Jie Tang
Irwin King
Ling Chen
Jianyong Wang (Eds.)

LNAI 7121

Advanced Data Mining and Applications

7th International Conference, ADMA 2011
Beijing, China, December 2011
Proceedings, Part II

2 Part II

Springer

Lecture Notes in Artificial Intelligence 7121

Subseries of Lecture Notes in Computer Science

Jie Tang Irwin King Ling Chen
Jianyong Wang (Eds.)

Advanced Data Mining and Applications

7th International Conference, ADMA 2011
Beijing, China, December 17-19, 2011
Proceedings, Part II

Springer

Volume Editors

Jie Tang
Jianyong Wang
Tsinghua University
Department of Computer Science and Technology
Beijing, 100084, China
E-mail: {jietang, jianyong}@tsinghua.edu.cn

Irwin King
The Chinese University of Hong Kong
Department of Computer Science and Engineering
Hong Kong, SAR, China
E-mail: king@cse.cuhk.edu.hk

Ling Chen
University of Technology
Faculty of Engineering and Information Technology
Sydney, NSW 2007, Australia
E-mail: ling.chen@uts.edu.au

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-25855-8 e-ISBN 978-3-642-25856-5
DOI 10.1007/978-3-642-25856-5
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: Applied for

CR Subject Classification (1998): I.2, H.3, H.4, H.2.8, J.1, F.1, I.4

LNCS Sublibrary: SL 7 – Artificial Intelligence

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

Printed on acid-free paper

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

Preface

The continuous growth of digital technologies leads to not only the availability of massive amounts of data, but also the emergence of new types of data with novel characteristics. It poses new challenges for the data-mining research community to develop sophisticated data-mining algorithms as well as successful data-mining applications. For the purpose of promoting the original research in advanced data mining and applications, bringing together the experts on data mining throughout the world, and providing a leading international forum to exchange research ideas and results in emergent data-mining problems, the 7th International Conference on Advanced Data Mining and Applications was held in Beijing, China, in 2011.

The conference received 191 paper submissions from 47 countries and areas. All papers were peer reviewed by at least three members of the Program Committee (PC) composed of international experts in data-mining fields, as well as one Vice PC Co-chair. The PC, together with our PC Co-chairs, worked very hard to select papers through a rigorous review process and extensive discussion, and finally composed a diverse and exciting program including 35 full papers and 29 short papers. The ADMA 2011 program was highlighted by three keynote speeches from outstanding researchers in advanced data-mining and application areas: Philip S. Yu (University of Illinois Chicago), Wolfgang Nejdl (L3S Research Center), and Stefan Decker(National University of Ireland).

Without the support of several funding agencies and organizations, the successful organization of the ADMA 2011 would not be possible. These include sponsorships from: IBM Research, China Samsung Telecom R&D Center, and Tsinghua University. We would also like to express our gratitude to the General Co-chairs for all their precious advice and the Organizing Committee for their dedicated organizing efforts. Last but not least, we sincerely thank all the authors, presenters and attendees who jointly contributed to the success of ADMA 2011!

December 2011

Jie Tang
Irwin King
Ling Chen
Jianyong Wang

Organization

ADMA 2011 was organized by Tsinghua University, China, and the School of Information Technology and Electrical Engineering, University of Queensland, Australia.

Organizing Committee

Steering Committee Chair

Xue Li	University of Queensland, Australia

General Co-chairs

Deyi Li	Chinese Academy of Engineering, China
Bing Liu	University of Illinois at Chicago, USA
Charu C. Aggarwal	IBM T.J. Watson Research Center, USA

Program Co-chairs

Jie Tang	Tsinghua University, China
Jianyong Wang	Tsinghua University, China
Irwin King	The Chinese University of Hong Kong, China

Local Co-chair

Jun He	Renmin University of China, China

Regional Organization Co-chairs

Ruoming Jin	Kent State University, USA
Ee-Peng Lim	Singapore Management University, Singapore
Marie-Francine Moens	Katholieke Universiteit Leuven, Belgium
Jimeng Sun	IBM T.J. Watson Research Center, USA
Hwanjo Yu	Pohang University of Science and Technology, Korea
Xingquan Zhu	University of Technology Sydney, Australia

Proceedings Co-chairs

Ling Chen	University of Technology Sydney, Australia
Guoliang Li	Tsinghua University, China

Sponsor Co-chairs

Minlie Huang	Tsinghua University, China
Li Zhang	University of Wisconsin-Madison, USA

Finance chair

Peng Cui	Tsinghua University, China

Publicity Co-chairs

Juanzi Li	Tsinghua University, China
Zhichun Wang	Tsinghua University, China
Jibing Gong	Chinese Academy of Sciences, China

Industrial Track Chair

Keke Cai	IBM Research, China

Registration Chair

Xiaonan Liu	Tsinghua University, China

Special Track Co-chairs

Hongyan Liu	Tsinghua University, China
Zhong Su	IBM Research, China

Web Master

Bo Gao	Tsinghua University, China

Program Committee

Vice PC Chairs

Ling Chen, Australia
Hong Chen, Hong Kong
Xiaoyong Du, China
Zhan-huai Li, China
Marie-Francine Moens, Belgium
Yizhou Sun, USA
Hao Wang, USA
Aoying Zhou, China
Wei Chen, China
Bin Cui, China
Ruoming Jin, USA
Xiaofeng Meng, China
Kyuseok Shim, Korea
Wei Wang, China
Ying Zhao, China

PC Members

Aijun An, Canada
Akihiro Inokuchi, Japan
Ali Daud, China
Andrzej Skowron, Poland
Atsuyoshi Nakamura, Japan
Bruno Cremilleux, France
Cheqing Jin, China
Chiranjib Bhattacharyya, India
Chun-Hung Li, Hong Kong
Cindy Lin, USA
Aixin Sun, Singapore
Alfredo Cuzzocrea, Italy
Amanda Clare, UK
Annalisa Appice, Italy
Brijesh Verma, Australia
Chenhao Tan, USA
Chi Wang, USA
Chotirat Ratanamahatana, Thailand
Chun-Nan Hsu, USA
Daisuke Ikeda, Japan

Daisuke Kawahara, Japan
Daoqiang Zhang, China
David Taniar, Australia
Di Wu, China
Dianhui Wang, Australia
Du Zhang, USA
Faizah Shaari, Malaysia
Fusheng Yu, China
Gang Li, Australia
Guohe Li, China
Guoqiong Liao, China
Hanghang Tong, USA
Harry Zhang, Canada
Hongan Wang, China
Hongzhi Wang, China
Huan Huo, China
Huidong Jin, Australia
James Bailey, Australia
Jan Rauch, Czech Republic
Jiakui Zhao, China
Jianhui Chen, USA
Jibing Gong, China
Jinbao Li, China
Jing Liu, China
Jizhou Luo, China
Keke Cai, China
Kritsada Sriphaew, Japan
Lian Yu, China
Licheng Jiao, China
Lisa Hellerstein, USA
Mao Ye, China
Mario Linares-Vásquez, Colombia
Masashi Sugiyama, Japan
Mengjie Zhang, New Zealand
Michael Madden, Ireland
Michele Berlingerio, Italy
Min Yao, China
Ming Li, China
Ming-Syan Chen, Taiwan
Nicolas Spyratos, France
Odysseas Papapetro, Greece
Panagiotis Karras, Singapore
Philippe Fournier-Viger, Taiwan
Qinbao Song, China
Qingshan Liu, China
Danzhou Liu, USA
Dao-Qing Dai, China
Dexi Liu, China
Diane Cook, USA
Donghui Zhang, USA
Eduardo Hruschka, Brazil
Feiping Nie, USA
Gaël Dias, Portugal
George Karypis, USA
Guojie Song, China
Guoyin Wang, China
Hanzi Wang, China
Hassan Abolhassani, Iran
Hongyan Li, China
Hua Lu, Denmark
Hui Xiong, USA
Hung Son Nguyen, Poland
James Kwok, Hong Kong
Jason Wang, USA
Jian Yin, China
Jian-Min Han, China
Jimmy Huang, Canada
Jing Gao, USA
Jinghai Rao, China
K. Selcuk Candan, USA
Kitsana Waiyamai, Thailand
Li Li, China
Liang Sun, USA
Lidan Shou, China
Manish Gupta, USA
Marco Maggini, Italy
Martine De Cock, Belgium
Masayuki Numao, Japan
Michael Bain, Australia
Michalis Vazirgiannis, Greece
Michele Sebag, France
Ming Ji, China
Mingchun Wang, Taiwan
Nicola Di Mauro, Italy
Ninghui Li, USA
Pablo Castro, Argentina
Patrick Gallinari, France
Qi Wang, China
Qing He, China
Ran Wolff, Israel

Ravi Kumar, USA
Sadok Ben Yahia, Tunisia
Sang-Hyuk Lee, Korea
Sanparith Marukatat, Thailand
Sheng Zhong, USA
Shengtao Sun, China
Shu-Ching Chen, USA
Shuliang Wang, China
Songhua Xu, USA
Stefan Skudlarek, Japan
Sung Ho Ha, Republic of Korea
Takehisa Yairi, Japan
Tao Qin, China,
Thepchai Supnithi, Thailand
Tim Weninger, USA
Tomonari Masada, Japan
Tru Cao, Vietnam
Tu-Anh Nguyen-Hoang, Vietnam
Wai Lam, Hong Kong
Weizhu Chen, China
Wenyang Bai, China
Wlodek Zadrozny, USA
Xiangliang Zhang, Saudi Arabia
Xiaohui Liu, UK
Xin Jin, USA
Xintao Wu, USA
Xue Li, Australia
Yang Xiang, USA
Yasuhiko Morimoto, Japan
Yi Chen, USA
Yihua Wu, USA
Yi-Ping Chen, Australia
Yongli Wang, China
Yubao Liu, China
Yuhua Li, China
Zhihai Wang, China
Zhipeng Xie, China
Zhongzhi Shi, China
Zi Yang, China
Zili Zhang, Australia
Rui Camacho, Portugal
Sai Wu, Singapore
Sanjay Jain, Singapore
Shen Jialie, Singapore
Shengfei Shi, China
Shichao Zhang, China
Shuigeng Zhou, China
Songcan Chen, China
Srikanta Tirthapura, USA
Stefano Ferilli, Italy
Tadashi Nomoto, Japan
Tao Li, USA
Tetsuya Yoshida, Japan
Tieyun Qian, China
Tomoharu Iwata, Japan
Toshiro Minami, Japan
Tsuyoshi Murata, Japan
Wagner Meira Jr., Brazil
Wei Liu, Australia
Wen-Chih Peng, Taiwan
Wilfred Ng, Hong Kong
Wynne Hsu, Singapore
Xiaohua Hu, USA
Xide Lin, USA
Xingquan Zhu, Australia
Xiuli Ma, China
Xuelong Li, China
Yang-Sae Moon, Republic of Korea
Yasuhito Asano, Japan
Yifeng Zeng, Denmark
Ying Zhang, Australia
Yonghong Peng, UK
Yu Jian, China
Yueguo Chen, China
Zhenying He, China
Zhihong Deng, China
Zhongfei Zhang, USA
Zi Huang, Australia
Zijiang Yang, Canada

Table of Contents – Part II

Table of Contents – Part I

Generating Syntactic Tree Templates for Feature-Based Opinion Mining

Liang Wu[1,2], Yuanchun Zhou[1], Fei Tan[1,2], Fenglei Yang[1], and Jianhui Li[1]

[1] Computer Network Information Center, Chinese Academy of Sciences
[2] Graduate University of Chinese Academy of Sciences
100190 Beijing
{wuliang,zyc,tanfei1987,flyang,lijh}@cnic.cn

Abstract. Feature-based sentiment analysis aims to recognize appraisal expressions and identify the targets and the corresponding semantic polarity. State-of-the-art syntactic-based approaches mainly focused on designing effective features for machine learning algorithms and/or predefine some rules to extract opinion words, target words and other opinion-related information. In this paper, we present a novel approach for identifying the relation between target words and opinion words. The proposed algorithm generates tree templates by mining syntactic structures of the annotated corpus. The proposed dependency tree templates cover not only the nodes directly linked with sentiment words and target words, but also subtrees of the nodes on syntactic path, which proved to be effective features for link relation extraction between opinions and targets. Experiment results show that the proposed approach achieves the best performance on the benchmark data set and can work well when syntactic tree templates are applied to different domains.

