to be matched are T_1 and T_2 ($|T_1|=m$ and $|T_2|=n$, $m<n$). (1).Pre-order traversal of T_1 and T_2 respectively; (2).*k*-generation sub-trees of each tree are generated. Then we obtain *k*-generation strings from the sub-trees. The aggregation of these strings (the *k*-generation string of the leaf nodes are excluded) is *k*-generation set; (3).Find the minimal *k*-string distance for each tuple in the *k*-generation set of T_1 recursively. The sum of the distances is recorded. Divide it by a figure we obtain $\Delta_k(T_1, T_2)$; (4).Compare $\Delta_k(T_1, T_2)$ with the pre-defined threshold τ. If $\Delta_k(T_1, T_2) < \tau$, then T_1 matches with T_2; Otherwise, similarity join fails. Finally, output the results.

In (1), pre-order traversal can be finished in $O(n)$ time, where n is the number of nodes. In (2), generating *k*-generation sub-trees can be finished in $O(n)$ time for a tree with size n. In (3), the outer loop is implemented $O(m)$ times while the inner loop is implemented $O(n)$ times. Step (3) will cost $O(mn)$ time in the best case. When the sizes of the two trees are comparable, it costs $O(n^2)$ time. Step (4) takes $O(1)$ running time. Therefore, the overall time complexity is $2(O(m)+O(n))+O(n^2)+ O(1)= O(n^2)$ in the worst case.

5 Performance Experiments

In this section, we will give detailed analysis on performance experiments of our algorithms. Our experiments were performed on a PC with Intel Core Duo 2GHz, 2G main memory and 250G hard disk. The operating system is Ubuntu. The IDE is CodeBlocks.

5.1 Efficiency of Our Method

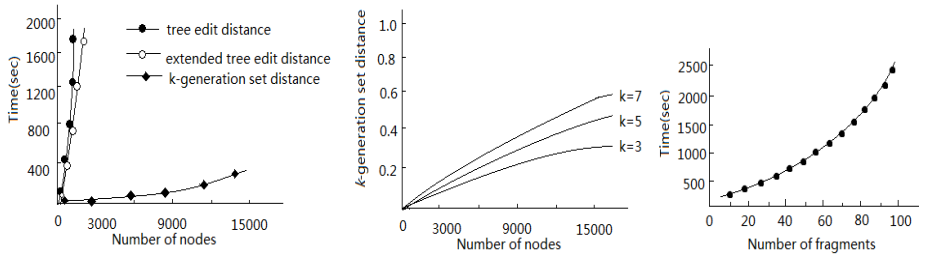
Compared with tree edit distance and extended tree edit distance, our method seems more efficient. We mentioned that our method cost $O(n^2)$ run-time while the fastest algorithms for the other two methods still need $O(n^3)$ run-time.

Experiments are carried out to make a comparison among related distances. In our experiments, an adequate number of trees are randomly created with the number of nodes ranging from 10 to 15000. And all the label information is chosen from a predefined alphabet. The result is shown in Fig. 6.(a). We can see the run-time of tree edit distance and extended tree edit distance grow more rapidly than *k*-generation set distance, which convinces that our method owns high efficiency.

5.2 Influence of Related Parameter

In order to test the influence of parameter k on the join quality of our method, we generated trees as Section 5.1. One discrepancy is that the trees are generated in pairs. The result of the experiment is shown in Fig. 6.(b).

Experiments concerning the number of XML fragments in each document are also tested. In this experiment, each tree contains about 200 to 300 nodes. And the trees are generated with label information chosen from a pre-defined alphabet. The height of the trees is logarithmic. The number of XML fragments in each document ranges from 10 to 100. The run-time is shown in Fig. 6.(c).



(a) (b) (c)

Fig. 6. (a) Comparison among different distances (b) Influence of k on our method (c) Experiments concerning the XML document size

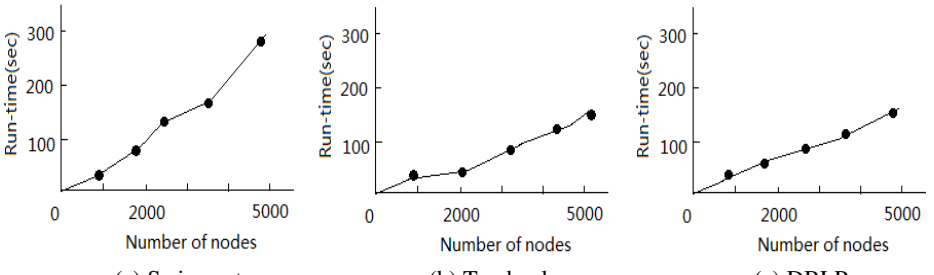


Fig. 7. k-generation set distance applied to large databases

5.3 Scalability and Flexibility

It is demonstrated that our method seems flexible in similarity join since two additional edit operations are added. When dealing with hierarchical data items, the convenience and strength of our algorithms are evident. Besides, our method can be scaled to large data sets. Experiments concerning various XML databases including Treebank, Swissprot and DBLP [4] are performed. The three databases vary in structure. Run-time of computing *3*-generation set distance is shown in Fig. 7.

6 Conclusions

XML documents are often stored as ordered labeled trees with certain roots. When integrating these data items, approximate join is usually applied. In the paper, we put forward two new edit operations: reverse and map. They contribute to more flexibility and effectiveness in the matching process. Thus, similarity join on XML based on the new defined measure is necessary. *k*-generation set distance is created to meet the needs. In our method, pre-order traversal is carried out first. Then *k*-generation subtrees are generated, which leads to the creation of *k*-generation set. Finally, matching between two *k*-generation sets is applied and *k*-generation set distance is acquired. Both theoretical analysis and performance experiments have proved the efficiency, scalability and other properties of our algorithms.

Future work will concern more about improving the join quality further. Besides, the stability of our method will be taken into consideration as well.

References

1. Augsten, N., Bohlen, M., Gamper, J.: Approximate matching of hierarchical data using pq-grams. In: Proc. of the 31st VLDB Conferences, Trondheim, Norway, pp. 301–312 (2005)
2. Bille, P.: A survey on tree edit distance and related problems. Theor. Comput. Sci. 337 (1-3), 217–239 (2005)
3. Li, F., Wang, H., Hao, L., Li, J., Gao, H.: pq-hash: An Efficient Method for Approximate XML Joins. In: Shen, H.T., Pei, J., Özsu, M.T., Zou, L., Lu, J., Ling, T.-W., Yu, G., Zhuang, Y., Shao, J. (eds.) WAIM 2010. LNCS, vol. 6185, pp. 125–134. Springer, Heidelberg (2010)
4. Li, F., Wang, H., Zhang, C., Hao, L., Li, J., Gao, H.: Approximate Joins for XML Using *g*-String. In: Lee, M.L., Yu, J.X., Bellahsène, Z., Unland, R. (eds.) XSym 2010. LNCS, vol. 6309, pp. 3–17. Springer, Heidelberg (2010)
5. Augsten, N., Bohlen, M.H., Gamper, J.: The pq-gram distance between ordered labeled trees. ACM Trans. Database Syst. 35(1) (2010)
6. Tatikonda, S., Parthasarathy, S.: Hashing Tree-Structured Data: Methods and Applications. In: ICDE (2010) (to appear)
7. Dulucq, S., Touzet, H.: Analysis of Tree Edit Distance Algorithms. In: Baeza-Yates, R., Chávez, E., Crochemore, M. (eds.) CPM 2003. LNCS, vol. 2676, pp. 83–95. Springer, Heidelberg (2003)
8. Han, Z., Wang, H., Gao, H., Li, J., Luo, J.: Clustering-Based Approximate Join Method on XML Documents. Journal of Computer Research and Development (suppl.), 81–86 (2009); ISSN:1000-1239/CN 11-1177/TP46
9. Guha, S., Jagadish, H.V., Koudas, N., Srivastava, D., Yu, T.: Approximate XML Joins. ACM SIGMOD (June 4-6, 2002)
10. Guha, S., Jagadish, H.V., Koudas, N., Srivastava, D., Yu, T.: Integrating XML Data Sources Using Approximate Joins. ACM Transactions on Database Systems 31(1), 161–207 (2006)

XML Query Processing Using Views

Caiyun Yao[1], Jiaheng Lu[1], Wei Wang[2], and Xiaofang Zhou[1,3]

[1] School of Information, Renmin University of China
[2] School of Computer Science and Engineering, University of New South Wales
[3] School of Information Technology and Electrical Engineering, University of Queensland
ceylon0614@msn.com, jiahenglu@gmai.com, weiw@cse.unsw.edu.au, zxf@uq.edu.au

Abstract. A fundamental problem in XML query processing is tree pattern query (TPQ) matching which computes all data instances in an XML database that match an input TPQ. More recently, there is growing attention on applying materialized views, which is an established and effective optimization technique in relational database systems, to TPQ matching. We study the query answering using views problem for tree pattern queries (QAV). The QAV problem is traditionally formulated in two ways: (i) find a maximal result, or (ii) find an equivalent result. For the former, there exists an idea of searching for a maximal contained rewriting, by applying some compensation to the result of view. Because not all the answers will be returned, some useful answers may be missing. Motivated by this, we study the latter one—finding an equivalent rewriting of tree pattern queries. We mainly focus on path query.

Keywords: Extended Dewey Encoding, Path Query Matching, XML View.

1 Introduction

A fundamental problem in XML query processing is tree pattern query (TPQ) matching [1] which computes all data instances in an XML database that match an input TPQ. More recently, there is growing attention on applying materialized views, which is an established and effective optimization technique in relational database systems [3], to TPQ matching. We study the query answering using views problem for tree pattern queries (QAV). The QAV problem is traditionally formulated in two ways: (i) find a maximal result, or (ii) find an equivalent result. For the former, there exists an idea of searching for a maximal contained rewriting, which is got by applying some compensation to the result of view [6]. Because not all the answers will be returned, some useful answers may be missing. Motivated by this, we study the latter one --- finding an equivalent rewriting of tree pattern queries. We illustrate the problem next.

1.1 Equivalent Result

Fig. 1(b) shows a materialized view computed by the expression V, "//Report//Course" on some database containing students reports which involve the information of courses the

L. Wang et al. (Eds.): WAIM 2011 Workshops, LNCS 7142, pp. 136–139, 2012.

students took and the score they got if they had pass the final exams. Fig. 1(a) shows one possible database D the V's result might have come from. We encode the nodes for easy reference.

Consider the query Q, "//Compulsory//Course" in the document D in Fig. 1(a), Course (2) and Course (5) meet the demand. Therefore, the result of applying Q on D is Course elements (2, 5). Suppose only the materialized result of V in Fig. 1(b) is available (as a data source). When applying Q on V, if the result got by applying Q on V is the same as the one got by applying Q on D, we call it an *equivalent result*.

<Report>(0)
<Compulsory>(1)
<Course>(2) <Name>(3) Physics</Name>
<Score>(4) 4 </Score></Course>
<Course>(5) <Name>(6) Calculus</Name>
</Course>
<GPA>(7)4</GPA>
</Compulsory>
<Optional>(8)
<Course>(9) <Name>(10) Yoga </Name></Course>
</Optional>
</Report>

(a) Sample XML Document D

<Course>(2) <Name>(3) Physics</Name>
<Score>(4) 4 </Score></Course>
<Course>(5) <Name>(6) Calculus</Name>
</Course>
<Course>(9) <Name>(10) Yoga </Name></Course>

(b) The View V of Sample XML Document D

Fig. 1. Report Document 1

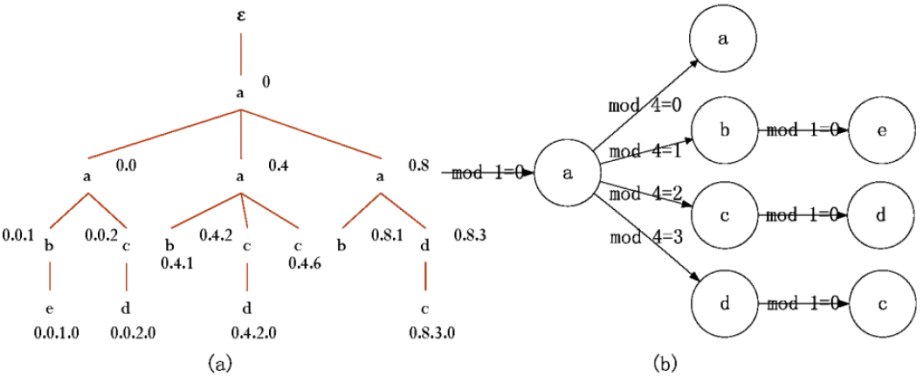


Fig. 2. An XML tree with extended Dewey code

1.2 Encoding Scheme

Encoding schemes (e.g. [2, 11]) are widely used in XML query processing to identity the relationship between XML tree nodes, We use extended Dewey code [11] in this paper, and the encoding of each node, in Fig. 2(a) is shown beside the label. The extended Dewey code of each node can be converted into a label-path, i.e., a sequence of node labels from root to a particular node, using a finite state transducer (FST). The finite state transducer of XML tree in Fig. 2(a) is given in Fig. 2(b).

Consider the encoding 0.0.2 in Fig. 2(a) (for **c**). The first label **a** can be derived, since the first number 0 in 0.0.2 satisfies 0 mod 1 = 0 from the input of FST in Fig. 2(b). The second label **a** will be derived, since the second number 0 satisfies 0 mod 4 = 0 on node **a** in FST. Finally, we derive label **c**, since the last number 2 has 2 mod 4 = 2 on node **a**. The label-path of **c** is derived as "**/a/a/c**".

2 Preliminary Research

Based on the work done before, we call extended Dewey schemes to solve QAV problem. A *distinguished node* of a query Q (resp. view V) corresponds to the answer element, denoted as D_Q (resp. D_V).

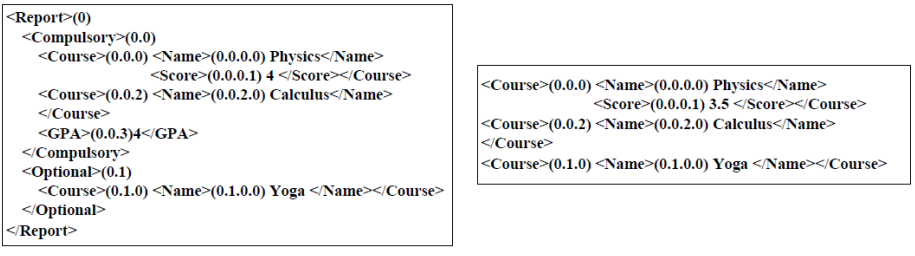


Fig. 3. Report Document 2

We encode the nodes in D in Fig 3(a) using extended Dewey. Consider the query Q, "//Compulsory//Course" in V. Extended Dewey encoding scheme offers information of path through root to the node itself in D, i.e., it can be derived that Course elements (0.0.0, 0.0.2) come from "/Report/Compulsory/Course", while Course element (0.1.0) comes from "/Report/Optional/Course". So the result got by applying Q on V is Course elements (0.0.0, 0.0.2). We already know that by applying Q on D, Course elements (0.0.0, 0.0.2) will be returned, so Course elements (0.0.0, 0.0.2) is an equivalent result.

Lemma 1. Let V be tree patterns and Q be path query. Suppose the original database D that V comes from is encoded using extended Dewey. There exists an equivalent result.

Proof. D is encoded using extended Dewey encoding scheme, hence from the codes of nodes, information of path through root to itself in D can be derived. If query Q is a path query, Q possesses only one path. It is clear whether there is D_Q-element in V corresponded to Q. Because if there exists, Q must be contained by the path derived.

3 Future Work

Most query language of XML can be regarded as the combination of pattern languages and structural expression, which are the generation of tree pattern query.

How much encoding scheme can help to get the equivalent result got by applying tree pattern query on view is a core problem. It still takes time to make sure the ability of encoding scheme according to the query. We may ask indexes to solve the problem remained by encoding scheme.

Acknowledgments. Jiaheng Lu was sponsored partially by the NSF China (60903056)， Beijing Municipal Natural Science Foundation (4112030) and Program for New Century Excellent Talents in University.

References

1. XML Path Language (XPath) 2.0, http://www.w3.org/TR/xpath20/
2. Al-Khalifa, S., Jagadish, H.V., Koudas, N., Patel, J.M., Srivastava, D., Wu, Y.: Structure join: A primitive for efficient XML query pattern matching. In: ICDE (2002)
3. Amer-Yahia, S., Cho, S., Lakshmanan, L.V.S., Srivastava, D.: Minimization of tree pattern queries. ACM SIGMOD (2001)
4. Bruno, N., Koudas, N., Srivastava, D.: Holistic twig joins: Optimal XML pattern matching. SIGMOD (2002)
5. Chen, D., Chan, C.-Y.: Viewjoin: Efficient view-based evaluation of tree pattern queries. In: ICDE (2010)
6. Fiebig, T., Helmer, S., Kanne, C.-C., Moerkotte, G., Neumann, J., Schiele, R., Westmann, T.: Anatomy of a Native XML Base Management System 11(4) (2002)
7. Halevy, A.: Answering queries using views: A survey. VLDB Journal 10(4) (2001)
8. Kaushik, R., Bohannon, P., Naughton, J.F., Korth, H.F.: Covering indexes for branching path queries. In: ACM SIGMOD (2002)
9. Kaushik, R., Shenoy, P., Bohannon, P., Gudes, E.: Exploiting local similarity for indexing paths in graph-structured data. In: ICDE (2002)
10. Lakmannan, L.V.S., Wang, H., Zhao, Z.: Answering tree pattern queries using views. In: VLDB (2006)
11. Lu, J., Ling, T.W., Chan, C.-Y., Chen, T.: From region encoding to extended Dewey: On efficient processing of XML twig pattern matching. In: VLDB (2005)
12. Tang, N., Yu, J.X., Ozsu, M.T., Choi, B., Wong, K.-F.: Multiple materialzed view selection for XPath query rewriting. In: ICDE (2008)
13. Xu, W., Ozsoyoglu, Z.M.: Rewriting XPath queries using materialized views. In: VLDB (2005)

XIO-SLCA: Optimize SLCA for Effective Keyword Search in XML Documents

Xia Li[1], Zhanhuai Li[1], PeiYing Wang[2], Qun Chen[1], Lijun Zhang[1], and Ning Li[1]

[1] School of Computer Science and Technology, Northwestern Polytechnical University, Xi'an 710129, China
[2] China Aeronautics Computing Technique Research Institute Xi'an 710068, China
{lixia,lizhh,chenbenben,Zhanglijun,lining}@nwpu.edu.cn
ljping1000@sina.com

Abstract. Keyword search has attracted a great deal of attention for retrieving XML data because it is a user-friendly mechanism. In this paper, we study the problem of effective keyword search over XML documents. The paper SLCA proposed that keyword search returns the set of smallest trees, where a tree is designated as smallest if it contains no sub-tree that also contains all keywords. The paper SLCA also provided detail description of the Indexed Lookup Eager algorithm (IL) to calculate SLCA. We analyzed and experimental studied the IL algorithm of SLCA deeply, find that there are 3 bugs which should not be disregarded. This paper investigates the problems to correct the existent 3 bugs of the algorithm IL, and proposes an optimize method called XIO-SLCA to improve keyword search quality. We have conducted an extensive experimental study and the experimental results show that our proposed approach XIO-SLCA achieves both higher recall and precise when compared with the existing proposal SLCA.

Keywords: Keyword Search, XML, IR, SLCA, XIO-SLCA.

1 Introduction

The Extensible Markup Language(XML) is becoming the dominant standard for exchanging data over the World Wide Web. As XML becomes the standard for representing web data, how to perform effective information retrieval on XML data has attracted much research interests in recent years[1,2,3]. The INitative for the Evaluation of XML Retrieval (INEX)[1], for example, was established in April, 2002 and has prompted XML researchers worldwide to promote the evaluation of effective XML retrieval.

[1] http://www.inex.otago.ac.nz/

L. Wang et al. (Eds.): WAIM 2011 Workshops, LNCS 7142, pp. 140–149, 2012.

1.1 Our Contributions

Existing studies mainly focus on efficiency of keyword search on XML databases[4,5], and accordingly, how to reduce unrelated results of a keyword query so as to improve the precision is urgent to investigate. We emphasize the quality of keyword search on XML databases in this paper, which is at least as important as efficiency.

To achieve our goal, we first introduce a concept, called XIO(XML Information Object)[6], which is a smallest meaningful XML twig as an information object result. Then we introduce the algorithm to score the XIO similarity between two XIOs. Finally, we investigates the problems to correct the existent 3 bugs of the algorithm IL in the paper SLCA[4], and proposes an optimize method called XIO-SLCA to improve the performance of keyword search. To summarize, the contributions of our work include:

1. We introduce the notion of XIO, which is smallest meaningful information object as a twig of XML data, and propose the algorithm to score similarity of two XIO, which can dispose off the unrelated result.
2. We correct the bug of the algorithm IL of SLCA[4] which may return the root node as a result of some query.
3. We correct the bug of the algorithm IL of SLCA which may omit some real result.
4. We propose the method to judge a SLCA result whether is a XIO to avoid unmeaningful result.
5. We conduct a performance study using real datasets with various characteristics. The results show that XIO-SLCA corrects the bugs, and outperforms the existing SLCA in terms of precision and recall.

1.2 Paper Organization

The remainder of this paper is organized as follows. We discuss the related work and our motivation in Section 2. Section 3 introduces the notion of XIO, and the algorithm to compute XIO similarity. We give an effective algorithm called XIO-SLCA to correct the bugs of the algorithm IL of SLCA as a total solution in Section 4. Experimental evaluations are provided in Section 5. Finally we conclude the paper in Section 6.

2 Related Work and Our Motivation

2.1 Related Work

Keyword search is a proven and widely accepted mechanism for querying in document systems and World Wide Web. It is a user-friendly way of querying XML data, and it allows users to search the information they are interested in without learning a complex query language or knowing the XML structure. However, there are many unrelated results of a keyword query due to the lack

of clear semantic relationships among keywords. In particular, an incompletely specified query may return too many results. Recently, keyword search has been investigated extensively in XML databases. Given a keyword query and an XML data source, most of related work [3,7] first retrieve the relevant nodes matching with every single keyword from the data source and then compute LCA or SLCA [4] of the nodes as the results to be returned. XRANK [8] and Schema-Free XQuery [7] develop stack-based algorithms to compute LCAs as the results. The paper [3] focus on the discussions how to infer return clauses for keyword queries XML data. The paper [9] takes the valuable LCA as results by avoiding the false positive and false negative of LCA. The paper SLCA [4] proposed the notion of SLCA and two efficient algorithms IL and Scan Eager which can produce part of the answers quickly so that users do not have to wait long to see the first few answers. The production of SLCA has been recognized widely. This paper objective is to improve the result quality based on SLCA, so that we will briefly describe the SLCA as 2.2.

2.2 Algorithms about SLCA

The paper SLCA [4] proposed keyword search returns the set of smallest trees containing all keywords, where a tree is designated as smallest if it contains no tree that also contains all keywords. The core contribution of the paper SLCA [4] is the Indexed Lookup Eager algorithm, exploits key properties of smallest trees in order to outperform prior algorithms by orders of magnitude when the query contains keywords with significantly different frequencies.

The notation about SLCA has been proposed in the paper SLCA [4], so that we will not describe them in this paper again. The paper SLCA [4] proposed the IL algorithm detailed, so that we do not describe it again.

2.3 Our Motivation

We analytically and experimentally evaluate the IL algorithm of SLCA, find that there are some bugs which can not be disregarded. This paper investigates the problems to correct the existent 3 bugs of the algorithm IL, and propose an algorithm called XIO-SLCA to correct the bugs of SLCA. XIO-SLCA optimize the existing algorithm IL of SLCA to avoid the meaningless result be returned and to avoid meaningful result be omitted. Our objective is to improve keyword search quality.

3 XML Information Object and XIO Similarity Score Algorithm

3.1 XIO: XML Information Object [6]

In many applications, the goal is to find xml information objects related to xml data set that best match a set of keywords. For example, as shown in Fig.1, one might want to find the article which author is Tom, the node article(15) and

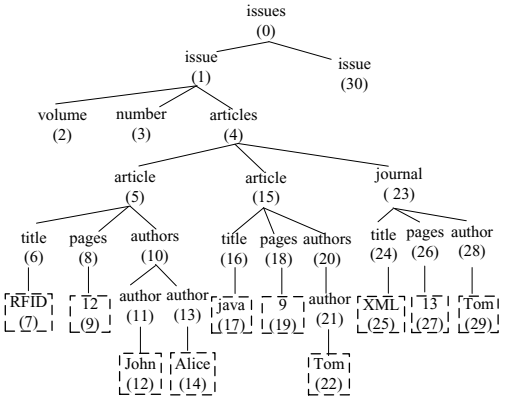


Fig. 1. An example of XML document tree

the node journal(23) will be return as xml information object. When user input the keyword query $Q = \{author\ Tom\}$, we can infer user want to find a XIO named article or journal rather than to find the attribute node author(21) or the node author(28) which is an attribute of the XIO(article). Actually, each xml document can be regarded as an xml information object, the root node is the summarization of the XML contents, and each XML can be regarded as a big XIO which contains many types of small XIO. The detailed definitions are shown as following.