Keywords: Opinion Mining, Sentiment Analysis, Syntactic Parsing, Relation Extraction.

1 Introduction

In recent years, more and more people contribute information to the Internet. They express their opinions and reviews about certain products, companies and etc. The increasing scale of opinions on the Internet are valuable for manufacturers and commerce companies, from which manufacturers can learn the feedbacks from consumers thus designing new products, and commerce companies may learn the tastes and preference of certain customers thus providing more accurate advertising. However, opinions hidden in the reviews and feedbacks are almost impossible to be manually tagged and summarized. Early works adopted machine learning algorithms and natural language processing techniques to solve this problem.

Previous work in this domain can be classified into three levels [1] according to their granularity, i.e., document level, sentence level and feature level. Document-based opinion mining mainly focuses on classifying a document as

J. Tang et al. (Eds.): ADMA 2011, Part II, LNAI 7121, pp. 1–12, 2011.

positive or negative. As the task is quite similar to text classification, some existing methods are adopted to solve this problem. In [2] Bo Pang et al. experiment several supervised learning methods including Maximum Entropy, naïve Bayesian and support vector machines (SVM) and various feature selection approaches to judge the orientation of documents. The experiment results proved the effectiveness. Subsequent research explored more supervised learning methods [3,4,13,5,6] and unsupervised learning methods [7], different kinds of features including lexical information and bilingual knowledge [8,9,10] are also applied.

Subjectivity classification and semantic polarity estimation are the core tasks of sentence level opinion mining. As both tasks are binary classification, early researchers used many traditional text classification methods such as naïve Bayesian [11,12], support vector machine [14] for detecting subjective expressions. In [12] similar naïve Bayesian based approaches are used to predict the semantic orientation of identified opinionated reviews and feedbacks. Other existing methods like SVM [15] also performs well when applied to detect the opinion polarity. Though both tasks use similar supervised approaches, the differences between them are the features used for learning. Subjectivity detection task focus more on opinion bearing words, thus exploring the subjective information, while sentiment classification task mainly exploits the polarity of the subjective verbs and adjectives and uses negation words like 'not' and 'never' to revise the polarity classification results.

A subjective review may contain multiple opinions, of which each one may consist of different modified targets and semantic polarity. For example in the following sentence: "The sound of this phone is good but the screen is bad", the customer is satisfied with the sound but dislikes the screen of the phone. Reviews like this are of great commercial value. Manufactures can improve their designs based on customers' feedbacks and meet the demands of different people to enlarge their market share. Feature-based sentiment analysis is first introduced by Minqing Hu and Bing Liu in [16] to solve this problem. Opinion mining in this level aims to discover the features appraisal expressions commented on and the corresponding sentiment each feature bears. The input of feature-based opinion mining related works often contains opinion lexicon, target word set and/or tagged corpora. When there are both opinion words and candidate target words in a sentence, system should decide whether the opinion word modifies the target. It can be viewed as a process of link relation extraction between the two elements.

In this paper, we propose a novel approach that exploits syntactic parsing results to extract the link relation. The proposed method first collects syntactic tree structures of appraisal sentences, then generates a collection of syntactic tree templates based on partial tree alignment algorithm [17]. The proposed syntactic tree templates cover not only the syntactic path, which consists of nodes directly between target words and opinion words on syntactic tree, but also the subtrees of the nodes on syntactic path. The subtrees contain a lot of modifiers of the

opinions thus can be useful for appraisal expression recognition. By exploiting the subtrees the precision is improved significantly and reaches as high as 94.7% on a benchmark data set. Some syntactic tree templates with high confidence can also be used for the detection of opinion words and target words, so a new opinion lexicon and target word set generation algorithm is also proposed, which is incorporated with syntactic tree templates and based on double propagation algorithm [18,19]. As the tree templates depict the opinionated expressions from the perspective of syntactic structure instead of words, which prove to be domain specific in many prior work, the tree templates are domain independent.

The remaining of the paper is organized as follows: In section 2, we present some representative work which also focused on extracting the link relation between opinions and targets. Section 3 describes the proposed model based on syntactic tree templates. A new lexicon generation approach is discussed in section 4. In section 5 experiments are given to show the effectiveness of the proposed method and Section 6 concludes the paper.

2 Related Work

Early researchers did some similar work. The distance-based approaches were first introduced in [16], Minqing Hu et al. first identify the candidate target word and then chose the nearest adjective as the corresponding opinion word. The heuristic seems very simple but is quite effective in practise. Subsequent work add more sophisticated rules to increase the precision, in [20], Kim and Hovy proposed different sentiment regions to select the opinion words. But when the appraisal expressions are complicated i.e., the polarity words and its modified targets are not near to each other, this kind of methods may produce errors.

As the opinion words tend to modify certain target words, some researchers try to explore the hidden association between opinion words and target words. In [27], Bloom et al. adopted a dependency parser to build a linkage specification lexicon by hand. Qi Su et al. [23] proposed a mutual reinforcement approach to explore the hidden relatedness by clustering the opinion words and target words, and construct the sentiment association set between the two groups by identifying their sentiment links based on co-occurrence.

Recently, a lot of work uses parsers to explore the syntactic relation between opinion expressions and target words. In supervised learning-based work, syntactic information is applied as classification features. In [14,28], they transform the syntactic trees into tree kernels and build a vector for each sentence. But as text to be parsed in feature level opinion mining is often very short, kernel-based methods may encounter the problem of sparseness and the tree kernels may lose some important information in original syntactic tree. Other researchers proposed template-based algorithms based on dependency trees. In [22], Popescu and Etzioni predefined ten syntactic patterns manually to extract the opinion-target pair. In [26], Yanyan Zhao et al. proposed automatic selected

syntactic path, which is directed syntactic structure linking polarity words and their target, to depict the relation between opinions and targets. Syntactic paths are able to recognize appraisal expressions but are too numerous and specific. Yanyan Zhao et al. introduce some rules to generalize them. During the process of generalization, it also bring in some errors and the precision decreases. The subtrees on syntactic path, which are opinion indicative, are not exploited by syntactic paths.

3 The Approach

3.1 Syntactic Tree Template

Syntactic tree templates are mined from the syntactic parsing trees of opinionated expressions. The template depicts the relation between the opinion word and its target. A template consists of two parts, syntactic path and opinion subtrees. syntactic path has been adopted in many natural language processing tasks like relation extraction and semantic role labeling, and proved to be effective for appraisal expressions recognition in [26]. But the generalization of syntactic path will decrease the precision. In order to recognize the relation more accurately, we propose opinion subtrees based on syntactic path. Before we describe our method, let us make observations about opinionated expressions:

Observation 1(Subtrees of syntactic path are opinion indicative features): On a syntactic dependency tree of opinionated expressions, the subtrees of the syntactic path usually are modifiers of sentiment words and/or modified targets.

Such as the following two sentences:

Sen1: The screen of the MP3 player is not so good.
Sen2: This player seriously is a good one though.

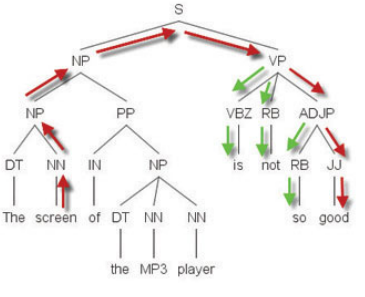


Fig. 1. Syntactic parse tree of Sen1

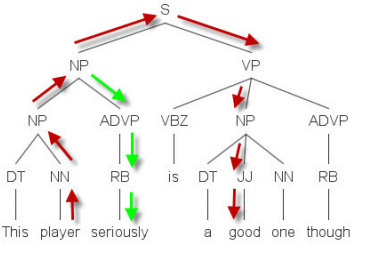


Fig. 2. Syntactic parse tree of Sen2

Figure 1 and Figure 2 are the syntactic parsing results of the two sentences respectively. The red arrow shows the syntactic path of both sentences. The syntactic path of sentence 1 can be represented as **TARGET↑NN↑NP↑NP↑S↑VP↓ADJP↓JJ↓ OPINION↓**, and that of sentence 2 is **TARGET↑NN↑NP↑NP↑**

S↑VP↓NP↓JJ↓ OPINION↓. The green arrows show some useful subtrees of the nodes on syntactic path. In Figure 1, the dark green tagged subtrees modify the sentiment words. Adverbs like "not" and "so" describe the strength of opinions. In Figure 2, the light green subtree which lies in the rising part of the syntactic path represents the attitude of the reviewers, which is also a signal of expressing opinions.

By incorporating the subtrees, the opinion parse tree template can fully exploit the syntactic information thus detecting subjective sentences more effectively. Guang Qiu et al. [18,19] also utilize syntactic parsing results to determine the relationship between opinions and targets. There may be some errors in parsing results, so they only use direct dependencies which is based on the short relations on the parse tree. The approach is feasible for lexicon generation, but may be inadequate for appraisal expression recognition. In [24], Wei Wei et al. proposed a pattern-based approach to extract product features and opinion words, which is compiled all by hand while our proposed syntactic tree templates are produced automatically. Our approach is quite similar to [26]. Since their algorithm may take in errors during the procedure of generalization, subtrees can improve the performance significantly.

3.2 Building Syntactic Tree Template Lexicon

The process of generating syntactic tree templates consists of two parts. First, the syntactic trees which belong to a same template are clustered together. We introduce a simple tree matching algorithm to check whether two trees can be placed into same class. Secondly, a syntactic tree template TE_k is produced for every tree cluster C_k.

Since the syntactic paths are numerous, Yanyan Zhao et al. adopted two generation rules to generalize them. We also use both of the rules, the following Gen 1 and Gen 2 are the two rules proposed in [26]:

Gen 1: If a syntactic path contains sequence of identical constituents, we replace them with one.

Gen 2: If some similar POSs or constituents represents a similar meaning, we generalize them by a normalized one.

Sentiment Tree Matching. We define the sentiment parse tree matching algorithm as follows: For a set of parse trees T = t_1,t_2,...,t_N, which are generated by adopting Stanford Parser [29], a syntactic path p_i is given for each of the tree t_i. By applying the two generalization rules Gen 1 and Gen 2, a generalized path p_i^g is generated for each syntactic path p_i. For two parse trees t_i and t_j, if p_i^g and p_j^g are same, then we define t_i and t_j are matched, i.e., they share an identical template. Based on the generalized syntactic path, we get a generalized parse tree t_i^g for t_i. Noticeably, the subtrees are not generalized.

Partial Tree Alignment. We define the sentiment parse tree alignment algorithm as follows: We randomly select a generalized parse tree from cluster C_k as the seed template TE_k, then merge all remaining generalized parse trees in C_k to TE_k. When merging tree t_{ki} to TE_k, we first align their generalized syntactic path from target word to sentiment word. Then for each node on syntactic path p^g_{ki}, we insert its subtrees to the corresponding node on TE_k. Figure 3 shows the procedure. The insertion times of a subtree are counted as the weight, and we set a threshold to filter those ineffective subtrees.

The syntactic tree template generation procedure is quite similar to simple tree matching algorithm and partial tree alignment algorithm proposed in [17]. Their approach aligns only those data fields in a pair of data records that can be aligned with certainty, and make no commitment on the rest of the data fields, which enables very accurate alignment of multiple data records. As our algorithm only detects the relation between target word and opinion word, we don't exploit the order information of subtrees for simplicity and the alignment algorithm is more straightforward and directed.