Definition 1. *(XIO[6]) Which is a smallest meaningful XML twig as an information object result. Given a set of labels $l_i|1 \leq i \leq n$ and an XML schema tree T, a XIO is defined as the root node of the subtree T_{sub} of T, such that T_{sub} contains at least one schema node labeled as $l_1, ..., l_n$.*

Definition 2. *(Alias XIO[6]) In $T = (N, E, r)$, there are XIO_1 and XIO_2 are T_{sub} of T, the XIO_1 and XIO_2 is alias XIO, if all the following conditions hold:*

1. $Root(XIO_1) \neq Root(XIO_2)$
2. $SIM(XIO_1, XIO_2) \geq \emptyset$ *Here $\emptyset$ is the similarity threshold.*

The $Root(XIO)$ is the root node label of XIO. $SIM(XIO_1, XIO_2)$ is the similarity score of the two XIOs, detail in equation 1.

3.2 XIO Similarity Score Algorithm[6]

Because the root node label of XIO is different or the expression of certain properties in different ways, we may mistakenly infer the same type XIO as different type XIO. Typical example is the existence of an alias object. For example, as shown in Fig. 1, the node article(15) and the node journal(23) should be regarded as the same type of XIO. In order to avoid mistakenly judged the

same type of XIO into different type XIO, we provide an algorithm to computing the similarity score between two XIO, as equation1.

The structure of same type XIO should be similar. Therefore, we consider two factors to compute the XIO similarity, the semantic and the structural information of XIO. That is, taking into account XIO contains the node information, and to consider XIO contains hierarchy of node. The similarity score algorithm of two XIOs as follows:

$$SIM(O_1, O_2) = SIM\_N(O_1, O_2) * SIM\_L(O_1, O_2) \tag{1}$$

In equation1, O_1 and O_2 are two twigs of T, $SIM(O_1, O_2)$ is the similarity of XIO_1 and XIO_2 $SIM\_N(O_1, O_2)$ is the node similarity of XIO_1 and XIO_2 , detail in equation2). $SIM\_L(O_1, O_2)$ is the node-level similarity of O_1 andO_2.

$$SIM\_N(O_1, O_2) = \frac{|label(O_1) \cap label(O_2)|}{|label(O_1) \cup label(O_2)|} \tag{2}$$

In equation2, $label(O_1)$ is the label of the node which is contained by O_1. When the number of two XIO contains the same node is more, the result value is greater. If all nodes of the two XIO are same, the node similarity score is 1. Thus, $0 \leq SIM\_N(O_1, O_2) \leq 1$.

$$SIM\_L(O_1, O_2) = \sum_{i=1,j=1}^{n} \frac{Min(O_{1L_{n_i,n_j}}, O_{2L_{n_i,n_j}})}{Max(O_{1L_{ni,nj}}, O_{2L_{ni,nj}})} \tag{3}$$

In equation 3, $O_{1L_{n_i,n_j}}$ is the path number between node n_i and node n_j in XIO_1. When the count of two XIO contains the same path is more, the result value is greater. If all paths of the two XIO are same, the node-level similarity score is 1. Thus, $0 \leq SIM\_L(O_1, O_2) \leq 1$. Apparently, $0 \leq SIM(O_1, O_2) \leq 1$.

3.3 Meaningless XIO[6]

The XIO similarity algorithm can avoid return unrelated result. For example, let a query Q(author, Tom, XML) be a query over the XML document in Fig. 1. We can obtain that the XIO is the node journal (23). Since it contains the node labeled as author and title respectively. In SLCA[4] or other keywords search engine, the node articles(4) and the node journal (23) will be return, intuitively, the node articles(4) should not be regarded as a result for the Q. In XIO-SLCA the node articles(4) will be disposed because it contains two child nodes: article(15) and the journal (23), and the two nodes are similarity according to equation1, so that we consider the article(15) and the journal (23) as the same type XIO, their parent node articles(4) is an aggregative node, it can't as a result node because it no independent meaningful. We define the meaningless XIO as follow.

Definition 3. *(Meaningless XIO) In T = (N, E, r), there are many XIO which are Tsub of T, the XIO is meaningless, if one of the following conditions hold. Denoted the root node of XIO by u.*

1. $u \in NV$
2. $\forall v \in N, parent(u, v) = true \rightarrow$
   $(Label(v_i) = Label(v_j))$ *or* $\forall v \in N, (SIM(XIO_i, XIO_j) \leq \emptyset$
   note, here $\emptyset$ *is the similarity threshold.*

For example, as shown in Fig. 1, the node articles(4) and the node authors(10) are meaningless XIO.

4 XIO-SLCA Algorithm

According to the following bugs of IL algorithm, We analyse the cause of each bug and give the correct method, then we propose the XIO-SLCA Algorithm as algorithm1.

bug1: may return the root node as a SLCA result.
bug2: may discard some useful result because of the algorithm error.
bug3: may return some results which are meaningless for user.

4.1 Root Node as SLCA Result

In SLCA[4] may return the root node as a result. Obviously, the root node should not be a result.

In the condition that if the last node of the result set which obtained from the former iteration of the loop is the same with the first node of the result set which obtained form the current iteration of the loop, because that the $v \neq null \ \&\& \ v \not\prec getFirstNode(B)$ is true in the current iteration of the loop at line 10, so the node v as the result be returned, but the node v may be regard as the incorrectness to discard in the latter iteration of the loop.

For example,detach the set S_1 to three parts, suppose the former two parts are small. In the first iteration of the loop at line 5-6, has $B = (0)$, and $v = 0$ into the secord iteration. In the secord iteration of the loop at line 5-6, because of $0 \not\prec 0$, so $B = (0)$, at line 7, $(v \neq null \ \&\& \ getFirstNode(B) \prec v)$ is false, but at line 10, $(v \neq null \ \&\& \ v \not\prec getFirstNode(B))$ is ture, so at line 11, the root node $v = 0$ will be return. Obviously, it is a bug that should not be ignored.

The correct method is to change the program of the IL algorithm, correct the line 10, $v \neq null \ \&\& \ v \not\prec getFirstNode(B)$ to $v \neq null \ \&\& \ v \not\preceq getFirstNode(B)$. In this way, when the node v is the same as the first node of the set B , in the current iteration loop, don't return the node v , when perform the next iteration of the loop, return the set B .

4.2 Discard the Useful Result

In SLCA[4] may discard some useful result because of the algorithm error. Obviously, it is the bug should not be omitted also.

For example, as the fig.2. Given a query $Q1 = \{XML \ Search \ author\}$, meaning to find the the author related with XML Search. In the fig.2, we know

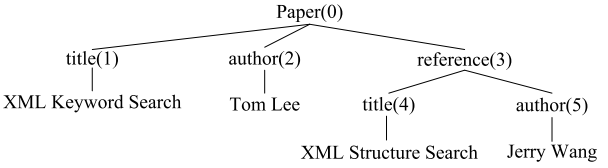


Fig. 2. An example of XML document tree for describe the bugs of IL

that the node $0 : paper$ and $3 : reference$ should be results. The node $0 : paper$ contains all information of the paper which reference the paper denoted by the node $3 : reference$, it is a meaningful result. But as the SLCA algrithm, the node $0 : paper$ should be discarded because there is $parent(\ 0, 3\ ) = ture$.

The correct method is that when compute a SLCA, delete the node from the set $S_1, S_2, ..., S_k$ which is produce the SLCA, avoid to disturb the subsequence compute. For the same example as front, in the fig.2, if educe the result node $3 : reference$ firstly, then to delete the node $2 : author$ and $XML\ Structure\ Search$. The subsequence compute will get the node $0 : paper$ from the node $1 : author$ and $XML\ Keyword\ Search$. Vice versa, if educe the result node $0 : paper$ firstly, then to delete the node $1 : author$ and $XML\ Keyword\ Search$. It will not disturb the subsequence compute also.

4.3 Return Meaningless SLCA Result

The method in the paper SLCA[4] may return meaningless result to user. The main cause is not to consider the semantic information of XML. It is the bug should not be omitted also.

In the XML document, one node should contains many information. For example, a node contains a paragraph of a paper, when user want to get the information about the paper, such as author, year ,title, etc. throuth a query keywords which are just in the same paragraph, as the SLCA algorithm, the node of the paragraph which contains the all keywords will be returned. Obviously, the result can not satisfy user's need. The aim of a query is to get more information than user known, if the result information just matching the query keywords, the query is no meaning for user. In other words, the return node is no independence meaning, should not as a result. How to judge a node is independence meaning or not?

We propose a method to judge the node is meaningless or not, in the Section3.3 The node is meaning exception the NV node and the collection node. The method to judge the collection node as the following.

(1) $parent(u, v) = true$
(2) $\forall v \in N \quad label(v_i) = label(v_j) \| AIO(v_i, v_j) = true$

If a result node v is meaningless then we achieve its parent node u through the function $parent(u, v) = true$, then to judge whether the node u is meaningless

or not, if not, continue, until the node u is an independence meaning node. For example in fig.2 , for a query $Q2 = \{XML,\ Search\}$. the result nodes of SLCA should be: 1 and 4 , but they are useless for user. In XIO-SLCA, the nodes: 1 and 4 should be judged as meaningless node, they can not be as results, so to get their parent nodes 0 and 3 . The nodes 0 and 3 contain more information than the query Q2, so they are useful for user.

We propose the XIO-SLCA algorithm to correct the IL algorithm, as algorithm 1.

Algorithm 1: XIO-SLCA Algorithm

```
   Input: a node set S
   Output: the result SLCAs
 1 begin
 2     v = null;
 3     while (there are mores in S_1 ) do
 4         Read P nodes of S_1 into buffer B;
 5         foreach (i = 2 → k) do
 6             B = get_slca(B, S_i);
 7             if (v ≠ null && getFirstNode(B) ≺ v) then
 8                 removeFirstNode(B);
 9             endif
10             if ( v ≠ null && v ⊀ getFirstNode(B) ) then
11                 if (v is a XIO) then
12                     Output(v);
13                 else
14                     u = parent(v);
15                     Output(u);
16                 endif
17             endif
18             v = removeLastNode(B);
19             Output(B);
20             B = { };
21         endfch
22     endw
23     Output(v);
24 end
```

5 Experiments

We empirically compared the performance of our XIO-SLCA with the SLCA [4] on the real-life data sets Sigmod[2] and Mondial[3]. We implemented both the SLCA and XIO-SLCA in Java. The test cases and the experimental results in Table1. There are 16 test cases from two data sets, QS_1-QS_8 is the test case based on Sigmod[2], QM_1-QM_8 based on Mondial[3]. Here,$precision = \mid R_q \cap R_s \mid /R_q$, $recall = \mid R_q \cap R_s \mid /R_s$, R_q is the real result of Query, the R_s is the standard result. Fig.3 illustrates the experimental results compared the performance of XIO-SLCA and SLCA. We observe that our algorithm achieves better search performance than the methods SLCA. The recall rate and precision

[2] http://www.cs.washington.edu/research/xmldatasets/
[3] http://www.dbis.informatik.uni-goettingen.de/Mondial/

Table 1. Test Case Queries on File: SigmodRecord and Mondial

NO.	Test Case	XIO-SLCA	SLCA
QS_1	Hatfield	5	5
QS_2	author Hatfield title	5	5
QS_3	title Description	15	15
QS_4	Exploiting Parallelism	2	3
QS_5	articlesTuple Exploiting authors	6	6
QS_6	Description Independence William	1	1
QS_7	author Gibson	2	3
QS_8	title Description author Gibson	2	3
QM_1	Organization	168	168
QM_2	abbrev AfDB	1	1
QM_3	country asia	54	54
QM_4	country asia Chinese	13	13
QM_5	country Chinese population	26	26
QM_6	europe republic	23	23
QM_7	monarchy asia	11	11
QM_8	capital name MOC	11	17

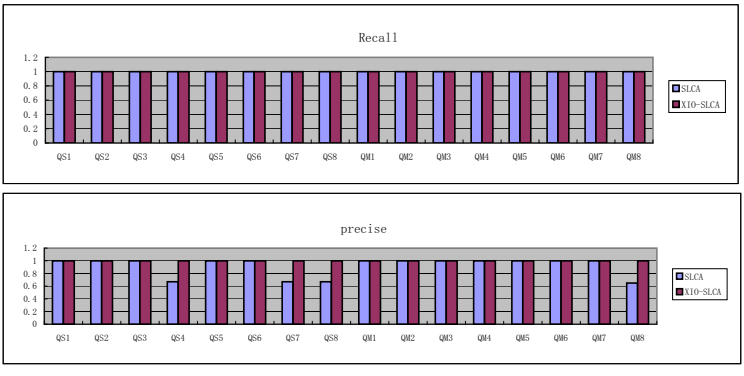


Fig. 3. The recall and precise of the 16 test cases

rate are 100% on XIO-SLCA. Although the recall rate of SLCA are 100%, but there are many results can not satisfy the user query, such as the test case QS_1,QS_3,QS_7,QM_1,QM_2,QM_3, which just return the NV node. But the precision rate, XIO-SLCA outperform over SLCA, because XIO-SLCA firstly judge a SLCA is meaningful or not before return as a result, otherwise, XIO-SLCA will discard the meaningless result node and return its parent node which is meaningful through judge the result is XIO or not, such as the test case QS_1,QS_3,QS_7,QM_1,QM_2,QM_3. The precision rate of XIOF methods and SLCA method is differences because unmeaningful results are returned in SLCA, while XIO-SLCA discard the meaningless result, such as the test case QS_4,QS_7. Hence, our method leads to an improvement over the existing approache.

6 Conclusion

In this paper, we study the thinking of SLCA and analyse the IL algorithm deeply. We find there are three bugs which should not be ignored. We propose the notion of XIO and related algorithms. We analyse each bug, and propose the algorithm XIO-SLCA to correct the bugs of the IL based on the notion of XIO. Then we have conducted an experimental study on real-life XML data sets. The experimental results show that XIO-SLCA achieves both higher recall and precise when compared with existing proposal SLCA.

Acknowledgments. This work is supported by the National Natural Science Foundation of China under Grant No.6080303 and No.JC20110225, the University Innovation Research and Training Program No.XJ1044, and the National High-Tech Research and Development Plan of China under Grant No.2009AA1Z134(863 Program).

References

1. Bao, Z., Ling, T.W., Chen, B., Lu, J.: Effective xml keyword search with relevance oriented ranking. In: Proceedings of the 2009 IEEE International Conference on Data Engineering, pp. 517–528 (2009)
2. Chen, Y., Wang, W., Liu, Z., Lin, X.: Keyword search on structured and semi-structured data. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, pp. 1005–1010 (2009)
3. Liu, Z., Chen, Y.: Identifying meaningful return information for xml keyword search. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, pp. 329–340 (2007)
4. Yu, X., Papakonstantinou, Y.: Efficient keyword search for smallest lcas in xml databases. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, pp. 527–538. ACM (2005)
5. Shao, F., Guo, L., Botev, C., Bhaskar, A., et al.: Efficient keyword search over virtual xml views. The VLDB Journal 18(2), 543–570 (2009)
6. Li, X., Li, Z., Wang, P., Chen, Q.: Xiof:finding xio for effective keyword search in xml documents. In: Proceedings of 2nd International Workshop on Intelligent Systems and Applications, pp. 99–104 (2010)
7. Li, Y., Yu, C., Jagadish, H.V.: Schema-free xquery. In: Proceedings of the Thirtieth International Conference on Very Large Data Bases, vol. 30, pp. 72–83 (2004)
8. Guo, L., Shao, F., Botev, C., Shanmugasundaram, J.: Xrank: ranked keyword search over xml documents. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, pp. 16–27 (2003)
9. Li, G., Feng, J., Wang, J., Zhou, L.: Effective keyword search for valuable lcas over xml documents. In: Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management, pp. 31–40 (2007)

The Development of XML Stored Procedures in XML Enabled Databases

Fahad Alahmari[1] and Eric Pardede[2]

[1] Department of Computer Science & Information Technology
RMIT University, Melbourne, VIC 3000
fahad.alahmari@rmit.edu.au
[2] Department of Computer Science and Computer Engineering
La Trobe University, Melbourne VIC 3086
E.Pardede@latrobe.edu.au

Abstract. In Relational Databases (RDB), the concept of stored procedures is quite familiar to users, these being procedural methods that can be used for data retrieval and manipulation. When users want to query or manipulate data, they can utilize these methods and pass in the parameters. By doing so, the users do not need to perform ad-hoc queries every time they want to perform typical actions. In contrast, XML databases offer no support for such methods as yet, meaning that users have to write ad-hoc XQuery or XPath, which can be time consuming, prone to errors and not user-friendly. This paper investigates how SQL stored procedures can be developed to effectively conduct various XQuery and XPath against XML data within the enabled databases. For implementation, we use SQL Server 2008.

Keywords: XML stored procedures, XML methods, XQuery.

1 Introduction

Stored Procedure has been long established in Relational DBMS. It allows database users to perform various queries that retrieve and manipulate relational data without writing ad-hoc queries every time they want to perform typical actions. On the other hand, XML Database offers no support for such methods as yet, meaning users have to perform ad-hoc XQuery and XPath.

In general, the manipulation of data in a database using ad-hoc queries is often not a simple process for database users. There are many scenarios where the decision to use ad-hoc queries is a difficult procedure. For example, in a hospital, if a doctor wants to retrieve information on a patient, he/she may not be able to create an ad-hoc query to manipulate the patient's information because they would need to understand SQL and XQuery statements, which may be outside their area of specialization. In such a circumstance, the best way to proceed would be to implement stored procedures that would help collect a set of SQL and XQuery statements and then store these in a database. This requires applications that can obtain the values and then pass these on as parameters to the stored procedure.

L. Wang et al. (Eds.): WAIM 2011 Workshops, LNCS 7142, pp. 150–159, 2012.

After a comprehensive analysis of existing approaches, several fundamental concepts relating to XML database stored procedures that need future study have been identified. One of these concepts is active XML, which might make it possible to store procedures in a native XML database. Therefore, the active XML approach is worthy of investigation in this paper. However, in our opinion, the active XML approach involves various areas such as web applications and services, which are beyond the scope of this paper. Hence, we do not investigate this approach in this paper. Rather, we focus on approaches that show how XML stored procedures can manage XML data in enabled databases.

The capabilities of SQL Stored Procedures can be extended to perform XQuery and XPath, which led us to use the term XML Stored Procedures (XML SP) to describe the operation.

This paper is organized as follows: in section 2, we discuss and summarize the background and related work that has been carried out in the area of XML and stored procedures. Section 3 outlines the operations of XML Stored Procedures involving queries and functionalities. The implementation of XML Stored Procedures with several examples will be given in section 4. Finally, the conclusion of this paper and the future work will be discussed in section 5.

2 Background and Related Work

The Active XML [4] project seems the best choice to implement stored procedures in a relational database. As XML performs an exchange of information over the web, an XML document can embed calls to a web service inside the XML document. Therefore, it might store some queries in a document, then call this document from another document to run these queries. The following example is a simple Active XML document which contains a service call. Service call, represented in bold font, will call the service `forecast@weather.com` from the namespace http://activexml.net

Active XML Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<newspaper
   xmlns="xmlns:axml="http://activexml.net">
<title>Le Monde</title>
<date>2-Apr-2003</date>
<edition>Paris</edition>
 <weather>
   % service call
      < axml:call service=”forecast@weather.com” >
          <city>Paris</city>
      < /axml:call>
  </weather>
</newspaper>
```

The service will pass the parameter 'Paris' by a SOAP message to the forecast@weather.com web service. The service answers the request by a SOAP message that contains the return result. After running the above example, the AXML is as follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<newspaper xmlns="
   xmlns:axml="http://activexml.net">
   <title>Le Monde</title>
   <date>2-Apr-2003</date>
   <edition>Paris</edition>
   <weather>
        <temp>16</temp>
   </weather>
</newspaper>
```

The principle idea of calling an XML document in Active XML is similar to the idea in relational and enabled databases. However, the techniques are quite different. Therefore, the possibility of implementing a stored procedure in a Native XML database is quite possible but there is a need for further investigation into how this can be done.

XML Trigger research provides some fundamental information that can be used to build XML Methods. In this background, we are interested more in the XML query trigger because it has a special mechanism that can deal with many new applications in XML-oriented DBMSs.

Grineva and Grinv [7] have shown two implementation methods for the XML query triggers. The first method, called the "native" method can be performed on a set of XML documents and depends on copying XML documents during the query execution. This method is quite simple but it is still limited because of the problem of copying the documents each time. The other method is implemented based on a shadow mechanism which can transform XPath expression to semantic expressions.

The investigation into building XML Methods is in its initial stage. Do and Pardede [3] used XML Database open source to implement XML Method. Their implementation of functions shows "portability of XML Method as an encapsulated style of wrapper program" [3]. However, their implementation was limited by the capability of XML DBMS.

3 Queries and Functionalities of XML Stored Procedures

In an XML-Enabled Database, the way to access the XML data can be done by using SQL and XQuery languages (SQL/XML). SQL/XML has many features and functions that facilitate access to both relational models and XML models.

The new XML data type introduced in enabled DBMS provides new features whereby users can store XML in the database. An XML data type might be used to

create a column, a variable or a parameter to a stored procedure or function. Furthermore, a column of type XML can be associated with an XML schema that is used to validate the XML instances. In addition, if an XML column is associated with an XML schema collection, it is called *Typed XML*; otherwise, it is called *Untyped XML*.

Query languages are used within XML Stored Procedures to retrieve parts of XML instance based on users' criteria. XML SP can perform various queries such as: XQuery 1.0, FLWOR, XQuery methods (query(), value(), nodes(), exist(), modify()) and DML language. In addition, XML SP is capable of running XQuery with SQL to manipulate data either in relational form or data form.

Table 1 shows several functionalities of the XML Stored Procedure. The table contains the taxonomy of the operations that can run within XML SP, its possible functionalities and its general syntax.

Table 1. XML Stored Procedures Functional Overview

Operation	Functionalities	Syntax
Insert	• Inserting a new element into an untyped xml column. • Inserting a new node into an untyped xml column. • Inserting multiple elements into the document. • Inserting attributes into a document. • Inserting a comment node. • Inserting data using a CDATA section • Inserting text node. • Inserting based on an if…else condition statement. • Inserting nodes in a typed xml column	`Insert Expression1 ({as first |as last} into | after | before Expression2)`
Delete	• Deleting element from a document stored in an untyped xml column. • Deleting nodes from a document stored in an untyped xml column. • Deleting elements from a document stored in an untyped xml variable. • Deleting nodes from a document stored in an untyped xml variable. • Deleting elements from a typed xml column. • Deleting nodes from a typed xml column.	`SET @x.modify('delete Expression1')`

Table 1. (*continued*)

Replace	• Replacing values in an XML instance. • Using the if expression to determine replacement value. • Updating XML stored in an untyped XML column. • Updating XML stored in a typed XML column.	`SET @x.modify(replace value of (Expression1 with Expression2)`
Querying	*Based on users' criteria*	`SELECT { Expression}| method} from TableName`

As shown in the above table, each operation has various functions that can perform XML SP against XML instances. These functions can query attributes, nodes, elements, and XML documents. The function of the Querying operation is dependent on users' queries and what users want to extract, which means the classification of queries here is quite difficult to anticipate. The syntax is in general form that can work with all functions but the differences between those functions should be considered. Each function for each operation needs to build its XML SP with its signature syntax.

XML SP can do more than the above functionalities. For example, it can shred XML data into multiple tables. Suppose a company has mapped its data using a relational model but has received particular data in XML format which it now wants to insert into its relational database. If the company decides to turn its relational database into XML, there will be complex issues. However, the best solution in such a scenario is to shred the XML document into a relational table. In SQL Server, the operation of shredding can be achieved within XML SP in two ways: OPEN XML function and XQuery language.

In OPEN XML, the system stored procedure `sp_xml_preparedocument` must be called. This procedure also accepts parameters and loads XML documents into the memory. Once the product has been inserted into a table, it can call the `sp_xml_removedocument` system stored procedure to remove XML data from the SQL Server's memory.

The other way to shred XML within XML SP is to use XQuery language. XQuery functions allow shredding XML without an intensive strain on the memory on the server.

The comparison presented in the table above shows that both OPEN XML and XQuery have the capability to shred XML documents. However, the OPEN XML function consumes a large amount of memory because it needs to call other procedures, whereas XQuery reduces the consumption of the server's memory.

Table 2. A comparison between OPENXML and XQuery within XML stored procedures.

Criteria	Stored Procedure with OPENXML to shred xml	Stored Procedure with XQuery to shred xml
Using XML data type as parameters	√	√
Shredding XML into tables	√	√
Reduce the call of other procedures	X	√
Reduce the consumption of the server memory	X	√

XML SP has other capabilities that can enhance database performance. As most users prefer to use GUI for entering data and queries, the entry data that needs to be manipulated can be converted into an XML model rather than being converted into plain code as shown in Figure 1. Consequently, an XML SP can be built to take the XML model as an input parameter. The procedure with its XML will determine the correct action which should be taken. Basically, the most active queries from users are classified into two categories [5] of XML DML queries: firstly, `Insert, Delete, Update`; and secondly, `Select` statement, which is used to query from the database. Using these two categories, two other XML stored procedures can be created. The first can perform the XML DML queries and the second can perform Select queries. These two procedures can be called from the main middle procedure [5].