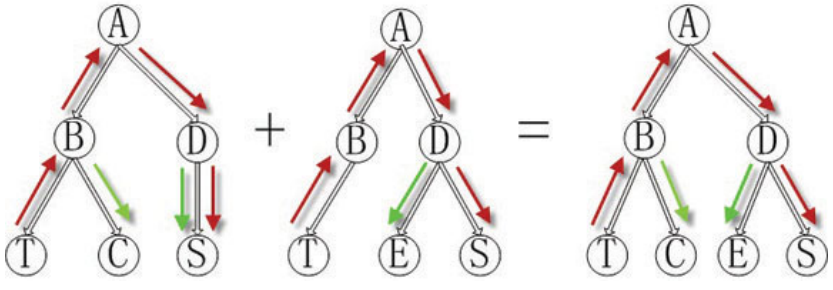


Fig. 3. Partial Tree Alignment

After tree alignment, a syntactic tree template is constructed. It consists of a generalized syntactic path and effective subtrees. We define the number of sentences contained by a template as the weight of it.

3.3 Similarity Computation

For a sentence S, if this sentence contains both candidate target and opinion word, we generate the syntactic parse tree t_s and get the generalized tree t^g_s for it. The path of t_s is p_s, and the path of t^g_s is p^g_s. We try to find the template, which the sentence belongs to by comparing the generalized syntactic path p_s with the template p_k. We assume that the template it belongs to is TE_k. The path of the TE_k, which is the generalized path, is p_{tk}. The TE_k cover all the n parse tree which can be generalized to TE_k , and these trees can be represented

as t_{k1}, t_{k2}, t_{k3}, ..., t_{ki}, ... t_{kn}, and the paths for these trees are: p_{k1}, p_{k2}, p_{k3}, ..., p_{ki}, ... p_{kn}. We define the similarity between S and TE_k as follows:

$$SIM(S, TE_k) = \alpha SIM_p(S, TE_k) + (1 - \alpha) SIM_t(S, TE_k)$$

The total similarity contains two parts: one is the path similarity, which calculate the similarity between the p_s, and p_{k1}, p_{k2}, p_{k3}, ... p_{kn}; and the other is subtree similarity, which calculate the similarity between the subtree of the nodes in p_s^g and subtree of nodes in p_{tk}. And the α is the so-called weighted factor ($0 \leq \alpha \leq 1$) that control the contribution of the two parts.

The path similarity between the sentence S and the template TE_k computation is as follows:

$$SIM_p(S, TE_k) = \frac{LEN(p_s) - MIN_{1 \leq i \leq n}\{edit\_dist(p_s, p_{ki})\}}{LEN(p_s)}$$

The $LEN(p_s)$ is the number of nodes between the opinion word and the target word. And the $edit\_dist(p_s, p_{ki})$ is the edit distance between the t_s and t_{ki}. We try to find the minimum edit distance value between the t_s and the parse trees which belong to the TE_k, just as $MIN_{1 \leq i \leq n}\{edit\_dist(p_s, p_{ki})\}$.

If the nodes in the p_s^g is $node_{s1}^g$, $node_{s2}^g$, ..., $node_{si}^g$, ..., $node_{sm}^g$. And the corresponding nodes in the p_{tk} is $node_{tk1}$, $node_{tk2}$, ..., $node_{tki}$, ..., $node_{tkm}$. Then We define the subtree similarity as:

$$SIM_t(S, TE_k) = \frac{\sum_{i=1}^{m} C(node_{si}^g, node_{tkm})}{\sum_{i=1}^{m} |node_{si}^g|}$$

where

$$C(node_{si}^g, node_{tki}) = \begin{cases} |node_{si}^g|, & \text{if subtree of } node_{tki} \text{ contains } node_{si}^g{}'s subtree, \\ 0, & \text{otherwise.} \end{cases}$$

The $|node_{si}^g|$ is the number of nodes in the subtree of node $node_{si}^g$.

In order to extract those appraisal expressions, a threshold $\hbar$, is set to find those sentences which contains a high similarity with certain template. If $SIM(S, TE_k) \geq \hbar$, the candidate target and opinion word in S has the same relation with the template TE_k. The optimal value of $\hbar$ can be trained from the train data.

4 Lexicon Generation

We use the algorithm in [18,19] to find candidate target words and opinion words. They manually compile some extracting rules based on dependencies. If we regard the corpora as a graph, where the sentiment words and the target words are verticals and the propagation rules as edges. The algorithm is to expand the lexicon by traversing the graph from the seed verticals.

But if the graph contains too many isolated subgraphs, and/or seed verticals cover only a small part of the subparts, the recall will be small. In order to increase the connectivity of corpus, we introduce some top weighted tree templates as extraction rules.

5 Experiments and Results

In this section, we discuss the annotated corpus and experiment settings of baseline methods and our proposed algorithm. The experiment results of lexicon generation and appraisal expressions recognition on both in-domain and cross-domain are also analyzed and discussed.

5.1 Corpus

We conducted experiments with labeled corpus in [14]. The corpus is selected from [16]. Their data is collected from CNet.com and Amazon.com. Minqing Hu et al. labeled the product features and sematic polarity. Yuanbin Wu et al. asked two annotators to label the opinion expressions of the corpus.

5.2 Preprocessing Results

Results of lexicon generation are shown in Table 1 and Table 2. We experiment three methods and use precision, recall and F-measure to evaluate the performances. For both experiments, we randomly select 10% of the opinion words and experiment 10 times, the shown results are the average.

Table 1. Features extract evaluation

Methods	P	R	F
PDP	42.8%	85.5%	57.0%
DP	82.3%	67.7%	74.3%
DP_Tree	74.8%	98.3%	85.5%

Table 2. Opinion extract evaluation

Methods	P	R	F
PDP	62.5%	75.2%	57.0%
DP	41.5%	39.6%	40.5%
DP_Tree	43.7%	79.2%	56.3%

Table 1 illustrates the results of product feature extraction. PDP is the method adopted in [14], DP is the double propagation method proposed in [19,18] and DP_{tree} is the modified version of DP, which incorporates some top weighted tree templates. We see that DP algorithm outperforms method used in [14] in precision and F-measure. But the recall of DP is low. After applying template syntactic trees, the DP method outperforms the other two baselines in precision and recall. This improvement of performance proves the validity of tree templates, which improves the performance by strengthening connectivity.

Table 2 shows the experiment results of opinion lexicon generation. As the structure of the corpus is not suitable for double propagation algorithm, which consists of multiple isolated subgraphs. PDP gets the best F-measure, which adopts the distance-based algorithm in [16] and a opinion dictionary. But as the acquisition of both target words and sentiment words are preprocessing steps, the recall is more important. The improved version of double propagation achieves the best recall.

Table 3. Results of different methods

	Cell Phone			MP3 Player			Digital Camera			DVD Player		
Methods	P	R	F	P	R	F	P	R	F	P	R	F
Adjacent	40.3%	60.5%	48.4%	26.5%	59.3%	36.7%	32.7%	59.1%	42.1%	31.8%	68.4%	43.4%
SVM-1	69.5%	42.3%	52.6%	60.7%	30.6%	40.7%	61.4%	32.4%	42.4%	56.0%	27.6%	37.0%
SVM-2	60.7%	19.7%	29.7%	63.6%	23.8%	34.6%	66.9%	23.3%	34.6%	66.7%	13.2%	22.0%
SVM-WTree	52.6%	52.7%	52.6%	46.4%	43.8%	45.1%	49.1%	46.0%	47.5%	35.9%	32.0%	33.8%
SVM-PTree	55.6%	57.2%	56.4%	51.7%	50.7%	51.2%	54.0%	49.9%	51.9%	37.1%	35.4%	36.2%
Template	92.8%	48.2%	63.4%	91.1%	58.7%	71.4%	91.6%	50.0%	64.7%	94.7%	38.0%	54.3%
S-Path	81.2%	49.7%	61.7%	84.7%	59.0%	69.6%	79.4%	56.3%	65.9%	81.7%	43.3%	56.6%

Table 4. Results of total performance with cross domain training data

	Diaper			DVD Player			MP3 Player		
Methods	P	R	F	P	R	F	P	R	F
Adjacent	23.4%	78.8%	36.1%	31.8%	68.4%	43.4%	26.5%	59.3%	36.7%
SVM-1	22.4%	30.6%	25.9%	52.8%	30.9%	39.0%	55.9%	36.8%	44.4%
SVM-2	71.9%	15.1%	25.0%	51.2%	13.2%	21.0%	63.1%	22.0%	32.6%
SVM-WTree	38.7%	52.4%	44.5%	30.7%	59.2%	40.4%	38.1%	47.2%	42.2%
SVM-PTree	37.3%	53.7%	44.0%	59.2%	48.3%	46.3%	43.0%	48.9%	45.8%
Template	88.3%	98%	92.9%	89.0%	53.0%	66.5%	89.1%	59.1%	71.0%

5.3 Appraisal Expressions Extraction Experiments

Experiments Settings. In order to compare with the state-of-the-art results, we compare our method with the approaches in [14]. Adjacent, SVM-1, SVM-2, SVM-WTree and SVM-PTree are designed by Yuanbin Wu et al.. S-path is the result of syntactic path algorithm, which is also used in [26]. Template denotes the results of our syntactic tree template based approach.

Table 3 shows the results of in-domain experiment. Five-fold cross validation is conducted for each domain, which is same as the settings of [14]. So we directly use their experiment results. Table 4 shows the performances of cross-domain experiments. We also adopt the settings in [14], which takes digital camera and cell phone domain as training set and takes the other domains as testing set.

Results Discussion. Table 3 presents different approaches' results in four domains. We observe that Template and S-path outperform the other methods. Template achieves the best precision in all domains, but S-path baseline outperforms Template in Digital Camera and DVD Player domains. The results prove the capability of tree templates for detecting subjective expressions. When applied to practical tasks, the performance will improve significantly if it can be

trained with a larger data set. We don't present the experiment results of Diaper domain, as the performance changes greatly when started with different seeds. The F-measure even reaches 0 sometimes. It may be caused by Diaper's data size, which is too small to generate enough templates.

For evaluations on cross domain, our approach outperforms all the baselines. Since the training set is larger and the model is sufficiently trained, the performance in Diaper domain is improved obviously. The results verified our intuition: Though people tend to use different words to express their opinions in different domains, the syntactic structure of their expressions converges. Noticeably, the recall of Diaper reaches as high as 98.0%, which shows that when training set is sufficient the performance will improve.

6 Conclusions and Future Work

In this paper, we proposed an appraisal expression recognition approach based on tree templates and a revised version of double propagation, which exploits top weighted tree templates as extracting rules. The contributions of our work included: 1) Syntactic tree templates can effectively extract the relationship between opinion words and target words. 2) Edit distance between original syntactic paths can better avoid the noise brought by generalization. 3) The proposed method is valid for cross-domain data.

In future, we will explore to exploit the tree templates for sentence sentiment classification, as the subtrees of syntactic path can affect the polarity of the whole sentence. In addition, as traditional parsers are not efficient enough to be used in industry, tree templates can be used to implement a lighter approximation of parsers for opinion mining.

References

1. Liu, B.: Sentiment analysis and subjectivity Handbook of Natural Language Processing, 2nd edn., pp. 1–38 (2010)
2. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up? Sentiment classification using machine learning techniques. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 79–86 (2002)
3. Pang, B., Lee, L.: A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. In: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics, pp. 271–278 (2004)
4. Boiy, E., Moens, M.-F.: A machine learning approach to sentiment analysis in multilingual Web texts. Information Retrieval 12, 526–558 (2008)
5. Jin, W., Ho, H.: A novel lexicalized HMM-based learning framework for web opinion mining. In: Proceedings of the 26th Annual International Conference on Maching Learning, pp. 465–472 (2009)
6. Zhao, J., Liu, K., Wang, G.: Adding redundant features for CRFs-based sentence sentiment classification. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 117–126 (2008)

7. Turney, P.: Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In: Proceedings of the Association for Computational Linguistics, pp. 417–424 (2002)
8. Wan, X.: Co-training for cross-lingual sentiment classification. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, pp. 235–243 (2009)
9. Wan, X.: Using bilingual knowledge and ensemble techniques for unsupervised Chinese sentiment analysis. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 553–561 (2008)
10. Lu, B., Tan, C., Cardie, C., Tsou, B.K.: Joint bilingual sentiment classification with unlabeled parallel corpora. In: Proceedings of Joint Conference of the 49th Annual Meeting of the Association for Computational Linguistics and the Human Language Technologies Conference (2011)
11. Wiebe, J., Bruce, R., O'Hara, T.: Development and use of a gold standard data set for subjectivity classifications. In: Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics, pp. 246–253 (2008)
12. Yu, H., Hatzivassiloglou, V.: Towards Answering Opinion Questions: Separating Facts from Opinions and Identifying the Polarity of Opinion Sentences. In: Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing, pp. 129–136 (2003)
13. Boiy, E., Hens, P., Deschacht, K., Moens, M.-F.: Automatic sentiment analysis of on-line text. In: Proceedings of the 11th International Conference on Electronic Publishing, pp. 349–360 (2007)
14. Wu, Y., et al.: Phrase dependency parsing for opinion mining. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, vol. 3, pp. 1533–1541 (2009)
15. Gamon, M.: Sentiment classification on customer feedback data: noisy data, large feature vectors, and the role of linguistic analysis. In: Proceedings of the 20th International Conference on Computational Linguistics, pp. 841–847 (2004)
16. Hu, M., Liu, B.: Mining opinion features in customer reviews. In: Proceedings of the 19th National Conference on Artifical Intelligence, pp. 755–760 (2004)
17. Zhai, Y., Liu, B.: Web data extraction based on partial tree alignment. In: Proceedings of the 14th International Conference on World Wide Web, pp. 76–85 (2005)
18. Qiu, G., Liu, B., Bu, J., Chen, C.: Expanding domain sentiment lexicon through double propagation. In: Proceedings of the 21st International Joint Conference on Artifical Intelligence, pp. 1199–1204 (2009)
19. Qiu, G., Liu, B., Bu, J., Chen, C.: Opinion word expansion and target extraction through double propagation. Computational Linguistics 37, 9–27 (2011)
20. Kim, S.M., Hovy, E.: Determining the sentiment of opinions. In: Proceedings of the 20th International Conference on Computational Linguistics, pp. 1367–1373 (2004)
21. Kobayashi, N., Inui, K., Matsumoto, Y., Tateishi, K., Fukushima, T.: Collecting evaluative expressions for opinion extraction. In: Proceedings of the International Joint Conference on Natural Language Processing, pp. 596–605 (2004)
22. Popescu, A.M., Etzioni, O.: Extracting product features and opinions from reviews. In: Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing, pp. 339–346 (2005)

23. Su, Q., et al.: Hidden sentiment association in chinese web opinion mining. In: Proceeding of the 17th International Conference on World Wide Web, pp. 959–968 (2008)
24. Wei, W., Liu, H., He, J., Yang, H., Du, X.: Extracting Feature and Opinion Words Effectively from Chinese Product Reviews. In: Proceedings of the 2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery, vol. 4, pp. 170–174 (2008)
25. Whitelaw, C., Garg, N., Argamon, S.: Using appraisal taxonomies for sentiment analysis In:Proceeding of CIKM 2005 (2005)
26. Zhao, Y., Qin, B., Che, W., Liu, T.: Appraisal expressions recognition with syntactic path for sentence sentiment classification. International Journal of Computer Processing Of Languages 23, 21–37 (2011)
27. Bloom, K., Garg, N., Argamon, S.: Extraction appraisal expressions. In: Proceeding of HLT-NAACL 2007, pp. 308–315 (2007)
28. Jiang, P., Zhang, C., Fu, H., Niu, Z., Yang, Q.: An Approach Based on Tree Kernels for Opinion Mining of Online Product Reviews. In: Proceedings of the 2010 IEEE International Conference on Data Mining, pp. 256–265 (2010)
29. Klein, D., Manning, C.: Accurate unlexicalized parsing. In: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics, vol. 1, pp. 423–430 (2003)

Handling Concept Drift via Ensemble and Class Distribution Estimation Technique

Nachai Limsetto and Kitsana Waiyamai

Data Analysis and Knowledge Discovery Lab (DAKDL)
Department of computer engineering Faculty of Engineering Kasetsart University BangKhen campus Bangkok, 10900, Thailand
khim1204@gmail.com, fengknw@ku.ac.th

Abstract. In real world settings there is situation where class distribution of data may change after classifier is built resulting in performance degradation of classifier. Attempts to solve this problem from previous Class Distribution Estimation method (CDE method) yield quite interesting performance however we notice there is some flaw since CDE method still have some bias toward train data thus we decide to improve them with ensemble method. Our Class Distribution Estimation-Ensemble (CDE-EM) methods estimate class distribution from many models instead of one resulting in less bias than previous method. All methods are evaluated using accuracy on set of benchmark UCI data sets. Experimental results demonstrate that our methods yield better performance if class distribution of test data is different from train data.

Keywords: Data mining, Classification, Cost sensitive learning, Quantification, Concept drift, Class distribution Ensemble method.

1 Introduction

Concept drift [15] is a problem commonly found in classification task of data mining. It's a situation where class distribution changes as a time passes, thus result in distribution mismatch problem [7]. Example for concept drift is predicting weekly merchandise sales, given that you already built model for it. You are likely to find that accuracy of your model will yield lower accuracy as the time passes since shopping behavior is usually none-static, due to various reasons such as season etc. This problem can be solved in various ways such as robust classifier [2, 10], data sampling [5, 15], semi-supervised learning [4] and cost-based learning [1, 9 ,14, 19].

To maximize classifier performance on future data while having no information about future changes in class distribution [20], in this situation adapting classifier to changing conditions is inappropriate. But rather we should build a robust classifier that performs well under a wide variety of situations [2, 10]**.** While this approach of generating robust classifiers has its advantages and is applicable to our problem, it clearly is not the best strategy to use it alone when the class distribution of the new data can be reliably estimated via the class distribution from unlabeled data.

Sampling from the original distribution is another way of compensating for differing class distributions so that its class distribution matches the estimated class

J. Tang et al. (Eds.): ADMA 2011, Part II, LNAI 7121, pp. 13–26, 2011.

distribution of the new data. There is many sampling methods [15], such as oversampling and under-sampling, as well as more advanced methods which make better use of the data [5].

Semi-supervised learning methods [4] are relevant since our learning task involves labeled and unlabeled data however the results thus far for the SSL-based methods have not been very promising, since it is easier to accurately estimate the class distribution of the unlabeled examples than to classify them.

While above methods focused on improving classifier performance in response to class distribution's change of the data, a related, but different, task is the quantification task. Quantification involves estimating the prevalence of the classes over time [7]. Quantification is a simpler task than classification because good estimation of a class distribution can be obtained without producing accurate predictions for individual examples. As we have seen in previous work [19], quantification methods can also help solve our more complex task. Quantification techniques are discussed in detail by Forman [7, 8].

So far, cost-based learning methods have shown the best result. Some of them have involved expectation maximization (EM) [9, 14]. EM method uses the probability estimates to estimate change in class distribution and adjusts the original model's posterior probability outputs. Other works focus on minimizing the maximum deviation (regret) from the optimal Bayes classifier [1] to achieve better performance. While those methods perform quite well and have some advantage in some situation there is Class Distribution Estimation method (CDE method) [19] that yields better result in this circumstance. CDE method exploits the fact that if we can estimate the class distribution from which future examples will be drawn, then we can adjust the original classifier to account for the differences in class distribution.

In this research work, our idea is to develop new methods based on CDE and ensemble technique [13]. Our objective is to overcome previous weakness of CDE that tense to underestimate the change in class distribution thus lead to performance degradation if there is difference in class distribution between train and test data. We develop four methods using ensemble technique. First is Class Distribution Estimation Ensemble-AVG (CDE-EM- AVG), which uses average value from each model to estimate the change in class distribution. Second is Class Distribution Estimation with Ensemble-BM (CDE-EM- BM) that selects the best model among the constructed models. Third is Class Distribution Estimation with Ensemble-AVGBM (CDE-EM- AVGBM) that combines the two previous methods together. Fourth is Class Distribution Estimation with Ensemble-exclude (CDE-EM- EX) that uses the middle model to estimate class distribution change and discards inappropriate models. We evaluate all methods in terms of accuracy and robustness. Results from our experiments demonstrate that our methods yield better performance if class distribution of train data is different from test data.

The remainder of this paper is structured as follows. Section 2 describes important basic knowledge and previous works. Section 3 presents full detail of our methods. Experiment methodology is presented in Section 4 follow by experimented results in section 5. Section 6 summarizes our conclusions then referents are present in section 7.

2 Class Distributions Estimation-Based Methods (CDE-Based Methods)

In this section we describe CDE-based method which our method intend to improve and basic knowledge related to CDE-based method.

2.1 Cost Sensitive Classifier (CSC)

CSC [6] is a type of classifier which works with another classifier. Normally, a classifier model is built with cost of correctly classify and incorrectly classify equally for each class. However, in CSC, these costs will be adjusted to be different. With proper cost adjustment, CSC yields better performance in terms of accuracy and f-measure.

2.2 Class Distribution

Class distribution is the percentage of examples in a class varying from 0% to 100%. Class distribution will be different depending on its number of examples. In case of binary classification, class distribution can be represented with positive rate (pr) and negative rate (nr). Positive rate is a percentage of positive examples in the dataset and negative rate is a percentage of negative examples in the dataset. Notice that the difference in class distribution between training and testing sets will be resulted in performance degradation of a classifier.

2.3 Positive-to-Negative Ratio (pr/nr)

pr/nr is ratio of positive examples to negative examples in a dataset. In case of binary classification, (pr/nr) can be defined using equation 1.

$$(\mathrm{pr}/\mathrm{nr}) = \frac{\text{Number of positive class examples}}{\text{Number of negative class examples}} \tag{1}$$

2.4 Distribution Mismatch Ratio (DMR)

DMR is ratio of distribution mismatch between train data and test data. (pr/nr) is obtained from train data while (pr/nr)' obtain from test data. We can calculate DMR using equation 2.

$$\mathrm{DMR} = \frac{(\mathrm{pr}/\mathrm{nr})'}{(\mathrm{pr}/\mathrm{nr})} \tag{2}$$

Example: Given a dataset drawn from the original distribution (training data). It has 900 positive examples and 100 negative examples (pr = 9/10, nr=1/10) and the data drawn from the new distribution (testing data) has 200 positive and 800 negative examples (pr'=2/10, nr'=8/10). DMR indicates the factor by which the ratio of positive to negative examples changes between the original and new distribution. For this example, DMR = 9**:**¼ or, equivalently, 36:1. Thus, based on the ratio of the

positive rate to negative rate, the positives are 36 times more prevalent in the original distribution than in the new distribution. This means if we train model with original distribution (train data) and expect this model to work well in new distribution (test data) then the cost of negative example miss-classification should be 36 times more than the cost of positive example miss-classification to compensate for lack of negative example information.

2.5 Class Distribution Estimation Based Method (CDE)

CDE exploits the fact that if class distribution of unseen data can be estimated, then the original classifier can be adjusted to account for the differences in class distribution of the two datasets.

There are two CDE methods: CDE Iterate-n and CDE Adjusted Count (CDE-AC). CDE Iterate-n is shown in Figure 1.