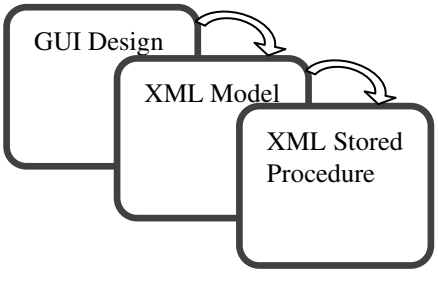


Fig. 1. The output of GUI converts to XML then passes to stored procedure

An XML stored procedure refers to the stored procedure that contains XQuery and XPath to manipulate inputted XML data. In Table 3, when the XML stored procedure

Table 3. The classification of stored procedures' functionalities and actions

Main SP	Stored Procedure	Action
XML SP (Input XML Parameter)	Update	Insert
		Delete
		Update
	Query	Select

received XML as an input parameter, the procedure will fetch the XML parameter and then decide what appropriate action should be performed. This procedure could be developed to call another two procedures (Update and Query) that are performed to take XML parameters, and then within each one, the decision to move to the next step will be made.

As shown in the above table, there are only two procedures can be invoked from the main XML SP. These two procedures perform the action significantly. As a result, the number of stored procedures can be reduced which reflects positively on database performance [5].

4 XML Stored Procedure Implementation

This section provides an insight into how an XML stored procedure can be implemented in an XML Enabled Database. It is important to mention that XML SP can be built and stored in an Enabled Database. Therefore, when users want to query or manipulate XML data, they can utilize these XML SPs without writing an ad-hoc XQuery or XPath every time they want to perform the same queries. The prototype for XML SP is implemented using visual C# 2008 in conjunction with SQL Server DBMS [6].

4.1 User Interface

Designing a user interface allows the evaluation of the business layer (programming code) more than once. Designing a simple user interface was one of our aims when

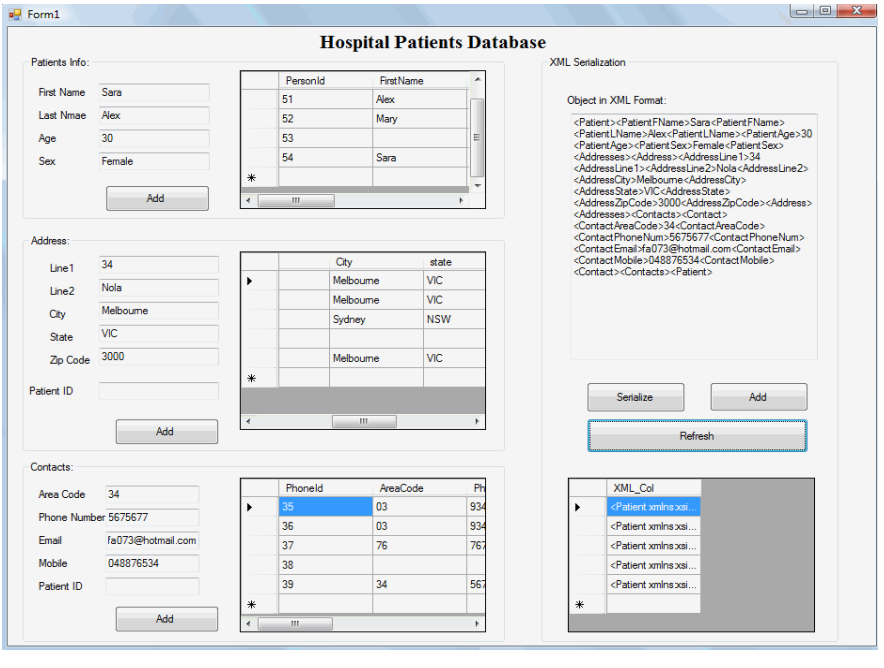


Fig. 2. User interface for evaluation of the implementation

we proposed this work. The main goal of my conurbation UI is to accept some values from a user, then manipulate XML data based on these values. Figure 2 shows a simple form created in Visual Studio 2008. The form is divided into three parts: text boxes to take users' values, data grid to manipulate and view the database tables and the XML section which shows these three tables in the format of an XML document.

4.2 XML SP Syntax to Manipulate XML Instances

Table 1 lists the possible functions that can be implemented by using XML SP. The operation Insert, delete, update, and querying can be implemented within XML SP as follows:

Insert a new element into the XML instance

```
CREATE PROCEDURE [dbo].[insertNewElement] (
   @XmlParameter xml)
AS
BEGIN
  Declare @param xml
  SET @param =(SELECT xml_Coulmn from table_name)
  SET @param.modify('insert sql:variable("@XmlParam")
    as first|as last|into|after|before|
    (queries_path)[element_number]')
  UPDATE table_name set xml_Coulmn =@ param
END
```

Delete existing element from the XML instance

```
CREATE PROCEDURE [dbo].[deleteElement] (
   @XmlParam xml)
AS
BEGIN
  Declare @param xml
  SET @ param =(SELECT xml_Coulmn from table_name)
  SET @param.modify('delete ("@XmlParam")')
  UPDATE table_name set xml_Coulmn =@param
END
```

Replace element into the XML instance

```
CREATE PROCEDURE [dbo].[reaplceNodeValue] ()
AS
BEGIN
  Declare @param xml
  SET @param =(SELECT xml_Column from table_name)
  SET @param.modify(replace value of
   (queries_path[element|node number]
```

```
    with "NewValue")
  UPDATE table_name set xml_Column =@param
END
```

Querying operation from the XML instance (as an e.g.)

```
CREATE PROCEDURE [dbo].[queryTest]
   @xmlData xml
AS
BEGIN
   select @xmlData.query('
        for $cd in CATALOG/CD
        where $cd/PRICE>10
        order by $cd
        return
        <CD>{$cd/TITLE}
        {$cd/PRICE}
        </CD>') as Result
END
```

4.3 XML SP Syntax to Shred XML

XML SP can use XQuery methods. The following example shows two of these methods: `value()` and `nodes()`. The `value()` method extracts the value of the node from the rowset with the type specified. The `nodes()` method returns an unnamed rowset and can also return multiple values. There is no need to include page numbers. If the paper title is too long to serve as a running head, it will be shortened. Suggestions as to how to shorten it are welcome.

Example of how XML SP can shred XML data into multiples tables

```
CREATE PROCEDURE InsertProducts
( @xmlData XML )
AS
BEGIN
   INSERT INTO Products
   ( ProductID, ProductName, Price,Quantity )

      SELECT
         Table1.Column1.value('@ProductID', NT'),
         Table1.Column1.value('@ProductName',
             'NVARCHAR'),
         Table1.Column1.value('@Price', 'FLOAT'),
         Table1.Column1.value('@Quantity','INT'),
      FROM
         @xmlData.nodes('/Products/Product') AS
   Table1(Column1)
END
```

Using XML with SQL Stored Procedures saves users from passing multiple parameters, and facilitates the operation of insert, delete, update and query from XML instances cleanly. If there are no XML stored procedures, then the user will need to implement an ad-hoc query each time they need to insert or query a person. Obviously, rewriting ad-hoc queries is time consuming, prone to errors and not user-friendly.

5 Conclusion and Future Work

This paper presented several functionalities of XML Stored Procedures within an XML Enabled Database. Some functions are implemented in SQL Server 2008 to demonstrate the capability of XML SP to mange XML data and relational models as well. In general, users are able to implement their methods and procedures using XQuery and XPath.

Future work can be extended to schema because most approaches and experiments were built with non-schema-based data. In addition, most of the approaches discussed in this paper are built in SQL Server. Therefore, it is important to extend this work to other enabled management systems in order to make an empirical comparison between these systems and its support of XML stored procedures.

All approaches in this work were implemented on an enabled database. These approaches need further investigation in order to extend this work to XML databases. Active XML [4] gives a few insights on how stored methods can be implemented in native XML databases using a web service. Active XML projects also deserve further investigation.

References

1. Landberg, A., Rahayu, J., Pardede, E.: Extending XML Triggers with Path-Granularity. In: Benatallah, B., Casati, F., Georgakopoulos, D., Bartolini, C., Sadiq, W., Godart, C. (eds.) WISE 2007. LNCS, vol. 4831, pp. 410–422. Springer, Heidelberg (2007)
2. Shao, F., Novak, A., Shanmugasundaram, J.: Triggers over XML views of relational data (2005)
3. Do, W.V., Pardede, E.: On Developing Methods for XML Databases. In: Gervasi, O., Murgante, B., Laganà, A., Taniar, D., Mun, Y., Gavrilova, M.L. (eds.) ICCSA 2008, Part II. LNCS, vol. 5073, pp. 1194–1203. Springer, Heidelberg (2008)
4. Abiteboul, S., Benjelloun, O., Milo, T.: The Active XML project: an overview. The VLDB Journal The International Journal on Very Large Data Bases 17(5), 1019–1040 (2008)
5. Gunathunga, J., Umagiliya, A., Kodituwakku, S.: Leverage the use of XML in dynamic GUI parsing and database stored procedures (2007)
6. Rys, M.: XML and relational database management systems: inside Microsoft® SQL Server™. ACM, New York (2005)
7. Grineva, M.P., Grinev, M.N.: Query triggers for XML DBMS: Efficient implementation on shadow mechanism. Programming and Computer Software 33, 204–213 (2007)

A Slope One Collaborative Filtering Recommendation Algorithm Using Uncertain Neighbors Optimizing

Jingjiao Li[1], Limei Sun[1,2], and Jiao Wang[1]

[1] College of Information Science and Engineering, Northeastern University, Shenyang, China
[2] Information and Control Engineering Faculty, Shenyang Jianzhu University, Shenyang, China
limeisun@126.com

Abstract. Collaborative filtering is one of widely-used techniques in recommendation systems. Data sparsity is a main factor which affects the prediction accuracy of collaborative filtering. Slope One algorithm uses simple linear regression model to solve data sparisity problem. Combined with users' similarities, k-nearest-neighborhood method can optimize the quality of ratings made by users participating in prediction. Based on Slope One algorithm, a new collaborative filtering algorithm combining uncertain neighbors with Slope One is presented. Firstly, different numbers of neighbors for each user are dynamically selected according to the similarities with other users. Secondly, average deviations between pairs of relevant items are generated on the basis of ratings from neighbor users. At last, the object ratings are predicted by linear regression model. Experiments on the MovieLens dataset show that the proposed algorithm gives better recommendation quality and is more robust to data sparsity than Slope One. It also outperforms some other collaborative filtering algorithms on prediction accuracy.

Keywords: collaborative filtering, recommendation system, data mining, knowledge discovery, k-nearest-neighborhood.

1 Introduction

Recommendation system is such software that can acquire personalized recommendations. In accordance with the message type used in the system, recommendation systems can be classified as content-based filtering, collaborative filtering and hybrid filtering [1]. Collaborative filtering is a frequently-used technique in recommendation systems [2]. It helps users to choose items with the aid of other users' experience. The users' preference message (such as ratings to items and browsing time) is recorded to help users filter items. Collaborative filtering is commonly divided into memory-based and model-based. Memory-based collaborative filtering can still be subdivided into user-based [3] and item-based [4]. There are also collaborative filtering algorithms that mix user-based and item-based [5], or mix content-based and item-based [6].

L. Wang et al. (Eds.): WAIM 2011 Workshops, LNCS 7142, pp. 160–166, 2012.

The basic problems in collaborative filtering are data sparisity, cold start and expandability [2]. Data sparisity means that many users' ratings to items are vacant. Because the historical ratings are used to train and predict, the prediction results are influenced due to too sparse historical ratings. To solve data sparisity, a lot of algorithms use probability-based or clustering smooth methods to predict the vacant ratings initially [2, 5, 7].

Slope One is one of item-based collaborative filtering recommendation algorithms that is based on linear regression. It is adaptive to data sparisity and can generate effective recommendation in real time [8]. Slope One algorithm is extensible and easy to be realized. It is used in many online recommendation systems such as Hitflip DVD recommendation system and in Discover MP3 recommendation system.

In this paper, we propose an improved collaborative filtering algorithm that combines uncertain neighbors with Slope One. Firstly, we select different number of neighbors for each user according to the user's similarity to improve the quality of rankings from users participating in prediction. Secondly, we use the selected neighbors' ratings to calculate the deviation between items by linear regression formula. Thirdly, we calculate the ratings that target user rates some products. Lastly, we select the top *k* products as recommendation results to the target user in line with the order of ratings. Experiments on the MovieLens dataset show that the improved algorithm can help to increase the accuracy of recommendations. It is also more robust to data sparisity.

2 Background

2.1 Question Description

In a recommendation system, there are m users and n items. The users set is $U = \{U_1, U_2, ..., U_m\}$ and the items set is $I = \{I_1, I_2, ..., I_n\}$. The ratings in the recommendation system are represented by a $m \times n$ matrix, called rating matrix. The rating matrix is denoted by $R(m,n)$,where r_{ij} means that user U_j rated item I_j by r_{ij}. The value of r_{ij} may be boolean or real. If user U_i hadn't rated item I_j, the value is zero. As shown in Table 1 , $R(m,n)$ is the rating matrix.

Table 1. Ratings Matrix($R(m,n)$)

	I_1	…	I_j	…	I_n
U_1	r_{11}	…	r_{1j}	…	r_{1n}
…	…	…	…	…	…
U_i	r_{i1}	…	r_{ij}	…	r_{in}
…	…	…	…	…	…
U_m	r_{m1}	…	r_{mj}	…	r_{mn}

Recommendation system is such a system that automatically predicts the interest of an active user and recommends appropriate items to him via collecting information (usually rating matrix) from other users. The active user is called target user. The system can provide personalized recommendation to every target user.

2.2 Slope One Algorithm

Slope One algorithm is a rating-based recommendation algorithm which works on the intuitive principle of a deviation between items for users. It determines how much better one item is liked by users than another. It uses the simple formula that only subtracts the average rating of the two items to work out the deviation. Then, given the ratings that user rates some items, the deviation can be used to predict ratings that user rates the other items. The predictor is as $f(x) = x + b$. In practice, $r_{Bj} = r_{Bi} + (r_{Aj} - r_{Ai})$. r_{Bj} is the predicted rating that user B rates item *j*. The prediction process consists of two sections: (1) calculate the deviation $dev_{j,k}$ between item *j* and item k ; (2) predict the unknown rating $P(u)_j$ which means the rating that target user rates item *j*. Recommendation is in line with the predicted ratings. I_{jk} is the user set that both rate item *j* and item *k*. R_j is the item set which are rated by target user.

$$dev_{j,k} = \sum_{u_i \in I_{jk}} \frac{r_{ij} - r_{ik}}{card(I_{jk})} \tag{1}$$

$$P(u)_j = \frac{\sum_{k \in R_j} (dev_{j,k} + u_k)}{card(R_j)} \tag{2}$$

There are two versions of Slope One: weighted Slope One and bi-polar Slope One.

Weighted Slope One: the deviation between two items has a weight c_{jk} which is defined by the number of users that rate both of the two items. $c_{jk} = card(I_{jk})$. The weighted predicted rating is calculated by $P^w(u)_j$.

$$P^w(u)_j = \frac{\sum_{k \in R_j} (dev_{j,k} + u_k) c_{jk}}{\sum_{k \in R_j} c_{jk}} \tag{3}$$

Bi-polar Slope One: because the users maybe like or dislike some items, the ratings are divided into two parts through a threshold that is determined by the average rating of the user. The deviation is also divided into like-deviation and dislike-deviation. The like-deviation is calculated by formula (4) and the prediction rating of user *j* is calculated by formula (5).

$$dev_{j,k}^{like} = \sum_{u_i \in I_{jk}^{like}} \frac{r_{ij} - r_{ik}}{card(I_{jk}^{like})} \tag{4}$$

$$P^{BI}(u)_j = \frac{\sum_{i \in R_j} p_{j,i}^{like} c_{ji}^{like} + \sum_{i \in R_j} p_{j,i}^{dislike} c_{ji}^{dislike}}{\sum_{i \in R_j} (c_{ji}^{like} + c_{ji}^{dislike})} \tag{5}$$

3 Using Uncertain Neighbors Optimizing

Slope One is very simple but the prediction accuracy is comparative with the other excellent prediction algorithm. Although Slope One doesn't predict the vacant ratings to remedy data sparisity, it achieves more accurate prediction. That is to say, in some cases, using more ratings to predict one rating will lower the prediction accuracy. So, the ratings should be denoised before prediction. We use only the ratings from the target user's neighbor to generate the deviation data instead of all users' ratings. Although the ratings used to predict become less, the quality of ratings becomes higher. If some user doesn't have the same interests with the target user, his ratings will influence the deviation between two items and lower the quality of prediction. Here we adopt uncertain neighbors to optimize the ratings. If a user is not the target user's neighbor, his ratings are deemed as noise ratings. After removing the noise ratings, the quality of prediction will improve further.

3.1 Choose Uncertain Neighbors

Choosing nearest neighbors mainly depends on similarity computation. In traditional user-based and item-based collaborative filtering, the common methods are Pearson correlation coefficient, cosine similarity and adjusted cosine similarity[4]. Because cosine similarity doesn't consider the different measure of users and Pearson correlation coefficient is often used to measure the similarity between items, the similarity computation used in this paper is adjusted cosine similarity.

$$sim(u_i, u_j) = \frac{\sum_{c \in I_{ij}} (r_{i,c} - \overline{r_i})(r_{j,c} - \overline{r_j})}{\sqrt{\sum_{c \in I_i} (r_{i,c} - \overline{r_i})^2} \sqrt{\sum_{c \in I_j} (r_{j,c} - \overline{r_j})^2}} \tag{6}$$

To improve the quality of neighbors further, a similarity threshold λ is defined. Another parameter k is also needed. k means the max number of neighbors. λ determines that a user can be the target user's neighbor when their similarity is greater than λ. Because the number of neighbors is uncertain, $k'NN(u_i)$ means the k' nearest neighbors that user i has, where $0 \le k' \le k$.

$$k'NN(u_i) = \{u_x \mid sim(u_i, u_x) \geq \lambda\} \tag{7}$$

3.2 Predict Ratings

By comparing weighted Slope One with bi-polar Slope One, we found the bi-polar outperform the weighted one in accuracy. Thus, we adopt the bi-polar Slope One to optimize the recommendation algorithm. We group the deviation between two items into like-deviation and dislike-deviation.

The improved Slope One using uncertain neighbors is described as follow. k is the max number of neighbors and λ is the user similarity threshold.

Imput: ratings matrix $R(m,n)$, target user u, target item i_j, k, λ

Output: the predicted rating $p(u)_j$ that user u rates item i_j

1. calculate the user similarity matrix from $R(m,n)$
2. calculate the $k'NN(u)$, here the parameter k and λ are needed
3. foreach item i_k that u rated ($k \neq j$) do
4. foreach $v \in k'NN(u)$ do
5. If user v has rated both item i_j and i_k
6. If the rating that user v rated $i_k \geq \bar{v}$
7. Calculate the like-deviation between item i_j and i_k using the rating that user v rated item i_k
8. else
9. Calculate the dislike-deviation between item i_j and i_k using the rating that user v rated item i_k
10. Endif
11. Endif
12. Endforeach
13. Endforeach
14. Using formula(5) to calculate $p(u)_j$

3.3 Data Sparsity Analysis

At first, the improved algorithm doesn't predict the vacant ratings to remedy data sparsity as Slope One does. Secondly, we will prove that the improved one can accommodate sparser dataset than the common Slope One.

Given m users, n items and N ratings in a recommendation system, we define the sparsity degree as N/mn. The smaller sparsity degree means the dataset is sparser. Let $k'NN(u_i)$ represent the uncertain neighbors of user u_i. We can get

$\left|\bigcup_{i=1}^{m} k'NN(u_i)\right| \le m$. Let set $X = \{a \mid u_a \in \bigcup_{i=1}^{m} k'NN(u_i)\}$, then all the ratings that the improved algorithm uses is in set Y= $\{r_{bj} \mid b \in X, 1 \le j \le n\}$. Due to $|Y| \le N$, the sparsity degree in the improved algorithm practically is $\frac{|Y|}{mn}$. Obviously we have $\frac{|Y|}{mn} \le \frac{N}{mn}$. That is to say, the improved algorithm is more robust to data sparsity.

4 Experimental Study

4.1 Data Set

Here we conduct several experiments to compare the recommendation quality between common Slope One and the improved algorithm. In the experiments we adopt MovieLens dataset which sparsity degree is: 100000/(943×1682)=6.30%. MovieLens is a movie website which accepts ratings that user rated movies with range from 1 to 5.

4.2 Measurements of Prediction Quality

We use MAE (Mean Absolute Error) to measure the prediction quality of our proposed algorithm with original Slope One and other collaborative filtering algorithms. p_i and q_i are the actual and predicted ratings respectively.

$$MAE = \frac{\Sigma_{i=1}^{N} |p_i - q_i|}{N} \tag{8}$$

4.3 Experimental Results

In the experiments, the dataset is divided into train set and test set. The proportion of train set is set as 0.8. That means we use 80% of the dataset as train set and the other 20% as test set. The user similarity threshold λ is set as 0.1. 100 users and 200 users are chose randomly each time. The parameter *k* is changed to report the performance of prediction.

We compare original bi-polar Slope One, our proposed UN-bi Slope One(bi-polar Slope One using uncertain neighbors optimizing), traditional collaborative filtering algorithms such as UBCF(User-Based Collaborative Filtering) and IBCF(Item-Based Collaborative Filtering), and recently EMDP(Effective Missing Data Prediction). The results are shown in Table 2.

Table 2. MAE Comparison with Other Algorithms (A smaller MAE means a better performance)

Train Users	Methods	MAE	Train Users	Methods	MAE
	bi-Slope One	0.749		bi-Slope One	0.74
	UN-bi-Slope One	0.743		UN-bi-Slope One	0.735
MovieLens 100	UPCC	0.811	MovieLens 200	UPCC	0.807
	IPCC	0.824		IPCC	0.812
	EMDP	0.769		EMDP	0.761

From the results we can see the bi-polar Slope One using uncertain neighbors optimizing outperforms the other algorithms. Because the improved algorithm removes the noise ratings, it can get better performance.

5 Conclusion

In past collaborative filtering algorithms, due to the data sparsity, the algorithms need to predict the vacant ratings to improve the recommendation quality. This operation increases the complexity of itself. In this paper, we propose an improved Slope One algorithm that is optimized with uncertain neighbors. By removing the noise ratings, it can gain better recommendation quality. Empirical analysis shows that our proposed algorithm outperforms other collaborative filtering algorithms and is more robust to data sparsity.

Acknowledgments. This work was financially supported by the National Natural Science Funds (60970157).

References

1. Choi, S.H., Jeong, Y.-S., Jeong, M.K.: A hybrid recommendation method with reduced data for large-scale application. Trans. Sys. Man Cyber. Part C 40, 557–566 (2010)
2. Ma, H., King, I., Lyu, M.R.: Effective Missing Data Prediction for Collaborative Filtering. In: SIGIR (2007)
3. Herlocker, J., Konstan, J.A., Riedl, J.: An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. Information Retrieval 5(4), 287–310 (2002)
4. Sarwar, B., Karypis, G., Konstan, J., et al.: Item-based collaborative filtering recommendation algorithms. In: Proc. of the 10th Int. Conf. on World Wide Web, pp. 285–295. ACM Press, New York (2001)
5. Wang, J., de Vries, A.P., Reinders, M.J.T.: Unifying User-based and Item-based Collaborative Filtering Approaches by Similarity Fusion. In: SIGIR (2006)
6. Barragans-Martinez, A.B., et al.: A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition. Information Sciences 180, 4290–4311 (2010)
7. Yıldırım, H., Krishnamoorthy, M.S.: A Random Walk Method for Alleviating the Sparsity Problem in Collaborative Filtering. In: Proc. of the 2008 ACM Conference on Recommender Systems, Switzerland (2008)
8. Lemire, D., Maclachlan, A.: Slope One Predictors for Online Rating-Based Collaborative Filtering. In: Proc. of SIAM Data Mining Conference, Newport Beach, California (2005)

A Social Reputation Management for Web Communities

Di He, Zhiyong Peng*,
Liang Hong, and Yu Zhang

State Key Laboratory of Softeware Engineering, Wuhan University
Computer School, Wuhan University
{he,peng,hong}@whu.edu.cn, rudyyuzhang@gmail.com

Abstract. This paper proposes a social reputation management mechanism for modern Web communities, considering the Web 2.0 principles such as friendship ties, share, collaboration and etc. A community is a group of people that have similar topics, so reputation is relevant to the topic. We aggregate the feedbacks, calculate community reputation and overall reputation separately, and then publish them in forms of scores. The case study is conducted to evaluate the reputation model in our Web community management system.

Keywords: Web Community, Reputation System, Topic, Social Ties.

1 Introduction

The World Wide Web is ever growing with various communities focusing on different sets of interests or topics, even for social and professional purposes. These Web communities are essentially characterized by social networks. One of the reasons why social network sites have become so popular today is that these social systems bring collectivities to the world where those people are isolated, far away from each other, and they can share some common interests or purposes easily [1]. Users may become members in various communities and play roles differently in each of them, build social ties, share information and participate in collaborations within a community, depending on the degree of interest or the topic of the community domain. The needs for establishing reputation mechanisms in order to facilitate Web community members' online activities become apparent.