CDE Iterate-n builds classification model from train data, applies it on test data then uses the result of classification to calculate DMR. Then, it adjusts the cost value of CSC classifier according to the cost value. This process continues n times. However, it is already shown in [19] that the more this process goes on the less accuracy we gain from it. We found that in most cases two iterations is already enough since the accuracy gain is usually less than 0.05% if we go for more than two iterations.

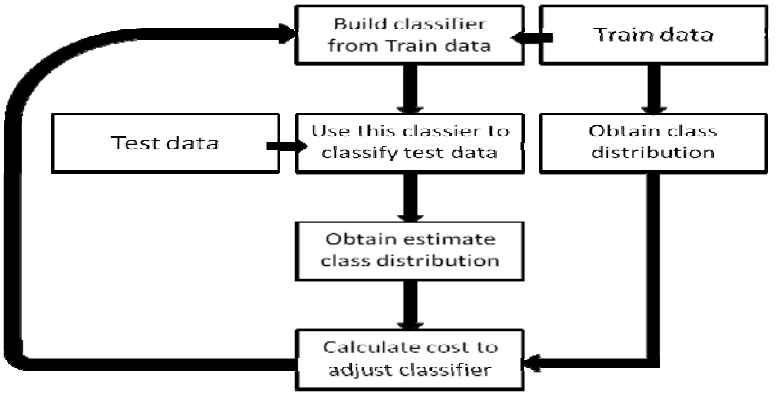


Fig. 1. CDE-Iterate-n

Second method is CDE Adjusted Count (CDE-AC) method which uses 10-fold cross validation on train data. In [18], the authors demonstrates good performance of CDE-AC, however we found that this method is not suitable if class distribution of train data and test data is very different. Beside the fact that cross-validation uses random method to generate many groups of data make the classifier even less reliable than desired.

While the goal of CDE methods is to make classifier less bias to class distribution of the train data, some bias still remain. Since class distribution estimation is obtained from only one model built from class distribution of train data. Thus, there is room to improve them via methods that implement more than one model such as ensemble method.

2.6 CDE Oracle Method

Oracle method [19] provides a potential upper bound for achievable performance by building a cost sensitive classifier from the original data (train data) using cost value obtained from correct class distribution of new data (test data).

3 Our Methods

In this section, we describe the proposed methods and their related knowledge. Our objective is to reduce class distribution biasing toward train data by combining CDE with ensemble method. The idea is to use more than one model to overcome the problem of biasing toward train data. We propose four methods which are CDE-EM-AVG-N, CDE-EM-BM-N, CDE-EM-AVGBM-M-N and CDE-EM-EX-N.

CDE-EM-AVG-N builds multiple models with different class distribution then uses average value obtained from these models to estimate class distribution of test data.

Instead of using average value from many models, CDE-EM-BM-N selects the best model which has been built from the most similar class distribution to test data. While this method may sound promising, it has problem about the correct model selection having class distribution similar to test data, thus leads to lower accuracy compared to other methods. This problem will be solved in the third method which is EM-AVGBM-M-N.

While CDE-EM-AVGBM-M-N selects the best model like CDE-EM-BM-N, it does not use that model alone. Instead, it uses neighbor models built from similar class distribution of the selected model to estimate class distribution of test data.

Finally, CDE-EM-EX-N uses the middle model (model built with cost of class distribution (pr/nr) equal to 1) as model selector to exclude inappropriate models building from class distribution that very different from the test data.

3.1 CDE-EM-AVG-N

Our CDE-EM-AVG-N handles change in class distribution and reduces bias toward training data via ensemble method. The objective is to use many models that have been built with different costs obtained from many class distributions to reduce biasing over original data thus yield better performance.

This method starts by take value N which is a number of classifier models going to build. Then, we calculate the model's interval by equation 3.

$$\text{interval} = \frac{100}{\text{N}} \tag{3}$$

After that we calculate $(pr/nr)_i$ to build classifier model i by equation 4.

$$(\text{pr}/\text{nr})_{\text{i}} = \frac{\text{i} \times \text{interval}}{100 - (\text{i} \times \text{interval})} \quad \text{i varying from 1 to N} - 1 \tag{4}$$

Cost calculation for cost sensitive classifier is performed by using $(pr/nr)_i$ instead of $(pr/nr)'$ (in equation 2). Then, CDE-EM-AVG-N builds N cost sensitive classifiers

from the original data (train data) using the obtained cost. Each $classifier_i$ will have different cost based on DMR_i. Then, it uses these cost sensitive classifiers on new data (test data) to obtain estimate class distributions and (pr/nr) of test data from each model. $\mathrm{AVG}(pr/nr)$ is calculated from equation 5.

$$\mathrm{AVG(pr/nr)} = \frac{\sum_{i=1}^{N}(\mathrm{pr/nr})_i}{N} \tag{5}$$

Now, CDE-EM-AVG-N calculates DMR again from all the models that have been built. It substitutes $(pr/nr)'$ by $\mathrm{AVG}(pr/nr)$ then it builds new cost sensitive classifier from the original data (train data) using this cost.

Flow chart of CDE-EM-AVG-N is shown in Figure 2.

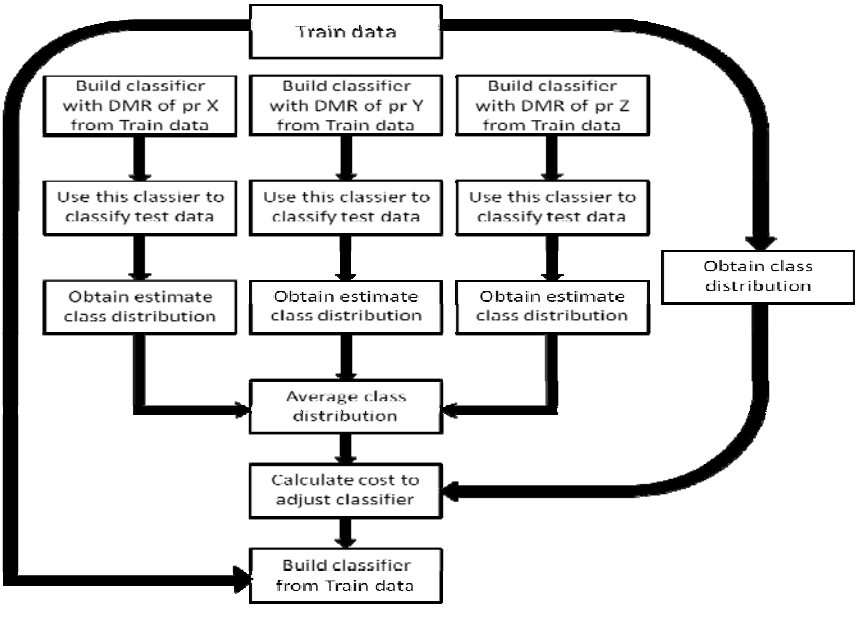


Fig. 2. Flow chart of CDE-EM-AVG-N

3.2 CDE-EM-BM-N

Instead of using average value from many models, CDE-EM-BM-N selects the best model having DMR nearest to 1.00 as possible. The selected model should have been built from the most similar class distribution to test data. We expect CDE-EM-BM-N to improve performance of classifier however the experiment results show otherwise.

3.3 CDE-EM-AVGBM-M-N

The CDE-EM-AVGBM-M-N is a combination of CDE-EM-AVG-N and CDE-EM-BM-N. It selects the best model like CDE-EM-BM-N but does not use that model

alone. Instead, it uses set of models that built from cost similar to class distribution of the selected model to calculate the average (pr/nr). The M value indicates how many models we use to find average value. This method intends to improve the previous method since we found that it is hard to select the correct model without error.

3.4 CDE-EM-EX-N

CDE-EM-EX-N is similar to the previous methods, however we use model that built with cost of class distribution (pr/nr) equal to 1, which mean two classes having equal number of examples, to estimate the value of (pr/nr) of new data (test data) and select the set of models that built from cost of similar class distribution to calculate the average (pr/nr). Though, this method is very similar to CDE-EM-AVGBM-M-N, experiment results show CDE-EM-EX-N has significant improvement in terms of accuracy and robustness.

4 Experiment Methodology

In order to evaluate the effectiveness of our proposed methods, we compare them with Oracle and CDE-Iterate-n method. The evaluation is performed in terms of accuracy and robustness. This provides information for how good the proposed methods are.

We describe our experimental setup here which consists of three UCI data sets. Description of the three datasets is given in Table 1.

Table 1. Description of 3 UCI Data Sets

Dataset	Size	% of Positive class (positive rate)	Partition size
Adult	32,561	24.08%	8000
Magic Gamma	19,020	64.84%	7000
Spambase	4,601	60.6%	1900

All the datasets we use contain only two classes. The original dataset size, the percentage of the examples that belong to the positive class and the number of examples in each partition are given. As described earlier, the problem we investigate requires two datasets: the original data (train data) and new data (test data).

In experiment 1 and 2 we want to test how these methods work in general situation so we do not adjust class distribution of train data, instead we train model with it real class distribution. In these two experiments, the original data (train data) is the entire dataset without any partitioning.

The goal of experiment 3 and 4 is to measure performance of these methods when class distribution of train data is very different from test data, thus the original data (train data) is partitioned from original dataset by 5% and 95% of positive class, respectively.

New data (test data) is generated in the same way. The new datasets have varying positive rate between 5% and 95% with 1% incremental rate. In order to generate class distributions without duplication, size of each partition must be limited. We use the maximum possible partition size for each dataset. The partition sizes used in the experiments are specified in the last column of Table1.

All the experiments in this paper utilize the J48 [17] implementation of C4.5 [12] from WEKA released 3.6.3. Changes in class distribution are compensated using WEKA's cost-sensitive learning capabilities with example reweighting.

All the proposed methods could easily be applied to other supervised learning methods since they are implemented as wrapper-based learners. Their classification performances are evaluated in terms of accuracy and robustness.

The robustness previously mentioned measures the average mismatch obtained by comparing one method with Oracle. Difference of Oracle and the method is summed only if its accuracy is more than the method. Oracle is considered as the upper bound method. If the method's accuracy is above Oracle we consider it is due to noise. Summed accuracy is divided with the number of time Oracle has more accuracy than the method. This shows how robustness is the method.

5 Experiment Results

In this section we present and analyze our experimental results. The detailed accuracy results for the Adult data set are shown in Figure 3 and Table 2.