The Oxford English dictionary defines reputation as "the beliefs or opinions that are generally held about someone or something". Reputation in such a social Web community platform thus can be considered as a component of the identity as defined by others members. In developing a reputation system, community members are able to get their reputation scores according to the historical behaviors or quality of shared contents. Therefore the greater the reputation and

* Supported by the National Basic Research 973 Program of China under Grant No. 2007CB310806, The National Natural Science Foundation of China under Grant No.61070011, Doctoral Fund of Ministry of Education of China No.20100141120050.

L. Wang et al. (Eds.): WAIM 2011 Workshops, LNCS 7142, pp. 167–174, 2012.

standing score of a member within the community, the greater the member's class/rank in community can be considered, the more willingness of member's participation will be incented, and the more friends or followers the member will gain to enlarge his or her social network and collaborate with. An eBay[1] empirical study indicates that sellers with better reputations sell their items more than fewer ones [2]. Reputation systems also provide an alternative to help users to create reliable social ties over the Web community [3]. From the point of ties and friendships in social networks, global consumers trust completely in friends' recommendations more than other advertising ways [4]. Although many existing online reputation systems can be beneficial to ecommerce systems and online communities, the designs of online reputation systems are far from ideal. Various issues including usability and effectiveness have been encountered [5]. Some common problems have not been well resolved lead to the unfaithful results or failures of reputation systems. First, there's low incentive for encouraging collaborative rating. Reputation score always comes from the collection and aggregation of members' past behavior and feedback. Many community members fail or refuse to leave feedback. Second, reputation could be artificially affected by malicious participants' actions. Especially in collusion scenarios, for example, some participants collaborate to rate someone positively [6]. The third problem is many reputation systems use an overall score. Or the overall score is simply summed by individual domain ratings. To address the above issues and study reputation management in a social network environment, we propose a reputation mechanism on top of Web 2.0 to support dynamics of web communities and reputation management with social and topic factors, considering the incentive score, friendships and the reputation propagation phenomenon, etc., and then aggregate the feedbacks into an overall reputation score and several community/topic scores. The remainder of this paper is organized as follows. Section 2 introduces related work and our Web community system and the challenge issues for reputation management. Section 3 illustrates the mechanism to compute community members' reputations. Section 4 discusses the simulation experiment on our Web community system. Finally, we addressed some critical challenges and future work in Section 5.

2 Related Work

Most ordinary online community platforms integrate simple metrics into reputation systems. Amazon[2] reputation system is a Web 2.0 version, considering multimedia input, community contribution, interaction and other Web 2.0 principles [7]. eBay also has its own reputation system, the detail is described in [8]. There are some relevant prototype reputation systems. The TRAVOS system is designed for large-scale open system, exploits two information sources (Direct Interaction and Witness Observation) to assess the participants' reputation, and employs a single rating score [9]. The beta reputation system considers

[1] http://www.ebay.com
[2] http://www.amazon.com

the opinion about the provider of information in order to discount the feedback accordingly. However there are no clear incentives in this Bayesian model [10]. Liang and Shi [11] argue that simple averaging rating considering the simplicity of algorithm design is a good method. Reputation also can be propagated between participators [12]. This is common to the basis of the PageRank algorithm: if the sum of the ranking of the nodes with edges to a given node is high, then the ranking of the node itself should be high. The basic idea of these current reputation systems is gathering the rating feedback, especially in online trade or biding systems or peer-to-peer (P2P) networks.

We are developing techniques to build a next-generation community management system . Our goal is to implement dynamic community and member management. The Web community is a bounded system [13], which allows individuals to construct public or semi-public profile, share connections, communicate and collaborate within it. These social network management systems are different from traditional web 1.0 sites in that each community is controlled by topics or rules of engagement. Community systems tend to foster their members who are authenticated in the system and regularly share interest or invoke related Web services. Meanwhile, members may have multiple identifiers, roles or reputations in different communities simultaneously. Reputation in Web communities is such an important aspect for knowledge sharing and collaboration among community members. In this paper, we propose a member-centered reputation mechanism and system for our Web Community Management System (WCMS), with topic and social connection features, in contrast to the related works.

3 Reputation Model and Mechanism

In our demonstrative Web community management scenario, communities can be created by the Service Providers who can provide their specific services to groups of community members. Each community can be defined with some rules. Web users can become community members only if they satisfy the engagement rules. They can join multiple communities at the same time and move from one to another over time. Communities can be divided and merged so that their members can communicate and collaborate efficiently. We considered these factors for reputation management. Our reputation model is used to evaluate the community members' reputations in WCMS. Reputation is then used to help a community member judge the trustworthiness or degree of another. We employ social networking features by accumulating all the available feedback in the community in order to develop a robust reputation estimation mechanism.

3.1 Member and Community

In this section, we present the formal definitions the reputation model in a Web community environment.

Definition 1. Member Profile δ_m: The profile of a community member m is a tuple $\delta_m = \langle R, \phi, A \rangle$, where R is the overall reputation rank score that each member assigns to himself, ϕ is the set of member's friends, and A is the member's other social properties like the first/last name, birthday and etc.

Definition 2. Community Γ_c: A Web community c is a tuple $\Gamma_c = \langle \Delta, T, F \rangle$ which Δ is the set of member profiles in the community c, T is the topic and engagement rule, and F is the set of members' collaborate rating actions in the community c.

We use an Object Deputy [14] Database system to store members as objects and communities as relevant deputy classes. A community may have some conditions which are defined as predicates of the deputy class. Only when a web user satisfies the condition, he or she can join the community. When a web user joins a community, a deputy object is created as an instance of the deputy class representing the community. A web user can have multiple deputy objects belonging to different deputy classes for different interests or topics in communities. A community member is a deputy object of deputy class inherited from the source class and a user can be the member of several communities simultaneously. Using the Object Deputy Algebra, communities can be merged or divided flexibly.

3.2 Reputation Rating

In our approach, community member A assigns a rating to member B in collaborative scenarios, such as bulletin board discussion threads, blog entries, photo albums, or shared acknowledge like paper and etc.

Definition 3. $F_i(j)_c^t$ is the rating feedback assigned by member j to member i at time t in community c. We require that $0 \leq F_i(j)_c^t \leq 10$, and $F_i(j)_c^0 = 0$. Each member will adapt his/her rating of another member based on the shared content and his/her observation. Moreover, a specific rating feedback cannot be allowed to be assigned more than once, taking into account the malicious rating behaviors. Traditional approaches directly combine the ratings assigned by different members.

Definition 4. R_i^t is the reputation rating score of member i at time t. $R_i^t = \frac{\sum_{j \in c} F_i(j)_c^t}{n}$ .We require that $R_t^0 = s > 0$, with the initial starting rating score s. We set a starting reputation score which may be a prevention of an entry barrier. If members start with a reputation of zero, this may be a barrier to entry into the marketplace or community [5]. This incentive strategy is not perfect enough, because we argue that reputation scores should lose relevance over time. After the cross of the entry barrier, member also should be encouraged to engage the community sharing and collaborative actions.

Definition 5. ρ is a time fading factor for past reputation scores, and the time fading factor ρ can be any value between 0 and 1. The discount of past reputation score at time t is $\Delta R_i^t = R_i^t \times \rho^{\Delta t}$.

A low ρ value means that past rating feedback behaviors are shrunk more quickly. When $\rho = 0$ means that past reputation scores are totally ignored. Another extreme case is that for $\rho = 1$ all the historical values are kept forever. This is still a simplistic approach that does not consider the reputations of the witnesses. We should also consider the rater's reputation weight.

Definition 6. $W_j = \frac{R_j^t}{R_{max}^t}$ is the weight assigned to a rating of the rater j at time t, where the R_{max}^t is the maximum reputation score at time t .

Therefore, the rating feedback from those who have a better reputation should be weighed more heavily in reputation scores. This approach will also reduce some malicious members' effects, because they may have very low reputation scores.

3.3 Reputation Propagation

To consider a member's reputation score in a certain community to infer his/her reputation in other aspects. This shows that each member has a set of potentially changing neighbors with whom it is on the friend list.

Definition 7. $\phi_m = \langle \delta_0, \ldots, \delta_n \rangle$ is a friend list of member m. For every friend δ_i will have a reputation propagation factor $P_m(i)$ towards m .
We then define a reputation propagation operator, $\rightarrow$.

Definition 8. $P_m(i) = \delta_i \rightarrow \delta_m = 1 + \frac{2 \times \arctan(R_i^t - R_{avg}^t)}{\pi}$ is a single relation $i \rightarrow m$ on the reputation of member m, where R_{avg}^t is the average reputation at time t.

When consider this reputation propagation factor $P_m(i)$, the level of reputation propagates over a negative link in a referral friend is below the average reputation. Obviously, the value of $P_m(i)$ is between $(0, 2)$.

Definition 9. $P_m(\phi_m) = \phi_m \rightarrow \delta_m = \frac{\sum_{i \in \phi_m} P_m(i)}{n}$ is the group reputation propagation factor towards m , considering all m's friends.

The value of $P_m(\phi_m)$ is also between $(0, 2)$. Our basic idea is that a community member m who has a higher reputation should affect other members' reputation scores on the friend list. A higher reputation means above the average R_{avg}^t. If most friends' reputations of a member are greater than the average, the result of group reputation propagation factor will be greater than 1 and nearly 2. This value indicates that the propagation influence is positive. When the value of $P_m(\phi_m)$ is below 1, which shows that the reputation of m will be reduced by friends with much lower reputations.

4 Preliminary Experiments

In this section, the evaluation of our model is presented. Based on our Web community management system, we collect historical data to evaluate our model

to prove accuracy and stability. Our experiments involve about 65 actual active members within 5 different communities. We analyze every member's behaviors in WCMS and calculate each member's actual reputation.

4.1 Reputation Metrics

We now define some useful metrics in which to intuitively capture the results of our experiments. In order to encourage members to assign rating feedbacks, we take account of recording rating behavior V_t^c , which is the rating feedback score in community c at time t.

Definition 10. R_m^c is the member m's reputation score in community c.

$$R_m^c = (\sum_{i \in c}(\Delta R_i^t \times W_i) + \alpha \times \sum(V_t^c \times \rho^{\Delta t}) + R_t^0) \times P_m(\phi_m)$$

Where V_t^c represents the rating feedback behavior of member m in community c at time t , α is the empirical parameter set by administrators.

Definition 11. R_m is the member m's overall reputation score.

$$R_m = (\sum(\Delta R_i^t \times W_i) + \alpha \times \sum(V_t \times \rho^{\Delta t}) + R_t^0) \times P_m(\phi_m)$$

Where V_t represents the rating feedback behavior of member m at time t , α is the empirical parameter set by administrators.

We compute and then show members' reputation scores to the community public and personal portal in WCMS.

4.2 Case Study

The WCMS dataset is a collection of 65 active members, 352 friendship relations, and it contains over 1,000 rating behaviors. Fig. 1 displays a friendship network of DB community.

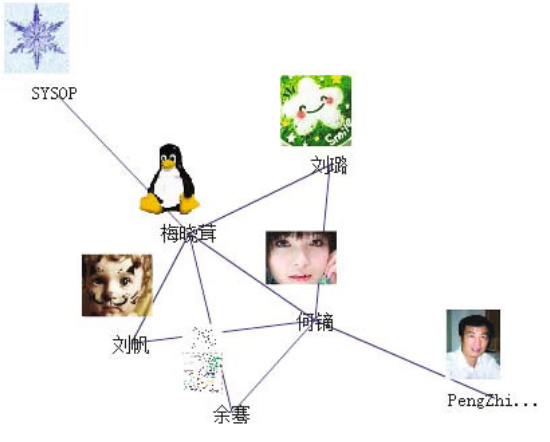


Fig. 1. A friendship network of DB community

Fig. 2. The community reputation score (left) and the overall reputation score

We calculate all members' reputation scores in DB community and their overall scores, for $\rho = 0.998$, $\alpha = 0.01$ and $R_t^0 = 1$. The result of top 5 is shown in Fig. 2. Notice that for a specific member, the community reputation score and overall score are different.

We also trace and investigate a student's overall reputation history changes, illustrated in Fig. 3. After his entry to WCMS and participation in community activities, his overall reputation score increased. At time $T6$, his reputation descended, the reason is that he left laboratory for about 2 months. It was not a short time.

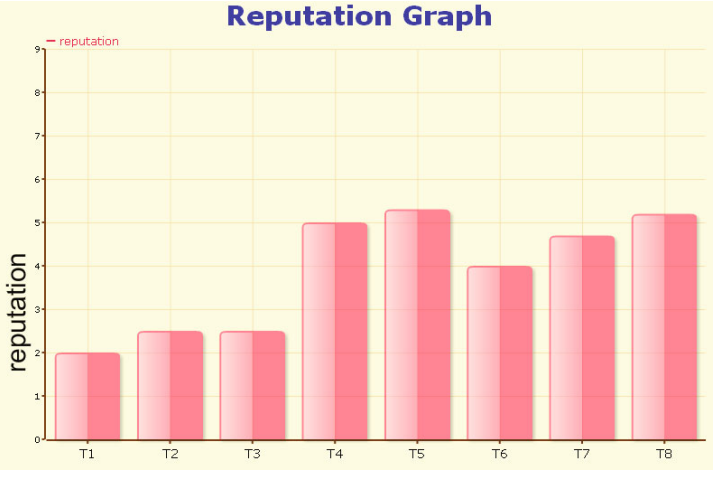


Fig. 3. An overall reputation historical change

5 Conclusion

This paper studies on the reputation rating and management method for a Web community management system. Advantages and disadvantages regarding the current reputation management systems are discussed in detail. We use social network characters, such as friendship and collaboration, considering the reputation propagation phenomenon, aggregate the feedbacks into an overall reputation score more precisely. We take account of varies community topics and

calculate the community reputation scores separately, which is helpful to improve present systems. As future work, we intend to carry out case studies and bring more social network analysis tools with more real community members and their interactions. Our present approach does not fully against malicious rating feedbacks and community members, especially in group conspirator scenarios. We are also interested in analyzing our proposed model with larger datasets to see how the model scales to larger systems.

References

1. Rheingold, H.: The Virtual Community: Homesteading on the Electronic Frontier. MIT Press, Cambridge (2000)
2. Resnick, P., Zeckhauser, P.: Trust Among Strangers in Internet Transactions: Empirical Analysis of eBay's Reputation System. The Economics of the Internet and E-Commerce 11, 127–157 (2002)
3. Mui, L., Halberstadt, A., Mohtashemi, M.: Notions of Reputation in Multi-Agents Systems: A Review. In: 1st International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 304–305. ACM, New York (2002)
4. Giles, M.: A World of Connections: A Special Report on Social Networking. The Economist, London (2010)
5. Malaga, R.: Web-Based Reputation Management Systems: Problems and Suggested Solutions. Electronic Commerce Research 1(1), 403–417 (2001)
6. Resnick, P., Zeckhauser, P., Friedman, E., Kuwabara, K.: Reputation Systems. Communications of the ACM 43(12), 45–48 (2000)
7. Zheng, W., Jin, L.: Online Reputation Systems in Web 2.0 Era. In: Nelson, M.L., Shaw, M.J., Strader, T.J. (eds.) AMCIS 2009. LNBIP, vol. 36, pp. 296–306. Springer, Heidelberg (2009)
8. Resnick, P., Zeckhauser, R.: Trust among strangers in Internet transactions: Empirical analysis of eBay's reputation system. The Economics of the Internet and E-Commerce 11, 127–157 (2002)
9. Patel, J., Teacy, W., Jennings, N., Luck, M.: A Probabilistic Trust Model for Handling Inaccurate Reputation Sources. In: Herrmann, P., Issarny, V., Shiu, S.C.K. (eds.) iTrust 2005. LNCS, vol. 3477, pp. 193–209. Springer, Heidelberg (2005)
10. Jøsang, A., Ismail, R.: The beta reputation system. In: Proceedings of the 15th Bled Electronic Commerce Conference on e-Reality: Constructing the e-Economy, pp. 17–19 (2002)
11. Liang, Z., Shi, W.: Analysis of Rating on trust inference in Open Environments. Performance Evaluation 65(2), 99–128 (2008)
12. Luo, X., Joshua, S.: MultiRank: Reputation Ranking for Generic Semantic Social Networks. In: Proceedings of the WWW 2009 Workshop on Web Incentives. ACM, NY (2009)
13. Boyd, D., Ellison, N.: Social network sites: Definition, History, and Scholarship. Journal of Computer-Mediated Communication 13(1), 210–230 (2007)
14. Peng, Z., Kambayashi, Y.: Deputy Mechanisms for Object-Oriented Databases. In: IEEE 11th International Conference on Data Engineering, pp. 333–340. IEEE Press, New York (1995)
15. Zhai, B., Shi, Y., Peng, Z.: Object Deputy Database Language. In: The Fourth International Conference on Creating, Connecting and Collaborating through Computing, pp. 88–95. IEEE Computer Society, Washington (2006)

A Collaborative Filtering Recommendation System by Unifying User Similarity and Item Similarity

Dongzhan Zhang and Chao Xu

Xiamen University, Computer Science Department, 361005 Xiamen, China
zdz@xmu.edu.cn

Abstract. Collaborative filtering recommendation system based on user similarity has been wildly studied because of its broad application. In reality, users keep partial similarity with larger possibility. Computing the whole similarity between users without considering item category is inaccurate when predicting rating for a special category of items by using collaborative filtering recommendation system. Aiming at this problem, a new similarity measurement was given. Based on the new similarity measurement, a new collaborative filtering algorithm named UICF was presented for recommendation. When predicting rating for the special item, UICF chooses the users as nearest neighbors which have the similar rating feature for the items with the same type of the special item, instead of for all the items. Experimental results show the higher quality of the algorithm.

Keywords: Recommendation system, collaborative filtering, item classification, partial similarity.

1 Introduction

Recommendation system has become an important research field of e-commerce IT technology. At present, many kinds of recommendation algorithm have been given by researchers such as Bayesian network、cluster algorithm, association rules, horting based on graph-theoretic, collaborative filtering recommendation algorithm and so on. Bayesian network creates decision tree models relates to recommendation by training set[1]. In the decision tree model, user information is represented by nodes and edges. Cluster algorithm gathers the users with similar taste into one cluster[2,3] and then the rating of target user for special items is predicted according to the rating of users in the same cluster. Online cluster algorithm can generate recommendation with a high speed by completing clustering process offline. Recommendation system based on association rules recommends items to the target user in line with its current behavior and association rules model[4]. Association rules model can be generated offline, so recommendation system based on association rules can ensure real-time requirements. Horting based on graph-theoretic[5] is a recommendation method based on graphic with its nodes representing users and its edges representing similarity between users. And it gives recommendation for target user by searching its neighbor nodes and then integrating the ratings of neighbor nodes.

L. Wang et al. (Eds.): WAIM 2011 Workshops, LNCS 7142, pp. 175–184, 2012.

Among these recommendation algorithms, collaborative filtering, hereinafter referred to as CF, is the most successful one[6]. The basic idea with traditional collaborative filtering algorithms is that rating of items which are not rated by user is predicted based on rating data of the user's nearest neighbors and then the item with the highest predicted rating is recommended to the user. In order to find out nearest neighbors for the target users, a method of measuring the similarity between users is necessary. But with the expansion of e-commerce systems and the sharp increase of users and items, the rating data is becoming extremely sparse, which reduces the accuracy of nearest neighbors computed for the object user and then makes collaborative filtering algorithm generate bad recommendation. To address this issue, a number of improved methods have been given by researchers, such as item-based collaborative filtering algorithm and its improvement[7-9], Collaborative filtering based on Matrix dimensionality reduction[10-11] and Collaborative Filtering Based on cloud model[12-13].

All these Methods mentioned above can resolve data sparsity and improve the accuracy of user similarity to a certain extent, but they all ignore a important issue that is users keep partial similarity with larger possibility. Computing the whole similarity between users is inaccurate when predicting rating for special type of items by using collaborative filtering recommendation system. Such as, user A and user B maybe keep the same taste for the S type items , maybe user A and user B are all very like the S type items or vice versa. But for T type items, there are large differences between their tastes, maybe user A likes the T types items very much and it is just Opposite to user B. To address this issue, this paper presents a new Collaborative filtering recommendation algorithm based partial user similarity considering the item types.

The rest of the paper is structured as follows. Section 2 introduces the main framework of collective filtering recommendation system based on user similarity and analyses the disadvantage of traditional user similarity measurement. Section 3 presents a new user similarity measurement and a new collaborative filtering recommendation algorithm based the new user similarity measurement, hereinafter referred to as UICF. Section 4 presents empirical studies and section 5 concludes with future directions discussed.

2 Backgroud

This section mainly introduces the framework of user-based collaborative filtering recommendation system and analyses the disadvantage of traditional user similarity measurements.

2.1 User-Based Collaborative Filtering Recommendation System

User-based collaborative filtering recommendation method, referred as UCF below, is based on the fact that users often like the items which are preferred by others users who have same taste with them in the past. User-based collaborative filtering

recommendation system uses the entire user-item rating database to generate recommendations. A typical UCF algorithm proceeds in four steps:

1. Computing User-Item matrix

Compute user-item matrix ***R*** according rating list. Assume the number of users and items are *m* and *n* respectively, the user-item matrix ***R*** is as below.

$$R = \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ r_{21} & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ r_{m1} & r_{m2} & \cdots & r_{mn} \end{pmatrix}.$$

In matrix ***R***, user and item are represented by row and column respectively and the matrix element r_{ij} represents the rating of user *i* for item *j*. r_{ij} is set to zero if user *i* hasn't given rating to item *j*.

2. Computing user similarity matrix ***S***

User similarity matrix ***S*** can be showed as below.

$$S = \begin{pmatrix} S(1,1) & S(1,2) & \cdots & S(1,m) \\ S(2,1) & S(2,2) & \cdots & S(2,m) \\ \vdots & \vdots & \ddots & \vdots \\ S(m,1) & S(m,2) & \cdots & S(m,m) \end{pmatrix}.$$

S(*i*,*j*) represents the similarity between user *i* and user *j*. The traditional measurements of similarity between users are cosine similarity, adjusted cosine similarity and correlation similarity[13-14] and cosine similarity is as formula 1.

$$S(i,j) = \cos(\vec{i},\vec{j}) = \frac{\vec{i}\cdot\vec{j}}{\left\|\vec{i}\right\|\cdot\left\|\vec{j}\right\|}. \tag{1}$$

3. Neighborhood selection

After the similarity computation, UCF algorithms have to select the most similar users for the object user. This is the important step because the final recommendations of UCF algorithm are generated using the ratings of its neighbors. So neighborhood selection has a deeply impact on the recommendation quality. Paper[14] introduces five strategies for neighborhood selection, and the most basic strategy is that selecting the top k nearest-neighbors who have rated the given item.

4. Generating recommendation

After choosing the neighbors for the target user, UCF computes the prediction rating for the given item based on a weighted aggregate of neighbor ratings, and then chooses the item with the highest rating as the recommendation for the object user. Most used aggregating function is weighted sum. To generate the prediction rating of item *i* for user *a*, the weighted sum aggregating function can be presented as formula 2.

$$P_{a,i} = \overline{r_a} + \frac{\sum_{u \in U} (r_{u,i} - \overline{r_u}) \times w_{a,u}}{\sum_{u \in U} \left\| w_{a,u} \right\|} . \quad (2)$$

$P_{a,i}$ is the prediction rating and $\boldsymbol{U}$ is the chosen neighbor set for user a. $\overline{r_a}$ and $\overline{r_u}$ are the mean ratings of user a and user u. $w_{a,u}$ is the similarity measurement between user a and user u.

2.2 Analysis of Traditional User Similarity Measurement

At present, the traditional user similarity measurements used by collective filtering recommendation system compute the user similarity by user-items rating matrix without considering the items category. Assume there are three users: A、B、U and five items:I_1、I_2, I_3、I_4、I_5. And the user-item rating matrix is as table 1.

Table 1. User-item rating matrix

Item-type	Book		Clothe		Clothe
User\Item	I_1	I_2	I_3	I_4	I_5
A	2	1	1	2	3
U	1	2	1	2	X
B	1	2	2	1	3

If we compute the user similarity by traditional user measurement such as cosine similarity measurement or others, assuming the similarity between user i and user j is S(i,j), then S(A,U) is equal to S(B,U) according the rating matrix. Now if we are going to predict the rating of I_5 made by U, then S(A,U) is greater then S(B,U) is more reality, because the rating feature of clothe-items made by user A and user U keeps more similarity, and I_5 is belong to clothe category. Obviously, the traditional user similarity is not inaccurate for the example here.

So the basic idea behind this example is that: Users keep different similarity referring to different item category, if we predict the rating of item which is belong to clothe category, then the similarity between users in clothe category is more weighty than similarity between users in other category when computing the whole users similarity.

3 A New CF Algorithm by Unifying User Similarity and Item Similarity

The similarity between two users is changing as referring to different categories of items. This section introduces a new partial user similarity measurement which computes similarity between users considering the item categories. Based the new user similarity measurement, a new CF algorithm UICF is advanced.

3.1 User Similarity Measurement Considering Item Category

According to the discussion in section 2.2, User similarity measurement without considering the item category is not accurate. So we divide the whole user similarity into some parts, and each part is related to an item category in item set contained by user item matrix. The whole similarity of S(i,j) between user i and user j can be denoted as formula 3.

$$S(i,j) = w_{IT_1} \times s_{IT_1}(i,j) + ... + w_{IT_i} \times s_{IT_i}(i,j) + ... + w_{IT_n} \times s_{IT_n}(i,j) . \quad (3)$$

where $s_{IT_i}(i,j)$ is the partial similarity between user i and user j related to IT_i item category and w_{IT_i} is the weight of partial similarity. Here if we predict the rating of item belonging to IT_i, then w_{IT_i} is set to one and others of similarity weight is set to a value between zero and one. The principle is that the more similarity kept by items in category IT_j with items in category IT_i, the greater value of w_{IT_j} is set to.We can compute the partial similarity $s_{IT_i}(i,j)$ by using traditional similarity measurement. For the example mentioned in table 1, the similarity between user A and user U is as formula 4,

$$S(A,U) = w_{clothe} s_{clothe}(A,U) + w_{book} s_{book}(A,U) . \quad (4)$$

And the similarity between user B and user U is as formula 5,

$$S(B,U) = w_{clothe} s_{clothe}(B,U) + w_{book} s_{book}(B,U) . \quad (5)$$

Because the predicted item I_5 belonging to clothe category, so w_{clothe} is set to one and w_{book} is set to a value which is less than one. We can also get the formula 6 and formula 7 according to traditional similarity measurement and rating matrix,

$$s_{clothe}(A,U) > s_{clothe}(B,U) . \quad (6)$$

$$s_{book}(B,U) > s_{book}(A,U) . \quad (7)$$

According to formulas 4, 5, 6, 7, we can get the result of $S(A,U) > S(B,U)$. The result by the new similarity measurement is more accurate and reality.

For formula 3, determining each weight value of w_{IT_i} is difficult. A simple way is to assume that the similarity between users in each category is independent. According the assumption, we can set zero to each w_{IT_i} except the similarity weight which the predicted item belongs to its related category. If so, the rating data can be used become more sparse. And the similarity measurement based on cloud[15-18] is used for spatial similarity computation to resolve the data sparsity. The computational method of new user similarity measurement is as algorithm 1.

Algorithm 1. Similarity measurement considering predicated item category

Input: item category information, user i,j and the being predicted item z
Output: the similarity $S(i,j)$ between user i and user j.
Steps:
1. for user i and use j, compute their rating sub-vector $\boldsymbol{VS}_i$ and $\boldsymbol{VS}_j$.
2. compute the cloud eigenvector $\boldsymbol{V}_i$, $\boldsymbol{V}_j$ of $\boldsymbol{VS}_i$, $\boldsymbol{VS}_j$ using backward cloud algorithm.
3. compute the partial similarity using $\boldsymbol{V}_i$, $\boldsymbol{V}_j$ according to formula 1.
3. compute $S(i,j)$ by using formula3.
4. return $S(i,j)$.

3.2 Recommendation Algorithm Based on User Similarity and Item Similarity

The complete recommendation algorithm based on user similarity and item similarity is as below.

Algorithm 2. Recommendation algorithm UICF

Input: user rating list and user *UID*.
Output: recommendation for user *UID*
Steps:
1. compute user-item matrix $\boldsymbol{R}$ according user rating list.
2. compute user similarity matrix $\boldsymbol{S}$ using Algorithm 1.
3. select nearest neighbor set $\boldsymbol{U}$ for user UID according matrix $\boldsymbol{S}$.
4. for each items which user *UID* hasn't made rating for, predict it rating using formula 2 and nearest neighbor set $\boldsymbol{U}$.
5. choose the item as recommendation which has the highest predicted rating.

3.3 Items Classification

For users and items, they both have their own category information. But sometimes the original category information of items is not suitable or is not reachable, at this situation we can classify items into categories by their rating features made by users. The basic idea of classification by rating features is that two items belong to the same group if they get the similar rating features from users. If movie *a* and movie *b* both get high ratings from children and get lower rating from adults, then movie *a* and

movie b are both cartoon with a high probability, and should be classified into the same category. Referring to dataset of MovieLens, users have their own category domain of {educator, engineer, …} and each user i has its own category information r_i, where r_i is (i,c_i) and c_i denotes which category user i belongs to. Users' own category information set $\boldsymbol{UR}$ is $(r_1,r_2,\ldots,r_s)$ where s is the number of user categories, so do items.

For the special item i, its users rating feature tuple is $\boldsymbol{T_i}$ $(t_1,t_2,\ldots,t_s)$, where s is the count of user categories and t_i is its mean rating from i category users. We use the k-center clustering method to classify items into different categories and the algorithm is as follows.

Algorithm 3. Items classification algorithm based on Ratings Features

Input: user-item matrix $\boldsymbol{R}$, users' own classification record set UR and group count k
Output: k groups of items
Steps:
1. for each user i , compute its rating feature tuple $\boldsymbol{T_i}$.
2. choose k users rating feature tuples randomly as the center point of k groups.
3. repeat.
4. for each no-center point $\boldsymbol{T_i}$, compute its similarity with the k center points using formula 1 and assign $\boldsymbol{T_i}$ to the group whose center point has the most similarity with $\boldsymbol{T_i}$.
5. randomly choose a no-center point $\boldsymbol{T'}$, exchange $\boldsymbol{T'}$ with the center point $\boldsymbol{T_c}$ of the group which it stays in.
6. compute the exchange cost cs.
7. if $cs<0$,then go back to step 4.else go to step 8.
8. return.

4 Empirical Evaluation

4.1 Experimental Setup

The dataset used by this paper was collected by MovieLens. The dataset includes totally 100 000 ratings for 1682 movies made by 943 users. The dataset was randomly divided into training set and testing set by 80%/20%. The method for evaluating the quality of a recommendation system can be mainly categorized into two classes: statistical accuracy metrics and decision support accuracy metrics[19-20].Mean absolute error(*MAE*) of statistical accuracy metrics is a wildly used method. Assume the prediction set of user rating is $\{p_1,p_2,\ldots,p_N\}$,and the corresponding factual user rating set is $\{q_1,q_2,\ldots,q_N\}$,then *MAE* can be expressed formally as formula 8,

$$MAE = \frac{\sum_{i=1}^{N} |p_i - q_i|}{N}. \tag{8}$$

So the lower the *MAE* is, the higher the quality of a recommendation system is.

4.2 Experiment Result and Analysis

In experimental evaluation, we compare three traditional similarity algorithms basic cosine, correlation and adjusted cosine with the UICF algorithm presented in this paper. The size nearest neighbor set for objective user is from 10 to 60.Figure 1 shows the experiment result. From fig. 1, we can see the UICF surpasses the CF algorithm based on traditional similarity measurement in recommendation quality.

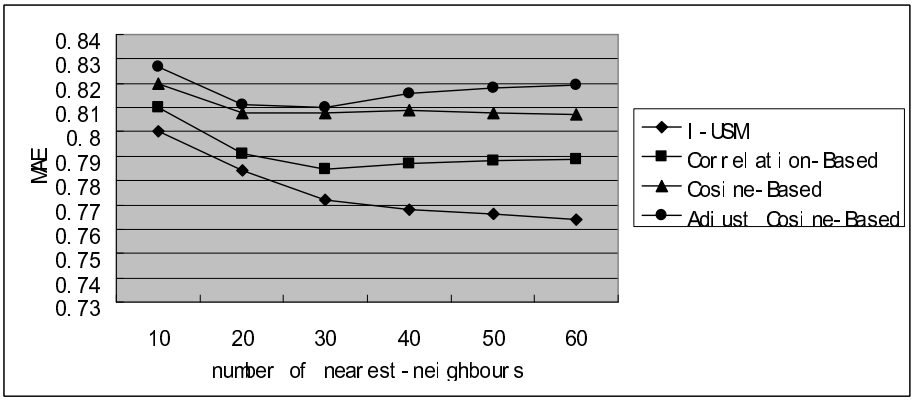


Fig. 1. MAE-Nearest Neighbors number graph

5 Conclusions

In this paper, considering the inaccuracy of user similarity measurement using the whole rating features, a new user similarity measurement based on partial similarity measurement was given. A new collaborative filtering algorithm named UICF was presented for recommendation based on the new similarity measurement. We compare tree traditional similarity algorithms basic cosine, correlation and adjusted cosine with the UICF algorithm. Experimental results show the higher quality of the recommendation algorithm.

Acknowledgments. My deepest gratitude goes first and foremost to Professor Dong zhan Zhang, my supervisor, for her constant encouragement and guidance. He has walked me through all the stages of the writing of this paper. Without his consistent and illuminating instruction, this paper could not have reached its present form. I also

owe my sincere gratitude to my friends and my fellow classmates who gave me their help and time in listening to me and helping me work out my problems during the difficult course of the paper.

References

1. Chickering, D., Hecherman, D.: Efficient approximations for the marginal likelihood of Bayesian networks with hidden variables. Machine Learning 29(2/3), 181–212 (1997)
2. Dempster, A., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society B 39, 1–38 (1997)
3. Thiesson B., Meek C., Chickering D., Heckerman D.: Learning mixture of DAG models. Technical Report, MSR-TR-97-30, Redmond: Microsoft Research (1997)
4. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Analysis of recommendation algorithms for E-commerce. In: ACM Conference on Electronic Commerce, pp. 158–167 (2000)
5. Wolf, J., Aggarwal, C., Wu, K.L., Yu, P.: Horting hatches an egg: A new graph-theoretic approach to collaborative filtering. In: Proceedings of the ACM SIGMOD International Conference on Knowledge Discovery and Data Mining, pp. 201–212 (1999)
6. Breese, J., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proc. of the 14th Conf. on Uncertainty in Artificial Intelligence, pp. 43–52 (1998)
7. Karypis, G.: Evaluation of item-based top-n recommendation algorithms. In: Pro. of the 10th Conf. on Information and Knowledge Management, pp. 247–254 (2001)
8. Sarwar, B., Karypis, G., Konstan, J.: Item-based collaborative filtering recommendation algorithms. In: Proc. of the 10th Int Conf. on Word Wide Web, pp. 285–295 (2001)
9. Chunxiao, X., Fengrong, G., Sinan, Z., et al.: A collaborative filtering recommendation algorithm incorporated with user interest change. Journal of computer Research and Development 44(2), 296–301 (2007) (in Chinese)
10. Liang, Z., Naijing, H., Shouzhi, Z.: Algorithm design for personalization recommendation systems. Journal of Computer Research and Development 39(8), 986–991 (2002) (in Chinese)
11. Junfeng, Z., Xian, T., Jingfeng, G.: An optimized collaborative filtering recommendation algorithm. Journal of Computer Research and Development 41(10), 1842–1847 (2004) (in Chinese)
12. Guangwei, Z., Deyi, L., Peng, L., Jianchu, K., Guisheng, C.: A Collaborative Filtering Recommendation Algorithm Based on Cloud Model. Journal of Software 18, 2403–2411 (2007) (in Chinese)
13. Shuliang, W., Yuan, X., Meng, F.: A Collaborative Filtering Recommendation Algorithm Based on Item and Cloud Model. Wuhan University Journal of Natural Sciences 16, 016–020 (2011) (in Chinese)
14. Zhang, J., Pu, P.: A recursive prediction algorithm for collaborative filtering recommender systems. In: Proceedings of the 2007 ACM Conference on Recommender Systems, pp. 57–64 (2007)
15. Dy, L.: Artificial Intelligence with Uncertainty. National Defense Industry Press, Beijing (2005) (in Chinese)
16. Dy, L., Cy, L.: Study on the universality of the normal cloud model. Engineering Science 6(8), 28–34 (2004) (in Chinese)

17. Dy, L., Cy, L., Du, Y., Han, X.: Artificial intelligence with uncertainty. Journal of Software 15(11), 1583–1594 (2004) (in Chinese)
18. Dy, L.: Uncertainty in knowledge representation. Engineering Science 2(10), 73–79 (2000) (in Chinese)
19. BinQuan, Z.: A collaborative filtering recommendation algorithm based on domain knowledge. Computer Engineering 31(21), 7–9 (2005) (in Chinese)
20. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-Based collaborative filtering recommendation algorithms. In: Proc. of the 10th World Wide Web Conf., pp. 285–295 (2001)

Supporting Query over Dynamic Combination of Data Sources for Social Media

Rongrong Li, Weixiang Zhai, and Zhiyong Peng

Computer School, Wuhan University, 430072
{rrli,peng}@whu.edu.cn, zhaiwx1987@hotmail.com

Abstract. Meta search and social media are combined in our work to build a platform to share cross-media data. Data are extracted from web on demand with meta-search engine since more accurate and complete retrieval results can be provided than a single SE. The key to affect the performance of data extraction is the schedule of data sources to provide data efficiently. A uniform model to describe data sources is suggested, with this model a method to construct query plan graph and an algorithm to get the optimal query plan are proposed. A dynamic and adaptive adjustment approach for query plan execution is described to deal with unexpected failure of data source connection or data extraction. The details of wrapper manager encapsulating data sources are explained at the end of the paper.

Keywords: Meta search, Social media, Combination query, Dynamic and adaptive adjustment.

1 Introduction

The popularity of Internet has led to the rapid growth of data on the Web. MetaCrawler, the first Meta Search Engine (MSE) [1], was released publicly on July 7, 1995. Up to now, MSEs have been developed rapidly since they can provide more accurate and complete retrieval results than a single SE.

MSE utilize different kind of data sources to provide data, such as, ordinary Search Engine (SE), surface web and hidden deep web. Most of the MSEs include three mechanisms, retrieval on demand, agent for retrieval interface and presentation of retrieval results. The key to affect the performance of MSE is how to schedule DSs to provide data efficiently.

Social media has already become a new way to reflect social nature of users, it provides tools and platforms for people to share ideas, opinions, experiences. Interactions are established within social network and aggregation platform, in which user's experience of being served is the most important of all. The main features of social media are the new way of interaction and information representations, they are

L. Wang et al. (Eds.): WAIM 2011 Workshops, LNCS 7142, pp. 185–196, 2012.

dependent on the understanding and generation of shared events from different aspects. Most of Social media sites, such as Facebook, MySpace, FilmAffinity and Flickr [3] include texts, images, audios or videos, while few of social media provide audios.

Since music itself has a natural social attributes, people use different music to express their own unique taste. People having similar taste in music often gather together to form a "tribe". Our goal is to modify the prototype based on our three layers architecture for Web Data Management (WDM) [4] to effectively share both music and semantic information of music extracted from web according to user's requirements, also allow users to exchange receptions of music, discover communities and automatically recommend new music to users.

The advantage of our platform is that data is extracted from web on demand (as illustrated in **Fig.1.**), it needs little storage to store media data which can be obtained from web with metadata on demand. Unstructured musical data with rich semantic relationships can be stored and managed by our object deputy database TOTEM [4] efficiently. Metadata is retrieved to complement the user's requirement into queries to be dispatched to relevant DSs to retrieve media data. Since the dynamicity and quality differences of data sources, while WDMS lacks the flexible capability to combine DSs and adaptive mechanism to adjust query plan execution, user's requirement may not be satisfied.

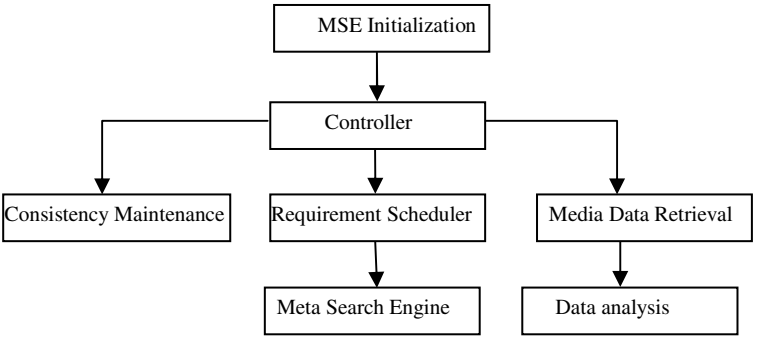


Fig. 1. Modules in Data Extraction Layer

It is the input and output of a DS that differentiate the query capabilities. The approach of DSs organization proposed in this paper separates the implementation of a query executed on a DS from coding for logical layer completely.

Our work is to study the approach of automatically combining DSs to satisfy user's need, including dynamic loading to construct a query plan, and dynamically adjusting the execution of plan. This paper makes the following contributions:

- A uniform model is proposed to describe interface parameters of a DS, DSs with similar interface parameters are classified into Virtual Data Sources.
- A query plan graph is proposed and an algorithm to generated optimal query plan is proposed based on the graph.
- An approach of dynamic and adaptive adjustment for query plan execution is proposed to adapt to unexpected DS changes.

The remainder of this paper is structured as follows. Section 2 introduces the uniform model for DSs. Section 3 introduces construction of query plan graph, and the algorithm to get the optimal query plan. The adjustment approach for query plan execution is proposed in Section 4. The implementation of combination query is illustrated in Section 5. Section 6 introduced related works, and Section 7 concludes.

2 Modeling Data Source

Metadata are the basis for media data search, high quality metadata mean media data can be obtained more likely, or metadata may not be accepted by searching engine for metadata. The key to affect quality of service of WDMS is the quality of metadata. Data space defined by user can only be described with incomplete metadata due to the ambiguity and uncertainty of user requirements. It is critical to complete metadata for effective media data search, while the capability of a single DS is too limited to provide complete metadata, multiple DS can be united. All Metadata involved in data spaces are stored in TOTEM without redundancy, based on the feature of flexible object view a data spaces is described with list of metadata. In this paper, global schemas ($attr_1$,$attr_2$,…,$attr_i$,…,$attr_n$) for metadata is assumed to be known. For example, *Song* (*Singer, Album, Title, Genre, Publish*).

2.1 Data Source Model and Interface Description

Although interface parameters of DSs are various, most of them are consistent with global schemas, some are specific to a few DSs.

DS is described with a six-member tuple (*id, server, global_input, global_output, local_input, local_output*).

- *id*(DS identity): each new DS involved in WDMS is assigned an integer identity.
- *server* is the string id of a DS. For example,"www.ctrip.com".
- *global_input* represents metadata attributes that can be used as the DS input.
- *global_output* is similar to global_input, illustrates attributes in metadata schema included in the output of a DS.

- *local_input* is different from the global input and output mapping with the metadata schema, the value of this element is effective inside the current DS or relevant data cources.
- *local_output*, denoted as {<$attr_i$, *dsid*>} similar to *local_input*. The value of $attr_i$ is *null* means no *local_output*, *dsid* represents the id of DS which can accept the *local_output* as input. *local_output* can be viewed as the bridge to connect to other DSs to obtain metadata with more *not null* attribute values.

For example, there are 4 attributes in metadata, *global_input* $(0110)_2$ (i.e.$(6)_{10}$) means the 2nd and 3rd attributes in metadata schema are input parameters. DS with certain input parameters encoded with an integer can be found more quickly.

Data Source Interface

The interface is defined as (*id, AttrList, UR*). *AttrList* is a list of attributes values can be used as queries extracted by projection on results returned from other DSs. *UR* (User's Requirement) corresponds to metadata attributes used to filter extracted data.

DS wrapper features, data extraction and analysis based on page template, are encapsulated in Wrapper Manager (WM). It needn't to care about how to get data with combination query in upper logical layer but to focus on the interfaces. Information about query plan combining DSs are sent to WM, then WM deliver relevant query to DSs. WM will output results after a series of process, including dispatching query, acquiring html, parsing page, de-noising, filtering with rules. The wrapper of a new DS will be incorporated into system simultaneously with DS model.

2.2 Virtual Data Source (VDS)

It is very expensive to directly combine DSs when there are enormous DSs. It is the contribute to metadata complementation directly global attributes in input and output that affect the combination and differentiate query capability of a DS from others, DS subsequences haveing similar capability may be contained in different combinations for query plans. Metadata schema are rarely changed, so the concept of VDS is proposed to describe these common subsequences of DSs. Some DSs contribute indirectly to metadata complementation, they can be combined into a DS subsequence, the last one of them output more complete metadata after accepting *local_output* returned by another DS, and the first one only accept a *global_input*. The subsequence of DSs also is included into a *DsSet* as a whole.

VDS, a smallest unit to be combined, is a class of DSs denoted as *VDS(id, global_input, global_output, local_input, local_output, DsSet)*.

Where global_output of each *VDS* is larger than 0, it means its can output more completed metadata than *global_inpu; DsSet* is a link list in which single DSs or DS subsequences are included.

global_input and *global_output* of DSs contained in a *DsSet* are the same. *global_input* of the first DS in a subsequence contained in *DsSet* and *global_output* of last one are the same as other single DSs contained in *DsSet*. The general process of DS classification for the list of DSs (denoted as *DS_List*) is as follows:

Step 1: Traverse *DS_List* and find successors for each DS which can accept output of the DS as input.

For each DS S in *DS_List* **Do**

Create a *VDSi* for *S*, *VDSi.DsSet={S};*

If *S. global_input*≠-1 && *S.global_output* = -1 && *S.local_output*≠*NULL,* //ds accept a *global_input* and return *local_output* but no *global_output*;

Find all successors *S_F* of *S*, let *VDSi.DsSet= VDSi.DsSet*∪ *S_F*. As illustrated in **Fig.2.**, DS with *id*=10 is the successor of DS 9 since DS 10 can use the *local_output* of DS 9 as its *local_input*;

Get next DS in *DataSource_List*;

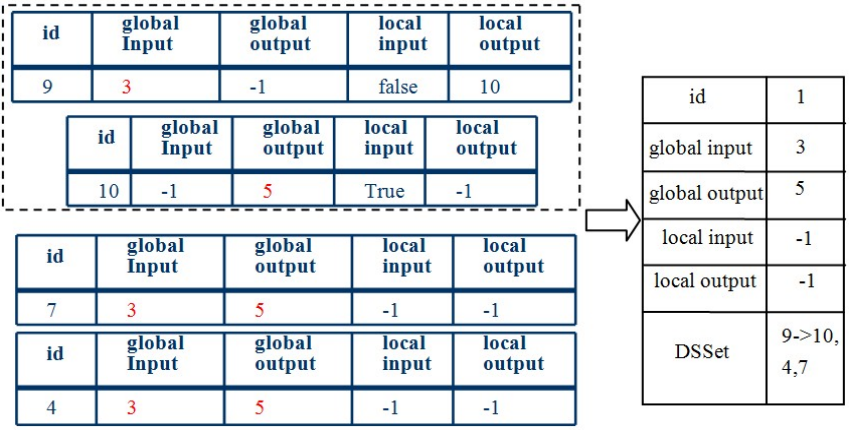


Fig. 2. Generation of a Virtual Data Source

Step 2：Merge all *VDSs* having the same *global_input* and *global_output*. Data sources sequence 9->10 , DS 4 and DS 7 are merged into a *VDS* **in Fig.2.**

Step 3：Find the *VDS* which has only *local_input* and no *global_input*, check if its output can be used as *local_input* of another *VDS* which return only *global_output* combine them into a sequence and add sequence into *DsSet*; if no, then delete it from the list of VDSs.

A streamlined list of *VDS* can be got after classification, its size is much smaller than original size of DSs list. It greatly reduces the number of nodes in combination, thus the

scale of data sources combination is only determined by the number of attributes in metadata and the number of data sources or VDSs which have both global_output and local_output.

3 Generate Query Plan Graph

3.1 Query Plan Graph

A query plan graph $G=<V,E>$ is a directed acyclic graph in which a ***vertex*** (also called *StateNode or SN*) represent a certain metadata status. Data structure of vertex is named *StateNode*(*states, local*). Similar to *global_input* mentioned in section 2.2, *states* is an integer representing status of metadata being completed. If *local* is a non negative integer, it represents id of a *VDS* which can use data in current *StateNode* as *local_input*; Otherwise, it means that *StateNode* has no local data. An edge <*u,v*> represents a *VDS*, *u* represents the input data of a *VDS* and *v* represents output, the *global_output* of a DS generally includes more *not-null* attributes than its *global_input*.

The aim of metadata searching is to complement attributes values to reduce the ambiguity and uncertainty of incomplete metadata, so all bits of *status* field in vertex $StateNode_n$ (*final,-1*) (i.e. SN_n in **Fig.3.)** are 1. That means all attributes in completed metadata having not-null values. The source vertex of query plan graph stands for user's requirement represented by a binary integer.

Constructing Query Path Graph

After DSs are classfied into *VDS,* the query plan graph can be constructed. A relaxation operation (as dotted rectangular illustrated in **Fig.3.**) is adopted to establish relationships between user's requirement and DSs, a vertex is added with an the edge added to generate a query plan tree, merge the same nodes in the tree and generate a query plan graph.

Defintion 3.1. A ***path*** exists between $StateNode_i$ and $StateNode_j$ if there are at least one VDS which can take *states* in $StateNode_i$ as input and output *states* in $StateNode_j$. One condition of the following two must be satisfied:

- If *vds*[*local_input*] is null, (*vds*[*global_input*] & $StateNode_i$.*states*) = = *vds*[*global_input*] must be satisfied;
  Otherwise, vds[*id*] = = $StateNode_i$.*local* must be satisfied;
- (*StateNode*$_i$.*states* | vds[*global_output*]) = = $StateNode_j$.*states* && *vds*[*local_output*] = = $StateNode_j$.*local.*

DsSet of a path, similar to *DsSet* of VDS, includes *DsSets* of all *VDS*s involved in the path. Because the content of user's requirement is inconsistent with the *global_input* of VDS, successors of initial node will be constructed directly with query relaxation, they satisfy the following conditions:

- The *states* field of the node is the combination of attributes in requirement, i.e., *(states & request) = states*;
- The value of the node's *states* is accord with the input of some VDS.