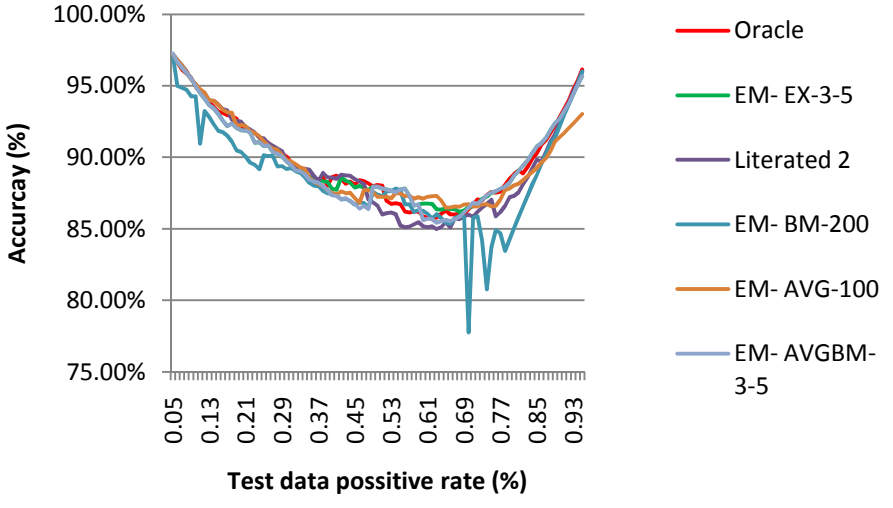


Fig. 3. Experiment 1: Accuracy Performance on Adult Dataset

Table 2. Experiment 1: Accuracy Performance on Adult Dataset

		CDE-based methods		Our methods			
Pr (%)	Oracle	Iterated1	Iterated2	EM- BM-200	EM-AVG-100	AVGBM-3-5	EM-EX-3-5
0.05	97.09%	95.10%	97.20%	97.09%	97.20%	97.25%	97.10%
0.1	94.95%	93.81%	95.11%	94.25%	95.13%	94.98%	94.98%
0.2	92.21%	91.06%	92.49%	90.36%	92.26%	91.89%	91.89%
0.3	90.05%	88.83%	89.83%	89.18%	89.80%	89.71%	89.71%
0.4	88.60%	85.54%	88.58%	87.44%	87.53%	87.34%	87.79%
0.5	88.06%	84.39%	86.64%	87.39%	87.23%	87.94%	87.94%
0.6	85.86%	84.44%	85.18%	86.25%	87.10%	85.60%	86.76%
0.7	86.35%	85.99%	86.00%	77.74%	86.71%	86.48%	86.51%
0.8	88.81%	87.89%	87.30%	85.05%	88.04%	88.70%	88.70%
0.9	92.81%	92.74%	92.36%	92.16%	91.39%	92.69%	92.69%
0.95	96.15%	96.20%	96.00%	95.98%	93.03%	95.66%	95.66%
AVG	89.80%	88.49%	89.44%	88.58%	89.57%	89.62%	89.83%

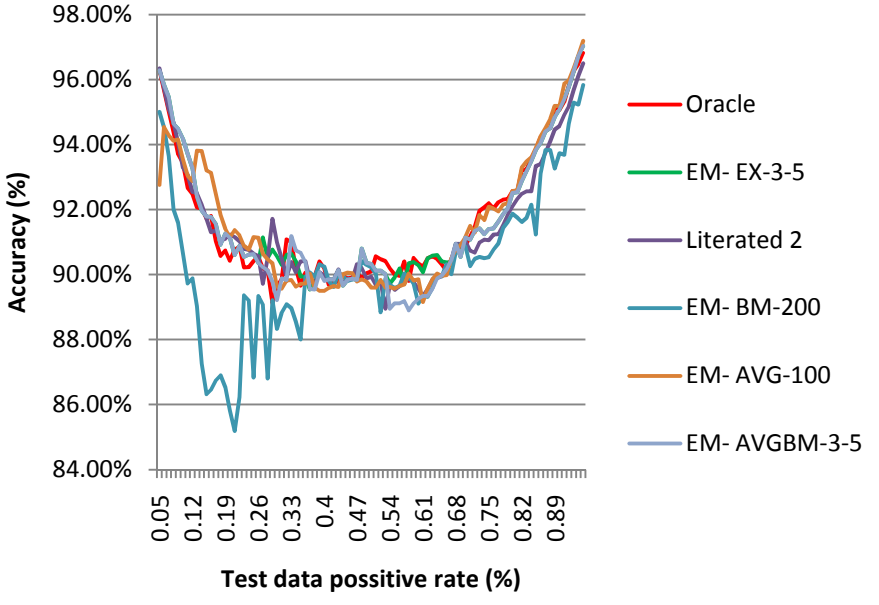


Fig. 4. Experiment 2: Accuracy Performance on Magic Gamma Dataset

Table 3. Experiment 2: Accuracy Performance on Magic Gamma Dataset

		CDE-based methods		Our methods			
Pr (%)	Oracle	Iterate1	Iterate2	EM- BM-200	EM- AVG-100	AVGBM- 3-5	EM-EX -3-5
0.05	96.29%	95.14%	96.34%	95.00%	92.76%	96.29%	96.29%
0.1	93.43%	93.91%	93.29%	90.67%	93.56%	94.19%	94.19%
0.2	90.43%	91.71%	91.23%	85.81%	91.14%	91.13%	91.13%
0.3	89.63%	89.63%	91.00%	88.33%	89.63%	89.21%	90.56%
0.4	90.19%	89.56%	89.96%	90.24%	89.50%	89.80%	89.89%
0.5	90.10%	89.53%	89.93%	90.21%	89.60%	90.33%	90.33%
0.6	90.34%	89.59%	89.56%	89.10%	89.86%	89.26%	90.29%
0.7	90.87%	91.16%	91.03%	90.87%	91.16%	91.14%	91.14%
0.8	92.57%	92.51%	92.10%	91.87%	92.57%	92.51%	92.51%
0.9	95.06%	95.19%	94.57%	93.73%	95.19%	95.09%	95.09%
0.95	96.81%	97.24%	96.50%	95.83%	97.19%	97.03%	97.03%
AVG	91.61%	91.57%	91.41%	90.25%	91.58%	91.50%	91.66%

In the following, we report some interesting results obtained from the different methods in case of *normal situation where the class distribution is obtained from real-world data*.

We notice that increasing iteration number of previous method does not necessary increase the accuracy of CDE-iterate-n, as shown in Table 3. Thus, we stop at the iteration number 2 (Iterate2).

CDE- EM- BM method does not yield good result as expected. We notice this mostly due to error that usually appears in the prediction of model built from similar class distribution of train data. Since CDE- EM- BM often chooses inappropriate model built from class distribution different from test data than desired.

CDE-EM-AVGBM method while not perform badly like CDE- EM- BM but still not show any sign of improvement neither. We presume this happen because the error in model prediction that mention earlier have negative impact on this method even if using average value improve some accuracy of it.

We found CDE-EM-AVG performance is better than above methods. This means using average value can improve overall accuracy. However, due to burden from the model that built from cost of class distribution very different from test data, the accuracy is not improved much.

CDE-EM-EX method performs quite well we presume that using many models can reduce some bias and by remove model that really not suitable for actual data we can improve some accuracy. Moreover after calculate measure of robustness, we found

that our CDE-EM-EX method is more robust than the previous methods since it shows less average mismatch from Oracle method. Experiment result show average mismatch in Magic Gamma of CDE-Iterate-2 is 0.5641% while CDE-EM-EX is 0.2496% and in Adult CDE-Iterate-2 is 0.6737% while CDE-EM-EX is 0.2252%.

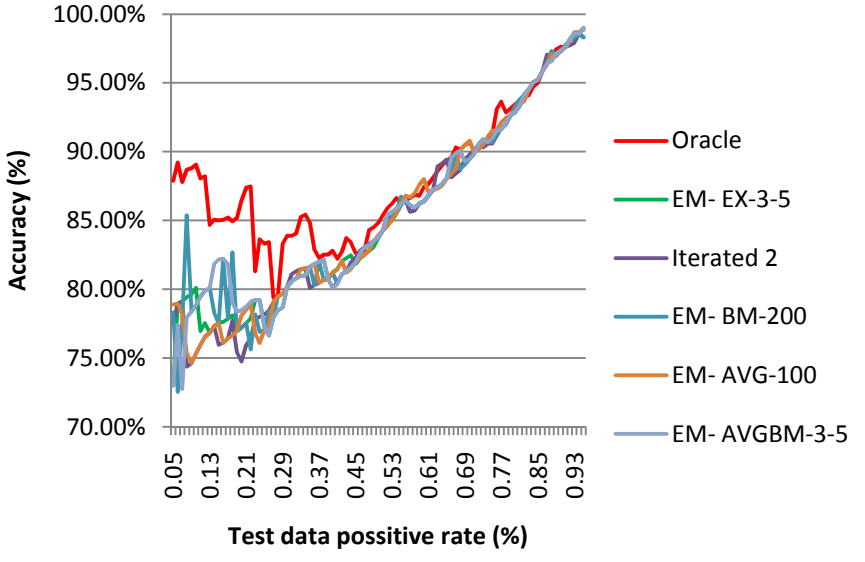


Fig. 5. Experiment 3: Accuracy Performance on Spambase (pr 95%) Dataset

Table 4. Experiment 3: Accuracy Performance on Spambase (pr 95%) Dataset

		CDE-based methods		Our methods			
Pr (%)	Oracle	Iterate 1	Iterate 2	EM- BM-200	EM- AVG-100	AVGBM-3-5	EM- EX-3-5
0.05	87.89%	72.84%	77.58%	78.32%	78.89%	73.00%	73.00%
0.1	89.05%	72.00%	75.32%	78.84%	75.32%	78.84%	80.11%
0.2	86.42%	74.00%	74.74%	77.21%	78.11%	78.47%	77.21%
0.3	83.89%	78.26%	80.16%	80.16%	80.16%	80.16%	80.16%
0.4	82.79%	79.84%	80.11%	81.21%	81.21%	80.11%	81.21%
0.5	84.84%	83.37%	83.79%	83.79%	83.95%	83.79%	83.74%
0.6	87.42%	86.26%	86.37%	86.37%	88.00%	86.37%	86.37%
0.7	90.74%	88.63%	89.79%	89.47%	90.79%	89.47%	89.47%
0.8	93.42%	93.37%	92.84%	93.32%	92.95%	92.84%	92.84%
0.9	97.63%	97.00%	97.26%	97.37%	97.37%	97.37%	97.37%
0.95	98.95%	98.68%	98.32%	98.32%	98.84%	99.00%	99.00%
AVG	88.11%	83.88%	85.05%	85.48%	85.24%	85.57%	85.36%

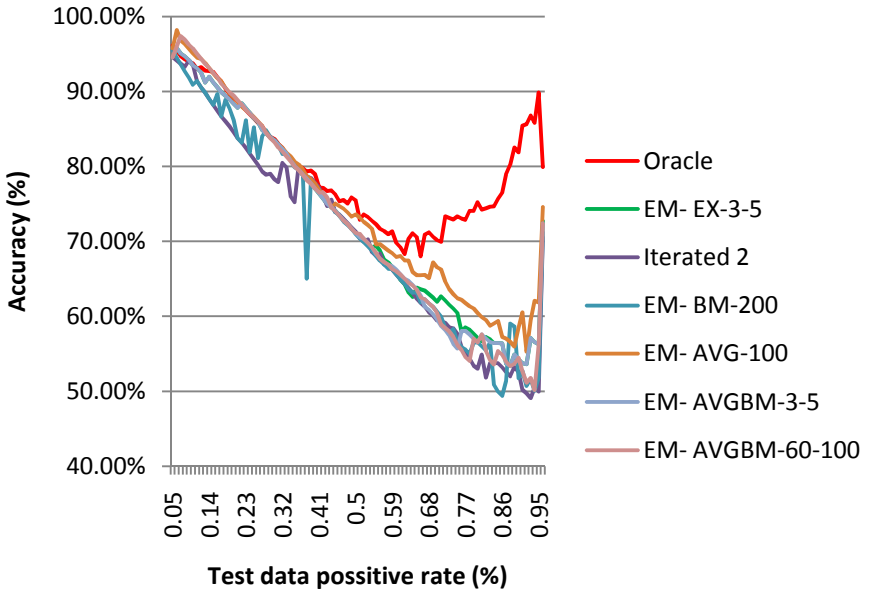


Fig. 6. Experiment 4: Accuracy Performance on Magic Gamma (pr 5%) Dataset

Table 5. Experiment 4: Accuracy Performance on Magic Gamma (pr 5%) Dataset

		CDE-based methods		Our methods			
Pr (%)	Oracle	Iterate 1	Iterate2	EM- BM-200	EM- AVG-100	AVGBM-3-5	EM- EX-3-5
0.05	96.14%	95.44%	94.60%	95.44%	95.74%	94.54%	94.54%
0.1	93.73%	90.90%	93.61%	90.90%	95.13%	93.50%	93.50%
0.2	89.04%	84.53%	84.53%	86.16%	89.19%	88.34%	88.34%
0.3	83.67%	77.44%	78.30%	83.41%	83.36%	83.49%	83.49%
0.4	78.97%	76.03%	77.79%	77.37%	77.76%	77.10%	77.10%
0.5	75.46%	69.29%	71.31%	70.94%	73.61%	71.07%	71.07%
0.6	69.80%	65.49%	65.63%	65.63%	67.91%	66.21%	65.87%
0.7	70.20%	58.31%	59.46%	60.63%	66.56%	59.63%	61.94%
0.8	75.21%	54.83%	53.00%	56.51%	60.43%	56.51%	57.16%
0.9	81.86%	47.99%	52.59%	51.74%	58.50%	54.27%	54.27%
0.95	89.86%	44.03%	49.93%	51.87%	61.86%	56.27%	56.27%
AVG	79.91%	69.26%	70.57%	71.21%	74.58%	72.41%	72.75%

From experiment 3 and 4 we show that previous methods are not efficient if class distribution of original data (train data) is too different from new data (test data). However our methods shows different result, while not yet achieves the same performance as Oracle, all of them shows an improvement from previous methods to a certain point. This indicates that combining CDE method with ensemble method can solve the biasing problem thus obtain better performance.