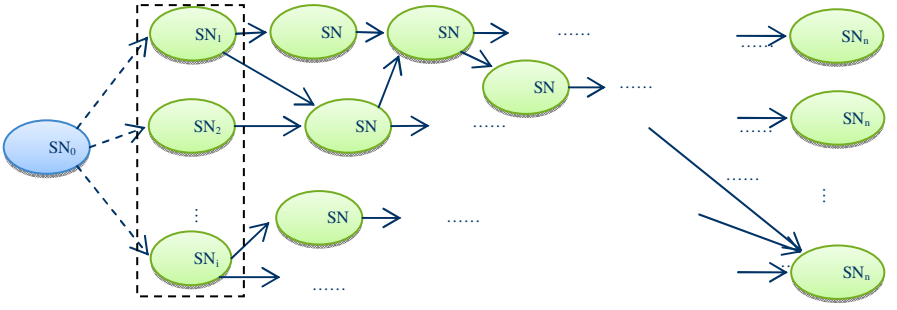


Fig. 3. Query Plan Graph

For example, if a *request* (*global_input*) is '01101', then 2^n-1 possible *states* of its successors are 00001,00100,01000,00101,01001, 01100, 01101, invalid *states* are removed according to the list of *VDS*s. Constructing a query plan graph is a process to complement the attributes in metadata gradually.

Two approaches can be used to construct query plan. One is to generate a tree for construct query plan, the leaves are final nodes, and the number of *final* nodes is the number of query plan trees. Another is to generate the query plan directly during involving new nodes in graph one by one. A new node will be judged whether it has already been included in the graph, then decide whether to generate new node or add new path. The former is easier to be understood and implemented, while cycle may be found in the process of combination in the latter.

3.2 Optimal Query Path

Except for the data source model mentioned in Section 2 representing the property of a DS, there are three other properties to reflect the quality of the DS as follows:

- Total time expense: total time of history queries executed on a DS;
- Total query times: total time of queries executed to a DS;
- Times of valid queries: times that valid data could be got after query submission.

Weights of Edges in Query Plan Graph

In order to evaluate the quality of DS, precision of query (as equation 1) and average response time (as equation 2) are used to measure of data source quality.

$$qp = \frac{valid\_query}{total\_query} \tag{1}$$

Where *valid_query* stands for the times acquiring valid data after query submission; *total_query* stands for the total query times, *qp* reflects the history precision of DS.

$$avg\_resp\_time = \frac{total\_time}{total\_query} \tag{2}$$

When user's requirement is scheduled the first time, weight of the edge is assigned the response time of DS to return data as soon as possible; when the requirement is executed again, weight is assigned the precision of DS to guarantee quality. The weight of edge is set to *avg_resp_time* or *1/qp* of available DS. If an edge corresponds to a list of DSs, its weight is assigned sum of *avg_resp_time* or *1/qp* of DSs in the list.

Algorithm for Searching for the Optimal Query Path

Definition 3.2. A query plan is a sequence of edges which represent DSs denoted as: $SearchPlan=\{ StateNode_1, StateNode_2, \ldots\ldots, StateNode_i, \ldots\ldots, StateNode_n \}$. The conditions satisfied by query plan are as follows:

1. There is a path between adjacent nodes;
2. The first node in the sequence is the source node of the DS relationship graph;
3. The last node in the sequence should guarantee the every attributes in metadata schema are true, so the last node represents $StateNode_n(final, -1)$.

```
PQ-Dijkstra(PQ)
// Input: PQ; Output: path is the nodes of query plan.
PQ = new Priority Queue;

for each v ∈V
   do  dist[v] = •;  path[v] = -1; dist[s] = 0;
    // array path is initialized.

for each v ∈V
    do Insert(v,dist[v]) into PQ;
       while(PQ is not empty)
         do   u = getMin(PQ);
              for each neighbor v of u
              do  w = weight of edge(u,v);
                newLen = dist[u] + w;
                   if (newLen < dist[v])
                    then   decreaseKey(PQ,v,newLen);
                          dist[v] = newLen; path[v] = u;
```

The path with the minimum weight in the graph is the optimal query plan. An algorithm based *Dijkstra* algorithm with a set S in which all DSs are sorted by weight (sum of weights from source to current vertex) is proposed, so that the elements with minimum weight always can be found in constant time. The time complexity of *PQ-Dijkstra* is $O((V*logV+E)$.

4 Dynamic and Adaptive Adjustment for Query Plan Execution

Query plan execution may fail owing to the dynamic nature of DS. When a DS becomes un-accessible since it is closed or removed, or its domain name or IP has been changed, or connection timeout led by network traffic. A threshold can be set to judge whether connection to a DS is timeout. When layout of a DS page is changed but its page template has not been altered in time, it is impossible to get valid data from page. It is unreasonable to construct query plan according to quality of DS only evaluated by statistics history of queries executed on it, query plan should be adjusted dynamically. However, changes are unpredictable and uses' queries may be submitted to system at any time, a DS candidate shall be selected or a new query path be taken instead to guarantee the quality of service.

Adjustment of a query plan is implemented by rolling back. An array named *SearchedNode* is maintained in which data status attained successfully are stored, also a link list of metadata is maintained in which each element contains multiple metadata attributes. To efficiently utilize the data already obtained, we will only mark them released but not remove them actually when rolling back. So a field named mark reflecting the validity of data is used. The following process will be executed when the execution of a query path is failed:

Step 1：Check if there is a candidate for the DS (i.e. edge) in the *VDS* the path; if no, remove the edge and delete the failure node to ensure all the final node will be reached from all adjacent nodes;

Step 2：Get nodes in *SearchedNode,* and recalculate the distance from each node in array to final node according to query plan modified, select the nearest node *v* to final node and remove nodes whose distance to final node is infinity;

Step 3：Roll back metadata obtained according to states values, to keep the data consistent with the states of node v mark metadata *released*;

Step 4：Construct query path from v to final node and return.

The re-execution of search can be avoided with the released mark when query plan is modified, and attributes of metadata stored is only set *released* to improve efficiency.

5 Wrapper Manager

To evaluate the performance of our combination query approach, we modify our WDMS prototype TMusic for cross-media search, include 19 well-known musical SEs or portals as data sources, such as soso.com, www.9sky.com, mp3.baidu.com,

sogo.com and etc. A table is created to manage these DSs uniformly (listed in Table 1.), extractor is also prepared for each data source.

Wrapper Manager (illustrated in **Fig.4.**) integrates all of DSs uniformly, it maintains a wrapper selector and indexes DSs on *id*. It is convenient to call corresponding DS wrapper with its *id* directly after the logical layer of program get the sequence of DS ids.

Table 1. List of data sources in modified prototype TMusic

ID	Name	Global_ input	Global_ output	Local_ input	Local_ output
0	9sky	-1	7	{album}	NULL
1	Baidu	24	-1	NULL	{album,0}
2	Soso	24	-1	NULL	{album 0}
9	Sogou	8	16	NULL	NULL
10	Sogou	24	7	NULL	NULL
11	Sogou	4	24	NULL	NULL
12	BaiduMp3	4	24	NULL	NULL
13	BaiduMp3	24	4	NULL	NULL
14	BaiduMp3	16	8	NULL	NULL
15	SosoMp3	8	16	NULL	{album,20}
16	SosoMp3	4	24	NULL	NULL
17	SosoMp3	16	-1	NULL	{singer,18}
18	SosoMp3	-1	8	{singer}	{album,20}
19	SosoMp3	2	25	NULL	{album,20}
20	SosoMp3	-1	28	{album}	NULL
21	9sky	2	25	NULL	{album,0}
23	9sky	1	24	NULL	{album,0}
24	BaiduMp3	20	8	NULL	NULL
25	SosoMp3	20	8	NULL	NULL

The content in a Web page includes two kinds of data:

Data listed are usually in similar style, and they are generated dynamically and gathered in a DIV block or TABLE;

- Isolated data are usually irregular, may exist anywhere.

These two kinds of data are labeled with CTAG and STAG in page template. With the uniform labels added, a generic page analysis can be designed easily.

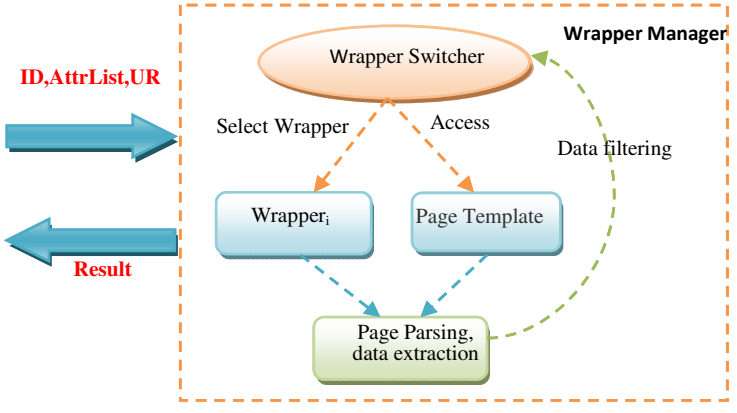


Fig. 4. Wrapper Manager

6 Related Work

User's requirement may not be accepted by every single data source, but the output of one source may be used as the input of another source. [5] focuses on analysis of the retrieval capability of complex query interfaces automatically, models each DS with atom query composed of a set of attributes submitted to DS to get data efficiently to show its query capability. [6] suggests an approach to combine DSs based on DS descriptions, user's requirement is modeled with relational model, then it is serialized and mapped to local schema of a DS which is constructed based on its description. [7] proposes an algorithm to generate query plan based on schema mapping with the correct definition of rules. [8] describes an approach to create query plan automatically according to dependencies between deep web sources, the algorithm to create Top-K query plan is proposed. Our combination query can be applied in MSE with more clear purpose compared to other methods. It supports combination of massive data sources with virtual data sources. User's requirement can be satisfied with the dynamic and adaptive adjustment mechanism for execution of query plan.

7 Conclusion

The purpose of our combination query is more obvious than other combination query, query combining VDSs supports search on enormous DSs, also supports dynamic and adaptive adjustment for query execution to guarantee user's requirement to be satisfied. User experience of searching with TMusic has been greatly improved after combination query over sources proposed in this paper is introduced into the prototype which combines meta-search and social media.

References

1. Selberg, E., Etzioni, O.: Multi-Service Search and Comparison using the MetaCrawler. In: WWW 1995 (1995)
2. Liu, K.-L., Meng, W., Qiu, J., Yu, C.T., Raghavan, V., Wu, Z., Lu, Y., He, H., Zhao, H.: AllInOneNews: development and evaluation of a large-scale news metasearch engine. In: SIGMOD 2007, pp. 1017–1028 (2007)
3. List of major active social networking websites, http://en.wikipedia.org/wiki/List_of_social_networking_websites
4. Peng, Z., Wang, H., Peng, Y., Xu, B., Huang, Z.: A Three Layer System Architecture for Web-Based Unstructured Data Management. In: APWeb, pp. 447–450 (2010)
5. Shu, L., Meng, W., He, H., Yu, C.: Querying Capability Modeling and Construction of Deep Web Sources. In: Benatallah, B., Casati, F., Georgakopoulos, D., Bartolini, C., Sadiq, W., Godart, C. (eds.) WISE 2007. LNCS, vol. 4831, pp. 13–25. Springer, Heidelberg (2007)
6. Levy, A., Rajaraman, A., Ordille, J.: Querying Heterogeneous Information Sources Using Source Descriptions. In: VLDB, pp. 251–262 (1996)
7. Li, Y., Liu, D., Zhang, W.: A Query Discovery Algorithm Based on Schema Mapping. Computer Science (2006) (in Chinese)
8. Gu, Z., Li, J., Xu, B.: Automatic Service Composition Based on Enhanced Service Dependency Graph. In: ICWS 2008 (2008)

Detecting Opinion Leader Dynamically in Chinese News Comments*

Kaisong Song[1], Daling Wang[1,2], Shi Feng[1,2], and Ge Yu[1,2]

[1] School of Information Science and Engineering, Northeastern University
[2] Key Laboratory of Medical Image Computing (Northeastern University), Ministry of Education, Shenyang 110819, P.R. China
songkaisongabc@126.com,
{wangdaling,fengshi,yuge}@ise.neu.edu.cn

Abstract. Nowadays, more and more users publish their opinions by means of Web 2.0. Analyzing users' opinion, discovering the relationship between opinions, and detecting opinion leader are important for Web public opinion analysis. Opinion leader is regarded as the most influential comment and user during the information dissemination process. However, most existing researches pay less attention on their internal relation, implicit relationship between users' opinions and the changes of opinion leader over time. In this paper, we focus on modeling comment network with explicit and implicit links for detecting the most influential comment dynamically, and modeling user network and clustering users for detecting the most influential user. We propose an approach with sentiment analysis, explicit and implicit link mining for modeling comment network in Chinese news comments. We also propose an algorithm for detecting most influential comment from the comment network dynamically. Moreover, we model user network based on the comments network, and detect the most influential user from the user network. Experiments using Chinese Sina news comment dataset show that our approach can detect opinion leaders and the changes of them over time dynamically.

Keywords: Opinion mining, opinion leader, comment network, sentiment analysis, clustering.

1 Introduction

In recent years, with the advent of Web 2.0, more and more users publish their opinions about social events and phenomena by means of tools such as blogs, microblogs, and forums, among which news comments contain rich public opinions. In these opinions, opinion leaders often provide constructive information for other users, and play a critical role during the information dissemination process. Analyzing

* Project supported by the State Key Development Program for Basic Research of China (Grant No. 2011CB302200-G) and National Natural Science Foundation of China (Grant No. 60973019, 61100026).

L. Wang et al. (Eds.): WAIM 2011 Workshops, LNCS 7142, pp. 197–209, 2012.

users' opinions, discovering the relationship between opinions, especially detecting opinion leaders are important for Web public opinion analysis. The researchers have done some pioneering work for discovering opinion leaders in Web 2.0 media. However, there are still some drawbacks in these researches:

(1) Most existing work pay more attentions on analyzing explicit relationship among users' opinions such as following, replying, but neglecting the implicit ones.

(2) A user's' opinion may be a gradual process and affected to a large extent by previous users. However, most existing methods for detecting opinion leaders have not considered the impact of both time and sentiment.

(3) Few people consider the internal relation between the comment's influence and user's one for detecting the most influential user.

(4) In addition, opinion leader with positive influence is more useful in real life.

Based on the characteristics of news comments, in this paper, we focus on modeling comment network with explicit and implicit links for detecting the most influential comment from the comment network dynamically, and clustering users for the most influential user. We model comment network by sentiment orientation analysis, opinion similarity calculating, explicit and implicit link mining, and other related techniques with opinion mining, and propose an algorithm named Dynamic OpinionRank for detecting the most influential opinion from the comment network. Moreover, we apply DBSCAN [3] to find the most influential user. According to the method proposed above, we detect the opinion leader including user and comment. The experiments using Chinese Sina news comments dataset validate the effectiveness of the proposed dynamical opinion leader detection algorithm.

The rest of the paper is organized as follows. Section 2 introduces the related work. Section 3 describes the problem definition. Section 4 analyzes sentiment orientation and model the comment network. Section 5 detects opinion leader including the most influential comment and user. Section 6 shows our experiment results. Finally Section 7 concludes the research and gives directions for future studies.

2 Related Work

In this paper, our purpose is to detect opinion leader in Web 2.0 social media. In this field, the researchers have done some pioneering work. Xiao et al. [9] propose a LeaderRank algorithm to identify the opinion leaders in BBS, which includes finding the interest user group based on topic content analysis and defining the authority value as the weight of the link between users. They also utilize LeaderRank algorithm to identify opinion leaders based on community discovery and emotion mining methods [10]. Freimut and Carolin [5, 6] present an approach, which initially detects opinions and relationships among forum users, extracts main influential factors for opinion forming in virtual communities, and identifies opinion leaders and analyzes opinion evolvement by social network analysis. Zhou et al. [12] introduce the concept of Opinion Networks and propose OpinionRank algorithm to rank the nodes in an opinion network. Zhai et al. [11] propose interest-field based algorithms to identify

opinion leaders in BBS, which not only take into account of the reply networks' structure but also the users' interest space. Feng and Timon [4] proposes a framework, which builds ontology for a marketing product to identify opinion leaders using the information retrieved from blog content, authors, readers, and their relationships.

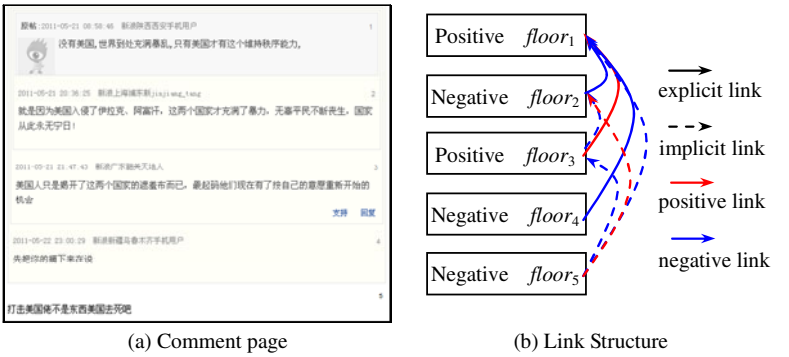
These studies above have some obvious shortcomings. Firstly, few of them do research based on Chinese. Secondly, although some of the works take emotional factors into consideration, they do not discover the impact of other features such as time. To overcome these shortcomings, in this paper we propose a dynamic detection technology for finding opinion leaders from Chinese news comments.

3 Problem Description

Let $C=\{C_1, C_2, \ldots, C_n\}$ be a comment set, and C_i $(1\le i\le n)$ be an item of comment. We can obtain the sentiment orientation O_i $(1\le i\le n)$ for every $C_i\in C$ by sentiment analysis techniques, and the value of O_i is defined as P, N, and M corresponding to positive (support), negative (oppose), and neutral sentiment, respectively. Moreover, we give the following definitions.

Definition 1 (explicit link and implicit link). For C_i and C_j $(1\le i,j\le n)$, suppose C_i be published earlier than C_j. If C_j is a follower or reply of C_i, C_j is regarded as having an explicit link to C_i. If they don't have the relationship, but C_i has semantic similarity with C_j (same or different), C_j is regarded as having an implicit link to C_i.

Definition 2 (positive link and negative link). If C_j has the same sentiment orientation with C_i, the link (explicit or implicit) is called as "positive link", otherwise as "negative link". According to this relationship among comments, we can obtain a link structure about C. Fig.1. shows a comment webpage and its link structure in the webpage.



(a) Comment page (b) Link Structure

Fig. 1. An Example of a Comment Page and Its Linked Structure

In Fig.1, $floor_1$ is the first reviewer, $floor_2$, $floor_3$ and $floor_4$ are the followers of $floor_1$, and $floor_5$ represents user himself. So $floor_2$, $floor_3$ and $floor_4$ are explicitly linked to $floor_1$. According to their content, $floor_5$ is linked implicitly to others.

Definition 3 (comment network and user network). Let E be the set of the links of definition 1 and definition 2 (as edge), and V be the comment set, i.e. $C=\{C_1, C_2, \ldots, C_n\}$ (as vertex), then $G_{CN}(V, E)$ be a comment network (as graph). Moreover, if the vertex in $G_{CN}(V, E)$ is defined with the user who published comments instead of comment, $G_{CN}(V, E)$ will express the relationship among users. We named it as $G_{UN}(V, E)$, i.e. user network. Different form $G_{CN}(V, E)$, the edges in $G_{UN}(V, E)$ are only explicit links.

$G_{CN}(V, E)$ and $G_{UN}(V, E)$ are directed graphs similar to link structure in Fig.1 (b). Difference from Fig.1 (b), the edge in $G_{CN}(V, E)$ or $G_{UN}(V, E)$ has a weight wt, and wt can be calculated according to similarity of two vertexes linking the edge.

Our final purpose is to discover opinion leader including the most influential comment in $G_{CN}(V, E)$ and the most influential user in $G_{UN}(V, E)$, and they may change over time. So we call the process as "detecting opinion leader dynamically".

For this purpose, we will do the following work.

(1) Analyzing sentiment orientation of every comment.
(2) Constructing the link structure as Fig.1 (b).
(3) Modeling $G_{CN}(V, E)$ and $G_{UN}(V, E)$ based on links between comments and time.
(4) Detecting opinion leaders including the most influential comment and user.

Above work will be introduced in Section 4, Section 5 and Section 6 in this paper.

4 Sentiment Analysis and Comment Network Modeling

After downloading related comments about news, we can find explicit links among the comments according to webpage structure. Then we will analyze the content of every comment and calculate the similarity between comments for analyzing their sentiment orientation, finding implicit links, determining positive links and negative links, obtaining the weight of every edge, and finally building the comment network.

4.1 Sentiment Analysis

We utilize HowNet [2] sentiment lexicon for judging sentiment orientation.

Firstly, every comment C_i ($1 \le i \le n$) is partitioned into m sentences i.e. $<S_1, S_2, S_3, \ldots, S_m>$, and apply ICTCLAS [7] to splitting sentence into l words i.e. $<w_1, w_2, w_3 \ldots w_l>$. After that, we extract adjectives, nouns, verbs with sentiment and match them according to the lexicon. Thirdly, we calculate the total number of negation words such as "no", "not". If the number of negation words is odd, the emotional orientation will remain unchanged. If it is even, the sentiment orientation will switch to the opposite. Finally, we get the value of sentiment orientation 1, -1 and 0 according to the rules shown below. We describe the algorithm as Algorithm CommentOrientation.

Algorithm CommentOrientation;
Input: C is any C_i;
Output: O as the sentiment orientation of C and $O \in \{P, N, M\}$;
//{P, N, M} corresponds to {positive, negative, neutral}
Description:

1. partition C into $<S_1, S_2, S_3, \ldots, S_m>$; //$S_i$ $(1 \le i \le m)$ is a sentence
2. for each sentence S_i in C;
3. {splitting S_i into $<w_1, w_2, w_3 \ldots w_l>$; //$w_i$ $(1 \le i \le l)$ is a word
4. for every sentiment word $w_{ij} \in <w_{i1}, w_{i2}, w_{i3} \ldots w_{il}>$
5. w_{ij}.value={1, -1, 0}| w_{ij}.orientation={P, N, M };
6. compute sentiment orientation of S_i by $S_i.orientation = \sum_j w_{ij}.value$
7. for negation word set $N=<n_{i1}, n_{i2}, n_{i3} \ldots>$ ($n_{ik} \in <w_{i1}, w_{i2}, w_{i3} \ldots w_{il}>$)
8. if |N| is an odd then S_i.orientation=-1* S_i.orientation;
9. }
10. compute orientation C by $C.orientation = \sum_{i=1}^{m} S_i$;
11. O={P, N, M }|C.orientation={>0, <0, =0};

4.2 Implicit Link Discovery

Through webpage structure, we can find explicit links easily. Compared with explicit links detection, implicit relationships are much more difficult to find out. According to definition 1, we have to analyze content of every comment for obtaining implicit links. Here we use Vector Space Model (VSM) to describe every comment and calculate the similarity between comments to get their implicit links.

In detail, for two comments C_i and C_j, we remove punctuation and eliminate stopwords from C_i and C_j. Then we split C_i and C_j into words respectively, and apply a vector consisting of meaningful characteristic words to describe C_i and C_j. After that, we utilize $wt_{ij}=TF_{ij} \times IDF_i$ to calculate weighting by document frequency and inverse document frequency. Here TF_{ij} can be normalized by using $0.5+0.5 \times (TF_i/MaxTF_i)$. In the end, two formulas $1+\log(TF)$ and $1+\log(N/DF)$ can be utilized for smoothing the results. After that, we set a threshold for judging whether C_i and C_j have implicit link. And if the threshold is satisfied, we say there is an implicit link $C_i \rightarrow C_j$ (if C_j is published earlier than C_i) or $C_j \rightarrow C_i$ (if C_i is published earlier than C_j).

After above process, we can generate all explicit links and implicit links between any C_i and C_j in comment set C.

4.3 Comment Network Modeling

Based on the results of sentiment analysis, explicit and implicit links mining, we can give out positive links and negative links for comment set C according to definition 2 and further build comment network according to definition 3. The process is described as Algorithm CommentNetworkBuild.

Algorithm CommentNetworkBuild;
Input: explicit links and implicit links in C, sentiment orientation O_i of every $C_i \in C$;
Output: $G_{CN}(V, E)$ //Comment Network of C;
Description:

1. for each $C_i \in C$
2. for each $C_j \neq C_i \in C$
3. if (C_i link to C_j) //the link includes explicit and implicit link
4. if C_i has the same sentiment orientation with C_j
5. C_i positive link to C_j;
6. else
7. C_i negative link to C_j;
8. assign weight wt_{ij} for edge $C_i \rightarrow C_j$;

In the Algorithm, Line 8 is for assigning weight for every edge with positive links and negative links. Initially, wt_{ij} is calculated with formula (1).

$$wt_{i,j}=tag \times similarity(C_i, C_j) \ (-1 \leq wt_{i,j} \leq 1) \tag{1}$$

The function $similarity(C_i, C_j)$ represents the similarity between comment C_i and C_j, and tag means the relationships of them. If $C_i \rightarrow C_j$ is positive link, tag=1. If $C_i \rightarrow C_j$ is negative link, tag=-1.