6 Conclusion

In this paper, we evaluated several methods for dealing with the situation where there is a change in class distribution after a classifier is built. Our results indicate that the more class distribution of test data is different from train data the more degrading in performance of CDE method. To solve this problem which is the result from biasing toward class distribution of train data, we induce ensemble method that combines many models instead of one resulting in less bias toward class distribution of train data and better classifier's performance. Moreover, CDE iterate-N method is not suitable for very large dataset since CDE iterate-N method does not support parallel programming because it requires result from previous iterated to start new iterate thus lead to data dependencies problem while our methods is more suitable since it can distribute work from each model without any dependency.

References

1. Alaiz-Rodriguez, R., Guerrero-Curieses, A., Cid-Sueiro, J.: Minimax.: Regret Classifier for Imprecise Class Distributions. Journal of Machine Learning Research 8 (2007)
2. Alaiz-Rodríguez, R., Japkowicz, N.: Assessing the Impact of Changing Environments on Classifier Performance. In: Bergler, S. (ed.) Canadian AI. LNCS (LNAI), vol. 5032, pp. 13–24. Springer, Heidelberg (2008)
3. Asuncion, A., Newman, D.J.: UCI Machine Learning Repository, (2007), http://www.ics.uci.edu/~mlearn/MLReposito-ry.html
4. Chapelle, O., Scholkopf, B., Zien, A.: Semi-Supervised Learning. MIT Press, Cambridge (2006)
5. Chawla, N.V., Bowyer, K.W., Kegelmeyer, W.P.: SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16 (2002)
6. Elkan, C.: The foundations of cost-sensitive learning. In: The Proceedings of the 17th International Joint Conference on Artificial Intelligence, pp. 973–978 (2001)
7. Forman, G.: Quantifying counts and costs via classification. Data Mining Knowledge Discovery 17(2) (2008)
8. Forman, G.: Counting Positives Accurately Despite Inaccurate Classification. In: Gama, J., Camacho, R., Brazdil, P.B., Jorge, A.M., Torgo, L. (eds.) ECML 2005. LNCS (LNAI), vol. 3720, pp. 564–575. Springer, Heidelberg (2005)
9. Latinne, P., Saerens, M., Decaestecker, C.: Adjusting the Outputs of a Classifier to New a Priori Probabilities May Significantly Improve Classification Accuracy: Evidence from a multi-class problem in remote sensing. In: The Proceedings of the 18th International Conference on Machine Learning, pp. 298–305 (2001)

10. Provost, F., Fawcett, T.: Robust Classification for Imprecise Environments. Machine Learning 42(3) (2001)
11. Provost, F., Fawcett, T., Kohavi, R.: The Case against Accuracy Estimation for Comparing Induction Algorithms. In: The Proceedings of the 15th International Conference on Machine Learning, pp. 445–453 (1998)
12. Quinlan, R.J.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
13. Rokach, L.: Ensemble-based classifiers. Artificial Intelligence Review 33 (2010)
14. Saerens, M., Latinne, P., Decaestecker, C.: Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. Neural Computing 14 (2002)
15. Tsymbal, A.: The problem of concept drift: Definitions and related work. Computer Science Department. Trinity College Dublin (2004)
16. Weiss, G.M.: Mining with rarity: a unifying framework. SIGKDD Explorations Newsletter 6(1) (2004)
17. Weiss, G.M., Provost, F.: Learning when training data are costly: The effect of class distribution on tree induction. Journal of Artificial Intelligence Research 19 (2003)
18. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann (2005)
19. Xue, J.C., Weiss, G.M.: Quantification and semi-supervised classification methods for handling changes in class distribution. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 897–906. ACM, Paris (2009)
20. Zadrozny, B., Elkan, C.: Learning and making decisions when costs and probabilities are both unknown. In: The Proceedings of the 7th International Conference on Knowledge Discovery and Data Mining, pp. 204–213 (2001)

HUE-Stream: Evolution-Based Clustering Technique for Heterogeneous Data Streams with Uncertainty

Wicha Meesuksabai, Thanapat Kangkachit, and Kitsana Waiyamai

Department of Computer Engineering, Faculty of Engineering
Kasetsart University, Bangkok 10900, Thailand
{g5314550164,g51850410,fengknw}@ku.ac.th

Abstract. Evolution-based stream clustering method supports the monitoring and the change detection of clustering structures. E-Stream is an evolution-based stream clustering method that supports different types of clustering structure evolution which are appearance, disappearance, self-evolution, merge and split. This paper presents HUE-Stream which extends E-Stream in order to support uncertainty in heterogeneous data. A distance function, cluster representation and histogram management are introduced for the different types of clustering structure evolution. We evaluate effectiveness of HUE-Stream on real-world dataset KDDCup 1999 Network Intruision Detection. Experimental results show that HUE-Stream gives better cluster quality compared with UMicro.

Keywords: Uncertain data streams, Heterogeneous data, Clustering, Evolution-based clustering.

1 Introduction

Recently, clustering data streams has become a research topic of growing interest. One main characteristic of data streams is to have the infinite evolving structure and to be generated at rapid rate. We call a stream clustering method that supports the monitoring and the change detection of clustering structures *evolution-based stream clustering* method. Apart from its infinite data volume, data streams also contain error or only partially complete information, called *data uncertainty*. In this paper, we focus on developing an evolution-based stream clustering method that supports uncertainty in data.

Many techniques have been proposed for clustering data streams. Most research has focused on clustering techniques for numerical data [1, 2, 3, 7]. Few have been proposed to deal with heterogeneous data streams, including numeral and categorical attributes simultaneously [8, 9, 10]. However, very few have proposed to monitor and detect change of the clustering structures. [7] proposed an evolution-based clustering method for numerical data streams. [8] extends it by integrating both numerical and categorical data streams. However, both methods do not support uncertainty in data streams.

J. Tang et al. (Eds.): ADMA 2011, Part II, LNAI 7121, pp. 27–40, 2011.

To support uncertainty in data streams, Aggarwal et al. [4] introduced the uncertain clustering feature for cluster representation and proposed a technique named UMicro. Later, they continued to study the problem of high dimensional projected clustering of uncertain data streams [1]. LuMicro [5] technique has been proposed to improve clustering quality, with the support of uncertainty in numerical attributes and not in categorical attributes. However, all these works have not been proposed in the context of evolution-based stream clustering.

In this paper, we present an evolution-based stream clustering technique called *HUE-Stream* that supports uncertainty in both numerical and categorical attributes. HUE-Stream is extended from E-Stream technique [7] which is an evolution-based stream clustering technique that integrates both numerical data streams without support of categorical data and uncertainty. Distance function, cluster representation and histogram management are introduced for the different types of clustering structure evolution. A *distance function* with probability distribution of two objects is introduced to support uncertainty in categorical attributes. To detect change in clustering structure, the proposed distance function is used to merge clusters and find the closest cluster of a given incoming data and proposed histogram management to split cluster in categorical data. Experimental results show that HUE-Stream gives better cluster quality compared with UMicro.

The remaining of the paper is organized as follows. Section 2 introduces basic concepts and definitions. Section 3 presents our stream clustering algorithm called HUE-Stream. Section 4 compares the performance of HUE-Stream and UMicro with respect to the real-world dataset. Conclusions are discussed in Section 5.

2 Basic Concepts of Evolution-Based Stream Clustering with Uncertainty

In the following, some notations and definitions of evolution-based stream clustering with uncertainty are defined. First, we assume that we have a **data stream** consists in a set of $(n + c)$ dimensional tuples $X_1 \ldots X_k \ldots$ arriving at time stamps $T_1 \ldots T_k \ldots$. Each data point X_i contains n numerical attributes and c categorical attributes, denoted by $X_i = (x_i^1 \ldots x_i^n, x_i^{n+1} \ldots x_i^{n+c})$. The number of valid values for a categorical attribute x^{n+k} is V^k, where $1 \leq k \leq c$, and j-th valid value for x^{n+k} is $\mathrm{V_j}^{\mathrm{k}}$, where $1 \leq j \leq V^k$.

2.1 Tuple-Level and Dimension-Level Uncertainty

In fact, there are many existing categories of assumption to model the uncertainty in data stream [5]. In this paper, our assumption is mainly focus on discrete probability density function which has been widely used and easy to apply in practice.

For each an uncertain tuple X_i, its $|x_i^d|$ possible values in *d-th* dimension can be defined by probability distribution vector as $x_i^d = \left\{\left(x_{i1}^d, p\left(x_{i1}^d\right)\right), \left(x_{i2}^d, p\left(x_{i2}^d\right)\right), \ldots, \left(x_{ib}^d, p\left(x_{ib}^d\right)\right)\right\}$, and $\sum_{\mathrm{a=1}}^{\mathrm{b}} \mathrm{p}\left(\mathrm{x_{ia}^d}\right) \leq 1$.

Dimension-level uncertainty of the *j-th* dimension of a tuple X_i denoted by $U(x_i^j)$ can be defined as follows:

$$U(x_i^j) = \begin{cases} 0, & \exists x_{ib}^j \in x_i^j : p\left(x_{ib}^j\right) = 1 \\ -\sum_{a=1}^{b} p(x_{ia}^j) * \log_2\left(p(x_{ia}^j)\right), & \text{otherwise} ; \end{cases}$$

Let X_i be a tuple, a tuple-level uncertainty of X_i denoted by $U(X_i)$ is an average of its dimension-level uncertainties defined as $U(X_i) = \frac{1}{n+c} * \sum_{j=1}^{n+c} U(x_i^j)$;

Therefore, uncertainty of all *k*-tuples data streams can be calculated as average of their tuple-level uncertainties.

2.2 Cluster Representation Using Fading Cluster Structure with Histogram

A Fading Cluster Structure with Histogram (FCH) has been introduced in [9]. In this paper, FCH is extended to support uncertainty in both numerical and categorical data.

An uncertain cluster feature for a cluster C contained a set of tuples $X_i \dots X_k$ arriving at time stamps $T_i \dots T_k$ is defined as FCH = ($\overrightarrow{FC1(C,t)}$, $\overrightarrow{FC2(C,t)}$, $W(C,t)$, $U(C,t)$, $\overrightarrow{HN(C,t)}$, $\overrightarrow{HCP(C,t)}$) as described below.

$\overrightarrow{\boldsymbol{FC1(C,t)}}$ is a vector of weighted sum of tuple expectation for each dimension at time *t*, i.e., the value of the j^{th} entry is $\sum_{i=1}^{K}\left(\mathrm{f(t-T_i)} * \sum_{a=1}^{|x_i^j|}\left(x_{i,a}^j * p\left(x_{i,a}^j\right)\right)\right)$, denoted by $FC1^j(C,t)$.