In fact, wt_{ij} can also express influence of C_j to C_i. Each comment is accompanied by the attribute of time, i.e. time of the comment published, and the relationship between two comments is related to time. For example, when we are reading comments, we may prefer to read those comments that published recently rather than the old ones. So we can come to a conclusion that the impact of time should be taken into consideration. As is said above, we propose a model, which is shown in Fig.2 to further explain the idea.

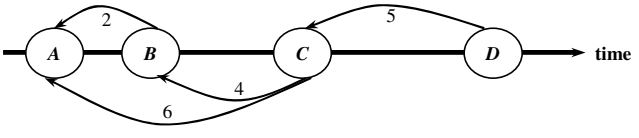


Fig. 2. Time Interval among Comments

As is shown in Fig.2, the former published comments can influence the latter ones. For instance, $B \rightarrow A$ means B is influenced by A. The distance between two comments is their time interval. The larger the distance is, the less possibility the latter will be influenced. For example, the time distance between C and A is larger than the distance between C and B, so we think that C prefers to be influenced by B rather than A.

For comments, the opinion leader is changing over time, so wt_{ij} will concern similarity of C_i with C_j and the time interval between them.

5 Opinion Leader Detection

Opinion leader can be regarded as the most influential comment or its publisher, i.e. user. In this paper, we can detect the most influential comment from $G_{CN}(V, E)$ by computing the score rank of comments, and further build user network $G_{UN}(V, E)$ and discover the most influential user from $G_{UN}(V, E)$.

5.1 The Most Influential Comment Detection

In the field of Web search and Web link analysis, PageRank algorithm [1] is used widely and has excellent effects in many applications. Inspired by PageRank algorithm, we propose a new random walk model called Dynamic OpinionRank, which take emotional and temporal features into consideration. In the meanwhile, a concept of opinion similarity has been proposed and the linked relationships among comments are completely different from the original page linked relationships.

News comments are usually displayed in the form of paging technique. When a user publishes comment, he may be influenced by other comments on two kinds of browsing patterns. Firstly, if the user browses comments backwards and forwards randomly, he will be affected by random comments. Secondly, if he is interested in a certain topic, he will focus on it and may search for relevant comments. That is to say, a user may be influenced by a fixed topic and expresses positive or negative opinions towards relevant comments. So $G_{CN}(V, E)$ may transfer at a certain probability.

Considering users' general browsing habits, our algorithm is based on a reasonable hypothesis that users like referring to the newly published opinions. If the released time of a comment is far from now, the user's comment has less probability to be influenced. Similarly, comments may have two probabilities below to be influenced:

(1) They will be influenced by a relevant topic in a probability f to some purpose.
(2) They may have a probability 1-f to be influenced randomly.

The definition of f is shown as formula (2).

$$f(t_1, t_2, D) = D^{\frac{|t_2 - t_1| \times K}{60 \times 1000 \times 60}} \tag{2}$$

According to the formula above, f is a function relevant to damping coefficient D, t_1 and t_2 represent published time (with the unit of microseconds) of former and latter comment respectively, so f is an alterable damping coefficient. If the cited comment is far from now, it has less probability to be visited. K is a control parameter and we initialize it to 2. D is a damping coefficient which is unrelated to time and we initialize it to 0.85. When a comment is influenced explicitly, it will be less influenced as $|t_2-t_1|$ becomes bigger. Otherwise it is influenced more deeply. If $|t_2-t_1|<1/K$, f is bigger than D. If $|t_2-t_1|>1/K$, f is smaller than D. And if there is no relationship, f is D. Based on the above theory, a promoted model which is similar to PageRank has been proposed as formula (3).

$$P = \left[(1 - f(t_1, t_2, D)) \frac{E}{n} + f(t_1, t_2, D) A^T \right] P \tag{3}$$

where A is an $n \times n$ matrix which is shown in formula (4) and n represents the total number of comments. The matrix reflects the mutual influences among comments, and a_{ij} means the comment i is influenced by the comment j.

$$A^T = \begin{bmatrix} a_{11} & . & . & a_{n1} \\ . & . & . & . \\ . & . & . & . \\ a_{1n} & . & . & a_{nn} \end{bmatrix} \tag{4}$$

As for $a_{ij}=wt_{ij}/OD_i$, OD_i represents out-degree of comment i (i.e. C_i). As comments do not always have a large number of out-degrees and in-degrees in comparison with Web page link analysis, matrix A may be not a transfer matrix. If a node only has in-degree but has not out-degree, OD_i will be 0. We adopt general processing method similar to PageRank, but double the result after taking sentiment features into consideration. And we use the formula $a_{ij}=(1/n)^2$ to initialize the line of a_{ij}.

We utilize the formula (5) for calculating final scores of C_i.

$$P(i) = \sum_{1 \le k \le n} \frac{(1 - f(t_i, t_k, D))}{n} + \sum_{(j,i) \in E} f(t_i, t_j, D) \frac{P(j) \times wt_{ji}}{OD_j} \tag{5}$$

where $P(i)$ presents authority value. If we follow the above methods, we will get the final ranking score of each comment in a period of time. Afterwards, we will select the top comment with the highest score as the opinion leader. According to the above operation, a comment network diagram can be modeled easily. Only if a vertex gets more in-degree, which means the comment expresses consistent emotional orientation with others, and time interval is not too long, the comment may get a higher ranking score. The Dynamic OpinionRank algorithm is given below.

Algorithm Dynamic OpinionRank;
Input: $G_{CN}(V, E)$; parameter D and K; threshold ε;
Output: PR //$PR[i]$ is ranking score of comment C_i;
Description:

```
1. compute matrix A from G_CN(V, E);
2. Repeat
3.   for i=1 to n
4.     {for all j satisfying a_ij≠0 compute f(t_i, t_j, D) using formula (2);
5.      compute P(i) using formula (5);
6.     }
7. Until |current P(i)-last P(i)|≤ε;
```

5.2 The Most Influential User Detection

According to the explicit relationships among comments, we can build up $G_{UN}(V, E)$ to express the relationship among users. If a user replies to another user, then there

will be a link between the two users. In $G_{UN}(V, E)$, we can calculate Degree Centrality and Proximity Prestige [8], and get Comment Quality from the ranking score calculated by Dynamic OpinionRank algorithm proposed above. They will be used to detect the influence of a user.

In [8], Degree Centrality and Proximity Prestige are calculated by formula (6) and formula (7) respectively, and reflect the out-degree and range of user *i*.

$$DC(i) = OD_i / (n-1) \tag{6}$$

$$PP(i) = (|I_i|/(n-1)) \Big/ \left(\sum_{i \in I_i} d(j,i) / |I_i|\right) \tag{7}$$

where I_i is the set linked to vertex (user) *i*, OD_i is the out-degree of user *i* (the same definition with comment), and $d(j, i)$ is the shortest link length between *i* and *j*. formula (8) is used for getting Comment Quality, and shows influence of comment.

$$CQ(i) = \frac{\sum_{j \in UC_i} Score(j)}{|UC_i|} \times \frac{CL(i)}{CML} \tag{8}$$

As for formula (8), UC_i represents the set of comments published by user *i*, $Score(j)$ is ranking score of comment *j* (from Dynamic OpinionRank), $CL(i)$ is the mean value of all comment lengths of user *i*, and *CML* is the max length of all users' comments.

Each user is described by $PP(i)$, $CQ(i)$, and $DC(i)$, and the vector represents a point in the three dimensional space. The method below is based on the hypothesis that if a user has strong influence, it will be away from the other points. So we use DBSCAN [3] algorithm to cluster these points, and the outliers may be the most influential users. We set reasonable radius and MinPts for clustering, and then use the formula (9) below to detect the most influential user from the outliers.

$$S(i) = (w_{PP} \times PP(i) + w_{CQ} \times CQ(i) + w_{DC} \times DC(i)) / 3 \tag{9}$$

where w_{PP}, w_{CQ}, and w_{DC} represent the weight of $PP(i)$, $CQ(i)$, and $DC(i)$, respectively.

6 Experiment Results

In order to get a real dataset, we collect comments from four different period of time. The dataset is about "Libya's civil war". Based on the purpose, we collect real comments about this title from Sina news forum (http://comment4.news.sina.com.cn/ comment/skin/ default.html?channel=gj&newsid=1-1-23003128&style=0#page=10) during two days (2011-08-17 08:04:37~2011-08-18 06:47:08).

6.1 Opinion Leader Detection from Comments

By building $G_{CN}(V, E)$, we apply the most influential comment detection method in Section 5.1, the ranking scores of comments are shown in Fig.3 and Fig.4.

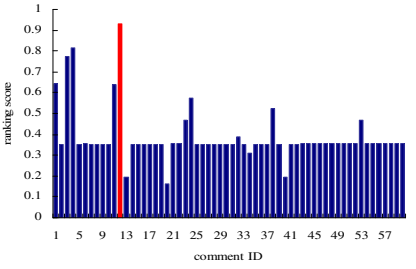


Fig. 3. Ranking (08:04:07-11:07:11)

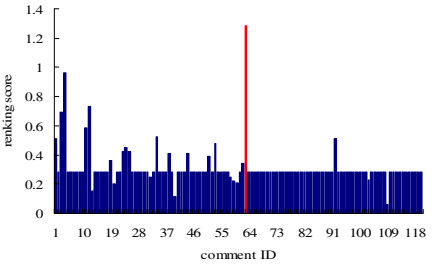


Fig. 4. Ranking (08:04:07-13:45:49)

In Fig.3, a total number of 60 comments are in this period of time. According to the diagram, it is easy to find that NO.12 gets the highest score. However, as there is much less mutual influence between the comments, the opinion leader may change in all probability as time goes on.

From the Fig.4, we can find that NO.12 is not the opinion leader, and NO.63 gets the highest scores, so it is the opinion leader now. Some comments get low value because they are opposed by the other users. Though, there are more comments, mutual influence is still not too much.

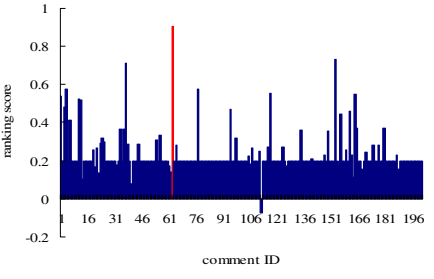


Fig. 5. Ranking (08:04:07-18:43:22)

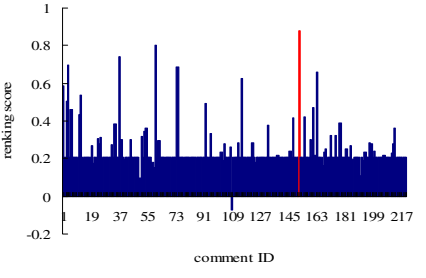


Fig. 6. Ranking (08:04:07-next day 06:47:08)

In the third period of time of Fig.5, NO.63 continues to be the opinion leader, and his scores are decreasing as the time goes on. With the increasing number of comments, newly published comment is increasing much faster, and there is much more mutual influence between comments. This figure shows new comments attract more attention, and it also proves the effectiveness of taking time into consideration.

In the Fig.6, after tracking a day of the news forum, the number of comments reaches to 220. Now NO.151 gets the highest score, and it is the opinion leader. Compared with the Fig.5, many comments' scores are increasing to a certain degree. The number of comments keeps stability in a certain range because of its real-time characteristic, and No.151 is probably the opinion leader in the end.

The most influential comment and ranking score are shown in Table 1 below.

Table 1. The Changes of Opinion Leader in the Comments

No	Content of Comment	Ranking Score
No.12	政府军打击不准确,无非是不伤害平民百姓的	**0**.9312
No.63	和当年萨达姆政权的政府发言人如出一辙！	**1**.2886
No.63	和当年萨达姆政权的政府发言人如出一辙！	**0**.905
No.151	民主是用导弹和战机换来的吗？民主是用导弹和战机换来的吗？民主是用导弹和战机换来的吗？	**0**.8772

From the Table 1 above, we can find the opinion leader may change over the time. And opinion leader's ranking is influenced by time. That's the reason why we introduce the feature of time in our proposed algorithms.

In order to evaluate the performance of our proposed Dynamic OpinionRank, we construct a standard by letting expert divide all comments of each period into two categories: strong influence and weak influence. We compare our method with published time, Degree Centrality, Proximity Prestige and OpinionRank proposed in [12] by calculating *F-Score*. The comparison result is shown in Fig.7 below.

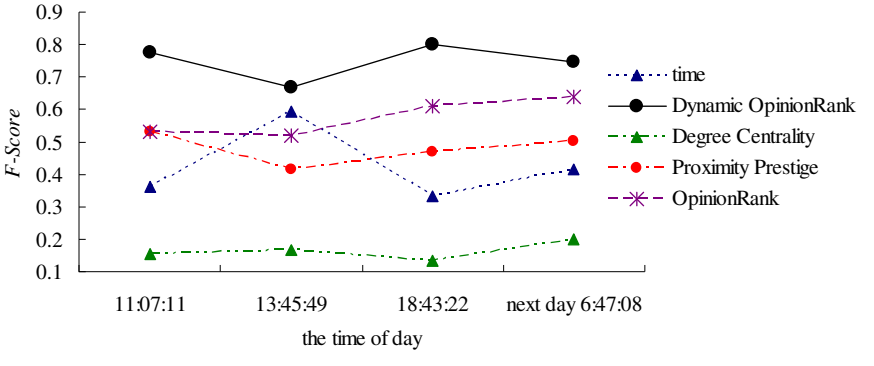


Fig. 7. F-Score of Each Method

From Fig.7, we can find Dynamic OpinionRank has a high accuracy and stability compared with other methods, and it proves the effectiveness of our proposed method.

6.2 Opinion Leader Detection from Users

We apply the method proposed in Section 5.2 for building $G_{UN}(V, E)$ and detecting most influential user. For DBSCAN clustering algorithm, we set radius ranges from

0.06 to 0.12, and initialize MinPts to 1 so that some clusters may contain one single user, which means an outlier. We generally get three to five clusters, and clustering result is shown in Table 2 below.

Table 2. Clustering Results for Detection Opinion Leader from Users

08:04:37~11:07:11		08:04:07~13:45:49		08:04:37~18:43:22		08:04:37~6:47:08	
Cluster	User Count	Cluster	User Count	Cluster	User Count	Cluster	User Count
1	43	1	81	1	117	1	129
2	1	2	1	2	3	2	1
3	2	3	1	3	1	3	1
4	1	4	1				
		5	1				
uid	1220398631	**uid**	1151164004	**uid**	1199491660	**uid**	1725406570

From Table 2, we can remove the first cluster from every cluster group, and use other clusters with fewer points to get the most influential user as opinion leader by formula (9) in each period of time. As $G_{UN}(V, E)$ is very sparse, so we set w_{PP}=10, w_{CQ}=0.3, and w_{DC}=10. At last, we can get opinion leader of each period of time which is shown in the last row of Table 2. According to the experiment, we detect both the most influential comment and user as opinion leader, we also draw a conclusion that the opinion leader may change dynamically as the time goes.

7 Conclusions and Future Work

In this paper, we focus on Chinese news comments, and utilize the techniques such as sentiment orientation analysis, and opinion mining for modeling comment network with explicit and implicit links, and further generating user network. Based on the comment network, we propose an algorithm called Dynamic OpinionRank, for detecting the most influential comment as opinion leader in Chinese news comments. Different from related work, the proposed comment network model considers not only explicit but implicit links, and Dynamic OpinionRank algorithm can detect the most influential comment and track their changes over the time. Moreover, we detect the most influential user in the user network by clustering algorithm according to internal relations between comment and user, and the ranking scores of the comments.

Although this paper puts forward many actual effective methods, there are still some places needed in the later work to improve. As the implicit relationship beyond the context content is difficult to define, there is still no way to find their connections effectively. In addition, combining emotional analysis and natural language grammar characteristics may improve the final accuracy greatly. The other implicit influential features need to be detected. All of them are our future research topics.

References

1. Brin, S., Page, L.: The Anatomy of A Large-Scale Hypertextual Web Search Engine. In: WWW-7 (1998)
2. Dong, Z., Dong, Q.: HowNet (2003), http://www.keenage.com/html/e_index.html
3. Ester, M., Kriegel, H., Sander, J., Xu, X.: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: KDD 1996, pp. 226–231 (1996)
4. Feng, L., Timon, C.: Who is talking? An ontology-based opinion leader identification framework for word-of-mouth marketing in online social blogs. Decision Support Systems (DSS) 51(1), 190–197 (2011)
5. Freimut, B., Carolin, K.: Detecting Opinion Leaders and Trends in Online Communities. In: ICDS 2010, pp. 124–129 (2010)
6. Freimut, B., Carolin, K.: Detecting opinion leaders and trends in online social networks. In: CIKM-SWSM, pp. 65–68 (2009)
7. Golaxy: ICTCLAS, http://www.ictclas.org
8. Liu, B.: Web Data Mining: Exploring, Hyperlinks, Contents, and Usage Data. Springer, Heidelberg (2007)
9. Yu, X., Wei, X., Lin, X.: Algorithms of BBS Opinion Leader Mining Based on Sentiment Analysis. In: Wang, F.L., Gong, Z., Luo, X., Lei, J. (eds.) WISM 2010. LNCS, vol. 6318, pp. 360–369. Springer, Heidelberg (2010)
10. Xiao, Y., Xia, L.: Understanding opinion leaders in bulletin board systems: Structures and algorithms. In: LCN 2010, pp. 1062–1067 (2010)
11. Zhai, Z., Xu, H., Jia, P.: Identifying Opinion Leaders in BBS. In: Web Intelligence/IAT Workshops 2008, pp. 398–401 (2008)
12. Zhou, H., Zeng, D., Zhang, C.: Finding leaders from opinion networks. In: ISI 2009, pp. 266–268 (2009)

An Approach of Semi-automatic Public Sentiment Analysis for Opinion and District*

Daling Wang[1,2], Shi Feng[1,2], Chao Yan[1], and Ge Yu[1,2]

[1] School of Information Science and Engineering, Northeastern University
[2] Key Laboratory of Medical Image Computing (Northeastern University), Ministry of Education, Shenyang 110819, P.R. China
{wangdaling,fengshi,yuge}@ise.neu.edu.cn, lavenderchao@126.com

Abstract. The contents generated by netizens on the Web can reflect public sentiments to a great extent, so analyzing these contents is very useful for government agencies in guiding their public information, propaganda programs, and decision support. Because of the civilization diversity and economy difference, the netizens inhabiting or employing in different districts may have the different sentiments for the same topic or event. Analyzing the sentiment difference of different districts will help government agencies make more pertinent decision. However, current researches in this domain have less considered the opinion distribution on different districts. In this paper, we propose an approach of semi-automatic public sentiment analysis for opinion and district, which includes automatic data acquiring, sentiment modeling, opinion clustering, and district clustering, and manual threshold setting and result analysis. In detail, on the one hand, we group public sentiment into some opinion clusters by means of clustering technique. On the other hand, based on the opinion clusters, we further partition every opinion cluster on district into district opinion and analyze the result. Experiment results in Tencent comments show the feasibility and validity of our approach.

Keywords: Public sentiment analysis, opinion clustering, opinion distribution.

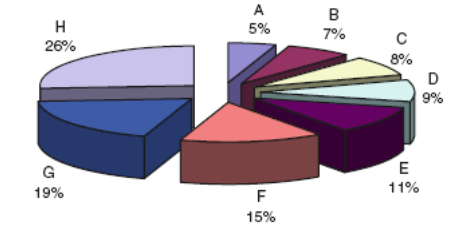
1 Introduction

More and more netizens publish their opinions about social events and phenomenon by means of social media such as blog, microblog, wiki, BBS, IM. The attitudes or beliefs of these netizens can reflect Web public sentiments and current issues to a great extent. For government agencies, analyzing public sentiment is very useful in guiding their public information and propaganda programs, and occasionally for helping in the formulation of other kinds of policies.

* Project supported by the State Key Development Program for Basic Research of China (Grant No. 2011CB302200-G) and National Natural Science Foundation of China (Grant No. 60973019, 61100026).

L. Wang et al. (Eds.): WAIM 2011 Workshops, LNCS 7142, pp. 210–222, 2012.

Public sentiment is the aggregate of individual attitudes or beliefs held by the adult population [14]. Current most researches classify public sentiment into positive, negative, and neutral category according to the sentiment orientation of netizens' opinion, some researches consider more exquisite sentiment such as Fig.1 [3].



Clusters	Common emotion words
A	"期望" (expectation) ,"希望"(hope), "接受"(accept)
B	"痛苦的"(painful), "心痛"(broken-hearted), "责怪"(blame)
C	"赞同"(approve), "谢谢"(thank), "陶醉"(intoxicated)
D	"失误"(mistake), "可怜"(pathetic), "伤害"(hurt)
E	"眼泪"(tear), "没想到"(unexpected), "危机"(crisis)
F	"相信"(believe), "了解"(understandable), "勇敢的"(brave)
G	"失望"(disappointed), "受伤的"(injured), "无奈"(helpless)
H	"遗憾"(regretful), "慰问"(console), "郁闷"(depressed)

Fig. 1. An Example of Opinion Clustering for More Exquisite Sentiment [3]

In many cases, public sentiments are relative to district, status, age, and sex of the netizens. For example, for the news of "Copenhagen Climate Conference 2009", the netizens from developed countries may have different opinions with ones from developing countries. For the comments of "Chinese physician-patient relationship reconsidered why nervous", physicians may have different opinions with patients due to different perspective. For the content about property in "New marriage law of China", male netizens may have different opinions with female netizens. Even senior citizens are different from younglings about the law. However, current researches in this domain have less considered the opinion distribution on different districts, status, age, and sex of netizens. The reason may be that netizens always publish their opinion in anonymous login, so their above information can not be known.

In fact, some websites can show the district of netizens according to their login information, and they even ask netizens to login using real-name. Meantime, we think that analyzing the opinion distribution of different districts will help government agencies make more pertinent decision. Based on this idea, in this paper, we apply the more exquisite and particular sentiment analysis in [3] but improve the sentiment representing model, and further consider the opinion distribution in different districts. For this purpose, we propose an approach of semi-automatic public sentiment analysis for opinion and district, which includes automatic data acquiring, comment modeling, opinion clustering, and district clustering, and manual threshold setting and result analysis. We crawl and download the comments from Tencent for experiments, and the results show the feasibility and validity of our approach.

The rest of the paper is organized as follows. Section 2 introduces the related work. Section 3 describes our approach. Section 4 designs our comment represent model and the clustering algorithm on opinion. Section 5 presents the method of concept climbing for district and sentiment distribution discovery for districts. Section 6 shows our experiment results and analyzes them. Finally, Section 7 concludes the research and gives directions for future studies.

2 Related Work

Our approach is based on sentiment, and the main platform for netizens to express their sentiment is the texts in social media such as blog, microblog, BBS, and IM, so text sentiment analysis is the foundation of our work. The text sentiment analysis mainly includes subjectivity classification, word sentiment classification, document sentiment classification and opinion extraction [12]. In text sentiment analysis, Melville et al. presented a unified framework for analyzing sentiment of blogs by combining lexical knowledge with text classification [9]. Tan and Zhang studied sentiment categorization on Chinese documents, and investigated four feature selection methods and five learning methods for classifying Chinese sentiment corpus [11]. Ye et al. performed automatic classifications based on the sentiment attitudes of online reviews with regards to travel destinations [16]. Wu and Tan proposed a two-stage framework for cross-domain sentiment classification and improved the performance of cross-domain sentiment classification [15]. Khan, Baharudin, et al. proposed a method for classifying subjective and objective sentences from reviews and blog comments [5].

These researches used different methods for sentiment classification in different documents, but most of them classify sentiment orientation into two categories (positive and negative) or three categories (positive, negative, and neutral). There are some studies about sentiment clustering. Luo et al. employed finer granularity clustering for opinion extraction and the clustering results were further used for the calculation of their sentiment orientation [7]. We ever modeled the hidden emotion factors based on Probabilistic Latent Semantic Analysis (PLSA) and clustered Chinese blogs according to the sentiment similarities between them [3].

In this paper, we also apply the clustering technique for acquiring more exquisite and particular public sentiment. Different from above work, we use LDA [1] model to represent the contents generated by netizens and deduce sentiment factors for opinion clustering. Especially, we further analyze the sentiment distribution in different districts based on opinion clustering.

3 Problem Description

Here we describe our problem on comments. In fact, our approach can also be applied to other user generated content on the Web.

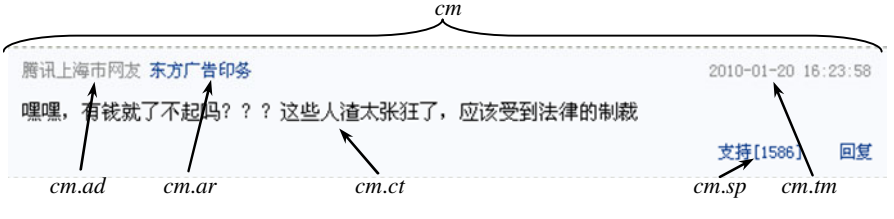


Fig. 2. An Example of Comment Item Structure

Let $CM=\{cm_1, cm_2, \ldots, cm_m\}$ be a comment set and $cm_i=<ar, ad, tm, sp, ct>$ (i=1, 2, …, m) be an item of the set (or a comment), where *ar*, *ad*, *tm*, *sp*, *ct* are author (not always the real name of a netizen), address, published time, support number, and content of the comment item cm_i. We mark them as $cm_i.ar$, $cm_i.ad$, $cm_i.tm$, $cm_i.sp$, and $cm_i.ct$ respectively. We take an example with Tencent comment site (http://comment5.news.qq.com/comment.htm) to explain it, which is shown as Fig.2.

Obviously, comment content *cm.ct* is the most important for sentiment analysis of every comment item, so we will model *cm.ct*, which utilizes a natural language process technique for obtaining sentiments words and their modifiers, and applies LDA-based method for deducing the latent sentiment factors embedded in *cm.ct*. Moreover, *cm.ad* is another important attribute for our work, we will use it to partition public sentiment on district, i.e. discover opinion distribution on district.

For above purposes, we propose an approach of semi-automatic public sentiment analysis for opinion and district. The implementing framework in this paper includes automatic and manual parts for different functions. The framework of our approach is shown as Fig.3.

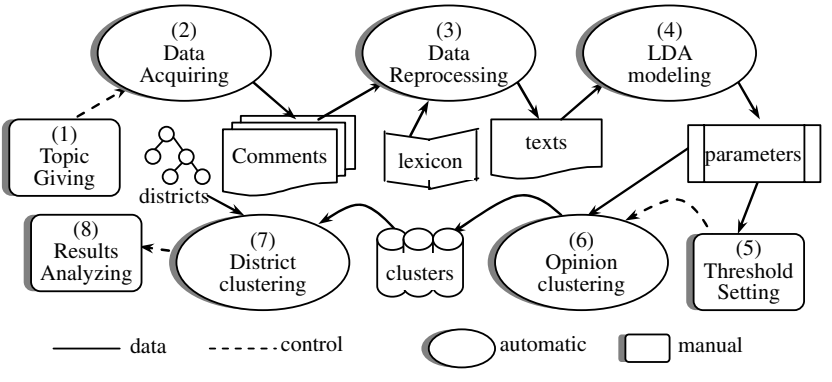


Fig. 3. Framework of Semi-Automatic Public Sentiment Analysis for Opinion and District

In Fig.3, the framework includes eight parts for the following functions.

(1) One or more topics are given by opinion analysts.

(2) For a given topic, related comments are crawled from all kinds of forums and comment sites.

(3) For every comment, related tools and sentiment lexicons such as ICTCLAS [4], HowNet [2] and NTUSD [6] are used for reprocessing, which includes segmenting sentence into words, extracting sentiment words and their modifiers, and storing them in text.

(4) LDA-based method is used to model the comments and deduce sentiment factors and related parameters.

(5) The sentiment factors are analyzed for giving thresholds of clustering algorithm.

(6) Deduced parameters in (4) and given thresholds in (5) are used for opinion clustering.

(7) Based on opinion clusters obtained in (6), further clustering on district is implemented, which uses district concept hierarchy knowledge.

(8) Opinion distribution on different districts is analyzed for decision support.

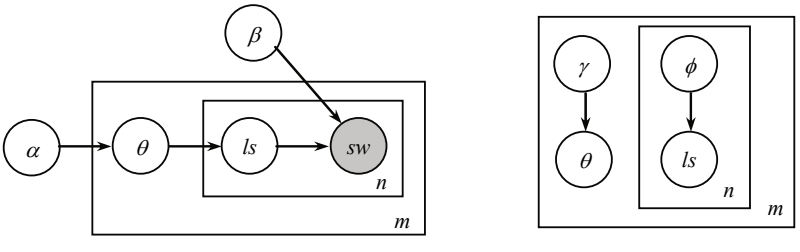
In above work, (1), (5), and (8) are manual, i.e. they are done by users or opinion analysts. These processes mean that for users or opinion analysts, they should present interested topics by themselves, give threshold by means of LDA deducing results, and analyze final sentiment distribution for decision support. (2), (3), (4), (6), and (7) are implemented automatically. So we call the approach as semi-automatic one.

4 Comment Content Modeling and Opinion Clustering

For obtaining sentiment from comments, we apply Chinese text processing tools ICTCLAS [4] for segmenting every comment item content *cm.ct* (abbreviated as *cm*) into words and tagging part of speech for each word. For all contents of $cm \in CM$, we extract sentiment words (adjectives, adverbs, and some of verbs) and their modifiers by means of HowNet [2] and NTUSD [6] sentiment lexicon. Finally we obtain a sentiment word set $SW=\{sw_1, sw_2, \ldots, sw_n\}$. For every $cm \in CM$, a vector of sentiment word $SV \subseteq SW$ will correspond to it. For all cm_i (i=1, 2, …, m), their sentiment word vectors SV_i (i=1, 2, …, m) will be utilized for modeling the latent sentiment factors using LDA, and further clustering the comments based on embedded opinions.

4.1 Comment Content Modeling Using LDA

Latent Dirichlet allocation (LDA) [1] is a probabilistic generative model, which has been applied recently in many text mining problems especially short reviews [10, 13]. Here we assume that CM is generated using a mixture of v latent sentiment factors $LS = \{ls_1, ls_2, \ldots, ls_v\}$ and utilize LDA model to identify the latent semantic relationships among *CM*, *LS*, and *SW*. The generative model is shown as Fig.4 (*a*).



(*a*) LDA Model (*b*) Simplified LDA Model

Fig. 4. LDA Model and Its simplicity [1]

In Fig.4 (*a*), comment set *CM* is composed with *m* comments $\{cm_1, cm_2, \ldots, cm_i, \ldots, cm_m\}$. cm_i is composed with *n* sentiment words $\{sw_1, sw_2, \ldots, sw_n\}$ and they associate *v* latent sentiment factors $\{ls_1, ls_2, \ldots, ls_v\}$. Based on the relationship, the model is presented with Formula (1).

$$P(\theta, ls \mid sw, \alpha, \beta) = \frac{P(\theta, ls, sw \mid \alpha, \beta)}{P(sw \mid \alpha, \beta)} \tag{1}$$

For estimating α and β, Blei [1] proposed a variational inference approach to simplify the model, which selected a variational distribution *Q* as Fig.4 (*b*) instead of *P* in Fig.4 (*a*) and Formula (2).

$$Q(\theta, ls \mid \gamma, \phi) = Q(\theta \mid \gamma) \prod_{i=1}^{n} Q(ls_i \mid \phi_i) \tag{2}$$

where the Dirichlet parameter γ and the multinomial parameters $(\phi_1, \phi_2, \ldots, \phi_n)$ are the free variational parameters.

Thus the optimizing values of the variational parameters are found by minimizing the Kullback-Leibler (KL) divergence between the variational distribution $Q(\theta, ls|\gamma, \varphi)$ and the true posterior $P(\theta, ls|sw, \alpha, \beta)$. By derivation of the variational EM algorithm, E-step (for finding the optimizing values of the variational parameters γ and φ) and M-step (for maximizing the resulting lower bound on the log likelihood with respect to α and β) are repeated until the result converges.

In Fig.4, θ is the probability of picking a latent sentiment factor *ls* for a comment *cm*, and it represents how much a latent sentiment factor *ls* "contributes" to the *cm*. So, we utilize θ to represent $cm \in CM$ for opinion clustering, and here θ can be regarded as $(P(ls_1|cm), P(ls_2|cm), \ldots, P(ls_v|cm))$.

4.2 Opinion Clustering Using *k*-Means

According to the sentiment expressed by content of each *cm* (i.e. *cm.ct*), *CM* can be group into *k* clusters $cl_1, cl_2, \ldots, cl_k$ and they compose cluster set *CL*. Because the *cm*s in the same cluster have similar opinions, we call each cluster as an opinion cluster.

For any opinion cluster $cl \in CL$, without losing of generality, we suppose $\{cm_1, cm_2, \ldots, cm_l\} \in cl$ ($l \le k$) and $SentiSim(x, y)$ means the sentiment similarity of comment item content *x* and *y*. Then, for any $i \le l$, $j \le l$, and $k > l$, it satisfies:

(1) $SentiSim(cm_i, cm_j) > SentiSim(cm_i, cm_k)$ and
(2) $SentiSim(cm_j, cm_i) > SentiSim(cm_j, cm_k)$.

Above statement means that every cluster is a collection of comment items that contents have similar opinions to one another within the same cluster and have dissimilar opinions to the comment items in the other clusters. This is the same as traditional clustering techniques, but here "similarity" means the same or similar opinion embedded in comment contents.

After modeling comment content, for all $cm \in CM$, *cm* can be represented as $cm = <ar, ad, tm, sp, <P(ls_1|cm), P(ls_2|cm), \ldots, P(ls_v|cm)>>$, so we apply $<P(ls_1|cm),$

$P(ls_2|cm)$, …, $P(ls_v|cm)>$ as the factor vector to present *cm.ct* and use $SentSim(cm_i, cm_j)$ to measure the similarity between two *cm*s as Formula (3).

$$SentSim(\vec{d_i}, \vec{d_j}) = \frac{\vec{d_i} \cdot \vec{d_j}}{\left\|\vec{d_i}\right\| \times \left\|\vec{d_j}\right\|} \tag{3}$$

where $\vec{d_i} \cdot \vec{d_j} = \sum_{m=1}^{v} P(ls_m \mid cm_i) P(ls_m \mid cm_j)$ and $\left\|\vec{d_i}\right\| = \sqrt{\sum_{l=1}^{v} P(ls_l \mid cm_i)^2}$

Based on the sentiment similarity, we can apply a clustering algorithm to group *CM* into *k* clusters. We take *k*-Means [8] algorithm as an example and the algorithm of clustering *CM* by *k*-Means is shown as Algorithm 1.

In Algorithm 1, Line 1) is for representing latent sentiment factors (see Section 4.1), and from Line 3)~7) is for clustering by *k*-Means. After clustering, *CM* can be partitioned into *k* clusters, i.e. *k* opinion clusters, based on the opinion similarity between their contents. Line 2) is for giving threshold *k*.

In *k*-Means algorithm, *k* must be given beforehand. Though there are some methods for selecting *k* automatically, but the selection should correspond to the data distribution and application requirement. Here we use manual analysis by means of LDA parameters. In LDA implementation, we can obtain not only the probability of latent sentiment factors (θ vector) but also latent sentiment words (Top 10 feature words of *v* latent sentiment factors). The former is used for similarity measure in clustering, and latter is used for manual analysis. Because $10 \times v$ (feature words) is less than $n \times m$ (all word of *CM*) out and away, the manual analysis is feasible.

Algorithm 1: Clustering *CM* by *k*-Means

Input: comment set *CM*;
Output: opinion cluster set $CL=\{cl_1, cl_2, \ldots, cl_k\}$ // *k* clusters of *CM*;
Method:
1) for all $cm.ct \in CM$, calculate $P(ls|cm)$ using LDA model
and represent *cm.ct* as $<P(ls_1|cm), P(ls_2|cm), \ldots, P(ls_v|cm)>$; //i.e. θ in LDA
2) give *k* by manual analysis; //by means of LDA parameters
3) arbitrarily choose *k cm*s from *CM* as the initial cluster centers;
4) repeat
5) (re)assign each *cm* to the cluster to which the *cm.ct* has the most similar sentiment with the mean value of the *cm.ct* in the cluster based on Formula (3);
6) update the cluster means, recalculate the mean value of the *cm.ct* for each cluster;
7) until no change;

5 Opinion Distribution Discovery Based on District

As mentioned in Section 1, some public sentiments are relative to the district of their publishers. For some special opinions, discovering the district distribution of netizens who hold the opinions is useful for government and opinion analysts to understand public opinions.

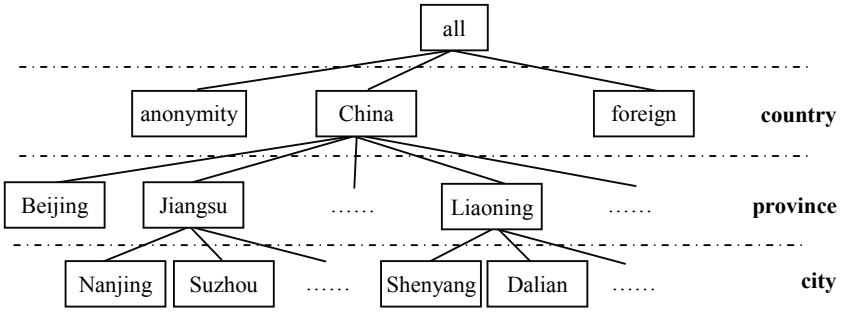


Fig. 5. Concept Hierarchy of China District

In Section 3, we describe a comment item *cm*=<*ar*, *ad*, *tm*, *sp*, *ct*>, where *ad* is the address or district of the item publisher. We can obtain the district distribution of an opinion according to *ad*. In Web comments, the address can be a city, a province, or anonymous. For unifying the address, we create a concept hierarchy tree as district domain knowledge. A China district concept hierarchy tree is shown as Fig.5.

The concept hierarchy tree can be maintained by offline handwork or online interaction. For an opinion cluster, based on the concept hierarchy tree in Fig.5, we can unify the address to province or other levels and group them by district. Algorithm 2 describes the process method. In the algorithm, the district is unified to province and the concept hierarchy tree is maintained by online interaction.

Algorithm 2: Discovering district distribution for opinion

Input: an opinion cluster *cl*, district concept hierarchy tree R-tree;
Output: district distribution of *cl*;
Method:
1) for all *cm*∈*cl*
2) {if *cm.ad* is in R-tree
3) if *cm.ad* is not in province level then climb *cm.ad* to province;
4) else accept *cm.ad* by dialog box; //man-machine conversation
5) }; cluster *cm* by *cm.ad*;

In the algorithm, Line 1)~4) can unity *cm.ad* to the same level. In this case, *cm.ad* is regarded as the feature of *cm*, and clustering all *cm*∈*cl* in Line 5) is just grouping them by *cm.ad*, i.e. merging the *cm*s with the same *ad* to a cluster. Here the similarity between two comments means they have the same *cm.ad*.

The result can show the district distribution of every public sentiment. However, the reason why we can obtain such opinion cluster and district distribution needs to be analyzed by opinion analysts. The analysis may require more humanities and geography knowledge. Meantime, the concept hierarchy needs to be built beforehand.

6 Experiment and Analysis

We collect the Tencent comments concerning "春节晚会节目低俗 (Spring Festival Evening Show is Vulgar)" (http://comment5.ent.qq.com/comment.htm?site=ent&id=22881311) by constructing URL parameters, downloading source codes, and parsing the source codes. The number of valid comment is 7438.

6.1 Opinion Clustering Experiment

We reprocess and model the comments using the process in Section 4.1. In LDA implementation, we apply GibbsLDA++ program and let v=10. Table 1 gives 10 sentiment factors and top 10 words of each factor in above comment set.

By analyzing the top 10 words of 10 sentiment factors in Table 1, we think it should concern 4 aspects about "spring festival evening show itself" (factor 1 and 3), "northeast couple dance opera" (factor 2, 4, and 8), "elegance and vulgarity" (factor 5, 6, and 7), and "approval or opposition" (factor 9 and 10), respectively. So we set k=4 for k-Means clustering. Moreover, we extract a sentiment word and its modifier words as the "label" or typical opinion of every cluster. The result is shown as Fig.6.

Table 1. Top10 Sentiment Factors and Top 10 Words of Each Factor

Sentiment factor	Top 10 words
1	央视 春晚 太 赵 现在 台 成 问题 应该 已经
2	二人转 喜欢 现在 人 知道 好 观众 艺术 东西 地
3	春晚 节目 年 广告 一年 太 确实 导演 越来越 感
4	赵本山 小品 小 真 演 笑 点 沈阳 今年 最
5	高雅 艺术 水平 老百姓 低俗 欣赏 晚会 高 应该
6	低俗 好 全国 顶 搞 人 委员 金铁霖 政协 垃圾
7	文化 中国 农民 俗 只 种 社会 文艺 代表 民族
8	东北 东北人 钱 人 南方 吃 北方 南方人 真 北方
9	人 想 懂 一个 中国 听 觉得 只 骂 再
10	说 支持 话 金 老师 下 请 句 好 看看

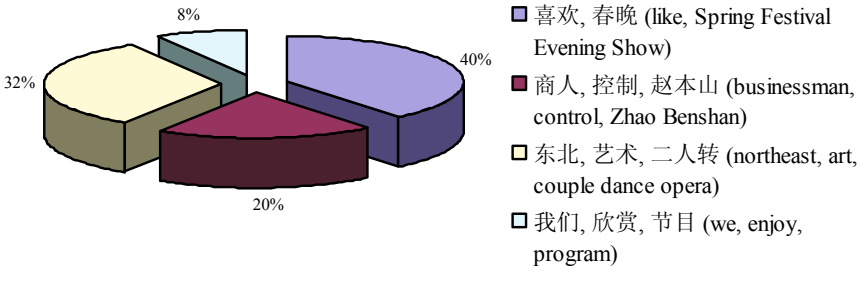


Fig. 6. Clustering Result on Opinion

Here we set v=10 using LDA model and obtain Table 1, it results in k=4. From Fig.6, we can see the label of every cluster is not a complete sentence but only some words. How to extract more suitable sentence as label will be our future work.

In fact, if we let v be other value, the result may be changed. How to set correct v value will also be our future work.

6.2 Opinion Distribution on District Experiment

Based on opinion clustering, we further partition every cluster according to district. We regard the process as district clustering. In this case, *cm.ad*, i.e. address of comment publisher is used to represent the comments. In this experiment, it is unified to province. According to the representation, similarity measure between comments becomes their corresponding province. If they have the same province, they will be assigned to the same cluster. The number of comments in each cluster for every province is shown as Table 2.

Table 2. Experiment Results for Opinion Distribution

province	cluster1	cluster2	cluster3	cluster4	total
Beijing	159	71	138	38	406
Shanghai	59	44	54	15	172
Tianjin	35	16	45	11	107
Chongqing	48	25	36	8	117
Hebei	81	36	74	15	206
Shanxi	60	30	42	10	142
Inner Mongolia	35	14	25	5	79
Liaoning	211	67	**167**	**43**	488
Jilin	122	28	71	35	256
Heilongjiang	121	46	88	24	279
Jiangsu	129	70	103	35	337
Zhejiang	213	138	140	42	533
Anhui	59	29	56	11	155
Fujian	95	54	66	16	231
Jiangxi	38	21	33	11	103
Shandong	104	52	126	23	305
Henan	95	60	100	26	281
Hubei	80	48	84	11	223
Hunan	53	41	60	17	171
Guangdong	**218**	**149**	153	41	**561**
Guangxi	79	48	62	24	213

Table 2. (*Continued*)

Hainan	17	6	7	2	32
Sichuan	89	62	97	20	268
Guizhou	18	8	10	1	37
Yunnan	21	15	23	7	66
Xizang	1	0	0	0	1
Shanxi	48	31	48	15	142
Gansu	38	12	27	5	82
Qinghai	7	0	4	0	11
Ningxia	4	0	4	0	8
Xinjiang	0	0	0	0	0

From Table 2, we can do further analysis by manual. Firstly, we analyze top 10 provinces of total comments, which can show the attention degree of different district. The result is shown as Fig.7.

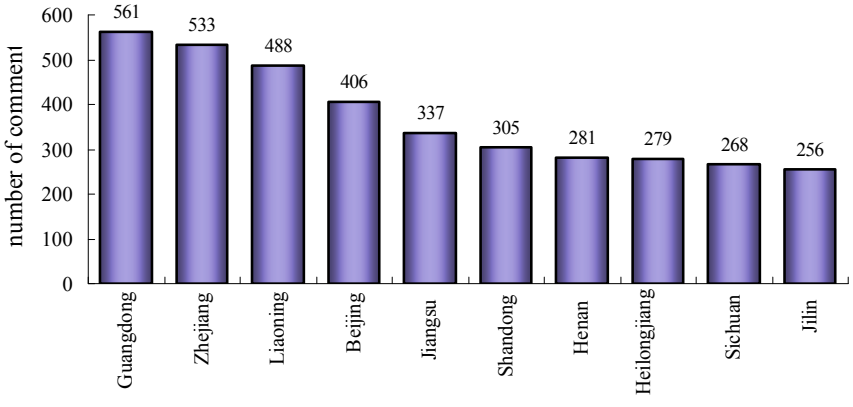


Fig. 7. Top 10 Provinces of Total Comments

According to Fig.7, in these provinces, Guangdong, Zhejiang, and Jiangsu are developed provinces in economy and culture, so their netizens give more attentions to "Spring Festival Evening Show". Because the current programs and shows have more characteristics and style of the north of China, especially the northeast, Liaoning, Heilongjiang, and Jilin (three provinces in northeast) are more interested.

Of four clusters in Table 1, cluster3 is about "northeast, art, couple dance opera", and the sentiment is positive, so Liaoning (one of the three provinces in northeast) has the most number of comments in this cluster. We can also see Liaoning has not the most comments in cluster2, because this cluster is about Zhao Benshan (a comedian from Liaoning), and the sentiment is negative.

7 Conclusions and Future Work

In this paper, we focus on Chinese comments, and propose an approach combining human analysis and machine computing, called semi-automatic public sentiment analysis for opinion and district, which includes automatic data acquiring, comment modeling, opinion clustering, district clustering, and manual threshold setting and result analysis. In comment modeling, we apply LDA model to represent comments and deduce their latent sentiment factors. In opinion clustering, we apply classical *k*-Means algorithm. Experiment shows that our approach is simple and advantageous, which has the following characteristics.

(1) Comparing with full automatic process, it can interact with users or analysts. Comparing with pure manual analysis, it can facilitate opinion analysis by means of LDA representation, sentiment factors deducing, opinion clustering, and district distribution results. So it is feasible and valid.

(2) Comparing with related work, it can obtain not only opinion clusters, but also opinion distribution on different district. Moreover, the analysis can be expanded to other attributes such as status, age, and sex of the netizens.

(3) From experiment results, the label or typical opinion extracted from every cluster is not perfect, which concerns the application of text mining and natural language process techniques.

Moreover, because of inauthentic information on the Web, our analysis results may be inaccurate, but the problem and above disadvantages can be tackled with our future work. So in the future work we should advance the automation and accuracy of opinion analysis by improving our model and process approach.

References

1. Blei, D., Ng, A., Jordan, M.: Latent Dirichlet Allocation. Journal of Machine Learning Research (JMLR) 3, 993–1022 (2003)
2. Dong, Z., Dong, Q.: HowNet, http://www.keenage.com/html/e_index.html
3. Feng, S., Wang, D., Yu, G., Gao, W., Wong, K.: Extracting common emotions from blogs based on fine-grained sentiment clustering. Knowl. Inf. Syst. 27(2), 281–302 (2011)
4. Golaxy: ICTCLAS, http://www.ictclas.org
5. Khan, A., Baharudin, B., Khan, K.: Sentiment Classification from Online Customer Reviews Using Lexical Contextual Sentence Structure. In: Mohamad Zain, J., Wan Mohd, W.M.b., El-Qawasmeh, E. (eds.) ICSECS 2011, Part I. CCIS, vol. 179, pp. 317–331. Springer, Heidelberg (2011)
6. Ku, L., Chen, H.: Mining opinions from the Web: Beyond relevance retrieval. JASIST (JASIS) 58(12), 1838–1850 (2007)
7. Luo, Y., Lin, G., Fu, Y.: Finer Granularity Clustering for Opinion Mining. In: ISCID, pp. 68–71 (2009)
8. MacQueen, J.: Some methods for classification and analysis of multivariate observation. In: Proc. of 5th Berkeley Symp. Math. Statist. and Prob., vol. 1, pp. 281–297 (1967)
9. Melville, P., Gryc, W., Lawrence, R.: Sentiment analysis of blogs by combining lexical knowledge with text classification. In: KDD 2009, pp. 1275–1284 (2009)

10. Phan, X., Nguyen, M., Horiguchi, S.: Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In: WWW 2008, pp. 91–100 (2008)
11. Tan, S., Zhang, J.: An empirical study of sentiment analysis for Chinese documents. Expert Syst. Appl. (ESWA) 34(4), 2622–2629 (2008)
12. Tang, H., Tan, S., Cheng, X.: A survey on sentiment detection of reviews. Expert Syst. Appl. (ESWA) 36(7), 10760–10773 (2009)
13. Titov, I., McDonald, R.: Modeling online reviews with multi-grain topic models. In: WWW 2008, pp. 111–120 (2008)
14. Wang, L., Yn, C., Jia, Y.: Toward Public Opinions Detection: Measuring the Similarity between Instant Messages. In: ODBIS 2008, pp. 21–28 (2008)
15. Wu, Q., Tan, S.: A two-stage framework for cross-domain sentiment classification. Expert Syst. Appl. (ESWA) 38(11), 14269–14275 (2011)
16. Ye, Q., Zhang, Z., Law, R.: Sentiment classification of online reviews to travel destinations by supervised machine learning approaches. Expert Syst. Appl. (ESWA) 36(3), 6527–6535 (2009)

Author Index