$\overrightarrow{\boldsymbol{FC2(C,t)}}$ is a vector of weighted sum of squares of tuple expectation for each dimension at time *t*, i.e., the value of the j^{th} entry denoted by $\boldsymbol{FC2^j(C,t)}$ is $\sum_{i=1}^{K}\left(\boldsymbol{f(t-T_i)} * \sum_{a=1}^{|x_i^j|}\left(\boldsymbol{x_{i,a}^j} * \boldsymbol{p\left(x_{i,a}^j\right)}\right)^2\right)$

$\boldsymbol{W(C,t)}$ is a sum of all weights of data points in a cluster C at time *t*, i.e., $W(C,t) = \sum_{i=1}^{K} f(t-T_i)$

$\boldsymbol{U(C,t)}$ is sum of all weight of tuple uncertainty in cluster C at time *t*, i.e., $\boldsymbol{U(C,t) = \sum_{i=1}^{K} f(t-T_i) * U(X_i)}$

$\overrightarrow{\boldsymbol{HN(C,t)}}$ is a α-bin histogram of numerical data values with α equal width intervals, i.e., the *j*-th numerical dimension with *l*-th bin histogram of $HN(t)$ at time *t* is $\sum_{i=1}^{K}\left(f(t-T_i) * \sum_{a=1}^{|x_i^j|}\left(x_{i,a}^j * p\left(x_{i,a}^j\right)\right) * y_{i,l}^j\right)$ where

$$y_{i,l}^j = \begin{cases} 1 & if\ l \cdot r + min(x^j) \le x_i^j \le (l+1) \cdot r + min(x^j);\ r = \frac{max(x^j) - min(x^j)}{\alpha} \\ 0 & otherwise \end{cases}$$

$\overrightarrow{\boldsymbol{HC(C,t)}}$ is a β -bin histogram of categorical data values with top β accumulated probabilities of valid categorical values and their weighted frequency for each categorical dimension at time *t*. The histogram of *j*-th categorical dimension of C is

$\overrightarrow{HC_j(C,t)} = \{(V_{C,1}^j, hcp_1^j(C,t), hcf_1^j(C,t), \dots, (V_{C,\beta}^j, hcp_\beta^j(C,t), hcf_\beta^j(C,t))\}$

where V_a^j is an *a*-th valid categorical value in *j*-th categorical dimension,

$hcp_a^j(C,t)$ is accumulated probability of V_a^j which can be formulated as

$$hcp_a^j(C,t) = \sum_{i=1}^{k}\sum_{v=1}^{\beta}\left(f(t-T_i) * p\left(x_{i,v}^j\right) * z\left(x_{i,v}^j, V_a^j\right)\right),$$

$$\text{where } z\left(x_{i,v}^j, V_a^j\right) = \begin{cases} 1 & if \quad x_{i,a}^j = V_a^j \\ 0 & otherwise \end{cases}.$$

and $hcf_a^j(C,t)$ is an accumulated weight frequency of V_a^j which can be formulated as $hcf_a^j(C,t) = (\sum_{i=1}^{k}\sum_{v=1}^{\beta}\left(f(t-T_i) * z\left(x_{i,v}^j, V_a^j\right)\right)$.

Note that $hcp_a^j(C,t)$ and $hcf_a^j(C,t)$ are used in calculating distance between two set of categorical data as described in section 2.3.

2.3 Distance Functions

A distance function plays important role in data clustering tasks. To deal with uncertainty in both categorical and numerical data, we propose new distance functions that take into account the uncertainty as further described as follows.

Cluster-Point distance can be formulated as:

$$dist(C, X_i) = (\delta) * dist_{num}(C, X_i) + (1-\delta) * dist_{cat}(C, X_i) \tag{1}$$

where $dist_{num}(\mathrm{C},\mathrm{X})$ is the distance between expected center of numerical attributes in cluster *C*, denoted by $Center_C^j$, and expected value of *j*-th numerical attributes of data point denoted by $E[X^j]$ as

$$dist_{num}(\mathrm{C}, X_i) = \frac{1}{\mathrm{n}} * \sum_{\mathrm{j}=1}^{\mathrm{n}} \frac{\left|\mathrm{Center}_{\mathrm{C}}^{\mathrm{j}} - \mathrm{E}[\mathrm{X}_{\mathrm{i}}^{\mathrm{j}}]\right|}{4 * \mathrm{SD}_{\mathrm{C}}^{\mathrm{j}}} \tag{2}$$

where $\mathrm{E}[\mathrm{X}_{\mathrm{i}}^{\mathrm{j}}] = \sum_{\mathrm{a}=1}^{\mathrm{b}}\left(\mathrm{x}_{\mathrm{i,a}}^{\mathrm{j}} * \mathrm{p}\left(\mathrm{x}_{\mathrm{i,a}}^{\mathrm{j}}\right)\right)$, and $\mathrm{Center}_{\mathrm{C}}^{\mathrm{j}}$ and $\mathrm{SD}_{\mathrm{C}}^{\mathrm{j}}$ can be derived from mean and standard deviation of expected values $\mathrm{E}[\mathrm{X}_{\mathrm{i}}^{\mathrm{j}}]$ of all data points in cluster C;
$dist_{cat}(C, X_i)$ is the distance derived from categorical attributes:

$$dist_{cat}(C, X_i) = \frac{1}{c} * \sum_{j=1}^{c} \sqrt{\sum_{a=1}^{\beta}\sum_{d=1}^{|X_i^k|}\left(inters\left(C, x_{i,d}^j, j, a\right) + diff(C, x_{i,d}^j, j, a)\right)} \tag{3}$$

$$inters\left(C, x_{i,d}^j, j, a\right) = \left(\frac{hcp_a^j(C,t)}{hcf_a^j(C,t)} - f(t-T_i) * p\left(x_{i,d}^j\right)\right)^2 * z\left(V_{C,a}^j, x_{i,d}^j\right) \tag{4}$$

$$diff(C, x_{i,d}^j, j, a) = (f(t-T_i) * p\left(x_{i,d}^j\right))^2 * \left(1 - z\left(V_{C,a}^j, x_{i,d}^j\right)\right) \tag{5}$$

Cluster-Cluster distance can be formulated as:
$dist_{num}(\mathrm{C}_1, \mathrm{C}_2)$ is the distance between expected center of numerical attributes in cluster C_1 and C_2 which can be defined as:

$$dist_{num}(\mathrm{C}_1, \mathrm{C}_2) = \frac{1}{\mathrm{n}} * \sum_{\mathrm{j}=1}^{\mathrm{n}} \left| \mathrm{Center}_{\mathrm{C}_1}^{\mathrm{j}} - \mathrm{Center}_{\mathrm{C}_2}^{\mathrm{j}} \right| \tag{6}$$

and $dist_{cat}(\mathrm{C}_1, \mathrm{C}_2)$ is the distance derived from categorical attributes, which can be defined as:

$$dist_{cat}(\mathrm{C}_1, \mathrm{C}_2) = \frac{1}{c} * \sum_{j=1}^{c} \sqrt{\sum_{a=1}^{\beta} \left\{ \sum_{d=1}^{\beta} \{ intersC(\mathrm{C}_1, \mathrm{C}_2, j, a, d) + diffC(\mathrm{C}_1, \mathrm{C}_2, j, a, d) + diffC(\mathrm{C}_2, \mathrm{C}_1, j, a, d) \} \right\}} \tag{7}$$

where

$$intersC(\mathrm{C}_1, \mathrm{C}_2, j, a, d) = \left(\frac{hcp_a^j(\mathrm{C}_1, t)}{hcf_a^j(\mathrm{C}_1, t)} - \frac{hcp_d^j(\mathrm{C}_2, t)}{hcf_d^j(\mathrm{C}_2, t)} \right)^2 * z\left(V_{\mathrm{C}_1,a}^j, V_{\mathrm{C}_2,d}^j\right) \tag{8}$$

$$diffC(\mathrm{C}_1, \mathrm{C}_2, j, a, d) = (hcp_d^j(\mathrm{C}_2, t))^2 * \left(1 - z\left(V_{\mathrm{C}_1,a}^j, V_{\mathrm{C}_2,d}^j\right)\right) \tag{9}$$

2.4 Evolution-Based Stream Clustering

Evolution-based stream clustering method supports the monitoring and the change detection of clustering structures. A cluster is a collection of data that have been memorized for processing in the system. Each cluster can be classified into 3 types which are isolated data point, inactive cluster, or active cluster. Here we give description of each type of cluster:

- An *isolated data point* is data that cannot be clustering to any cluster. The system may be used for further consideration with density increases in its region and formed a new cluster or fade its weight until it disappears.
- An *inactive cluster* is isolate data point or cluster that has a low weight. It is caused by an aggregation of isolate data point or fading decreases weight of active cluster.
- An *active cluster* is cluster that is considered a model of the system.

Fading Function

In order to capture the clustering characteristics of the data streams which evolve over time, we need to focus on new data more than on old data. To decrease weight of old data over time, a fading function is used. Let λ be the decay rate and t be the elapsed time; the fading function can be defined as follows:

$$f(t) = 2^{-\lambda t} \tag{10}$$

Weight of a cluster is the number of data elements in a cluster. Weight is determined according to the fading function. Initially, each data element has a weight of 10. A cluster can increase its weight by assembling incoming data points or merging with other clusters.

Evolution of Stream Clustering Structure

To capture the characteristics of data streams which have an evolving nature, evolution-based stream clustering method is composed of different steps. In the beginning, incoming data are considered as isolated clusters. Then, a cluster is formed when a sufficiently dense region appears. An inactive cluster is changed to active cluster when its weight reaches a given threshold. When a set of clusters have been identified, incoming data must be assigned to a closest cluster based on similarity score.

To detect change in clustering structure, the following clustering evolutions are checked and handled: *appearance*, *disappearance*, *self-evolution*, *merge* and *split*.

Appearance: The appearance of a new cluster occurs when there is a group of dense data points locating close to each other by considering the point-point distance and cluster-point distance as in section 2.3, respectively.

Disappearance: The disappearance of existing clusters happen when their data points having least recent time stamp and their weights are less than the *remove_threshold*.

Self-Evolution: In each active cluster, its characteristics can be evolved over time because the weight of old data points is faded so that the new data points can affect the characteristic of the cluster in a short time. By doing this, we use fading equation 10 and Cluster-Point distance to find the closet cluster of a new data point which will change the cluster characteristic.

Merge: Two overlap clusters, denoted by C_1 and C_2, can be merged into a new cluster by considering the distance between two clusters, Cluster-Cluster distance. Since the characteristics of two overlap clusters should exist in the merged cluster, *FCH* of merged cluster can be formulated as follows:

Let C_1 and C_2 denote two sets of cluster. $FCH(C_1 \cup C_2)$ can be calculated based on $FCH(C_1)$ and $FCH(C_2)$

- The values of entires $\overline{FC1(C,t)}$, $\overline{FC2(C,t)}$, $W(C,t)$, $U(C,t)$in *FCH* are the sum of the corresponding entires in $FCH(C_1)$ and $FCH(C_2)$.
- To obtain merged histogram of numerical data values,$\overline{HN_{merged}(t)}$, we first find the minimum and maximum value in each numerical dimension of the pair. Then we divide this range into α intervals with equal length. Finally, we compute the frequency of each merged interval from the histogram of the pair.
- For calculating merged histogram of categorical data values, $\overline{HC_{merged}(t)}$, we union two set of top α-bin categorical data value in each dimension of the pair. Then, we order the union set by its frequency in descending order. Finally, we store only top β-bin categorical data values into $\overline{HC_{merged}(t)}$.

Split: A cluster can be splitted into two smaller clusters when its inside behaviour is obviously separated. All attributes are verified to find the split-position. If split-position occurs in numerical or categorical attribute, the weight will be recalculated based on histogram of the splitting attribute. Then, cluster representation of the new clusters is determined based on the calculated weight. Histogram is used for consideration on the split of behavior of each cluster. It can be found in both numerical and categorical attribute.

- For **numerical attributes**, a valley which lies between 2 peeks of histogram is considered as splitting criteria. If the splitting valley is found more than one point, the best splitting valley is the minimum value valley. When the cluster splits, we split histogram in that dimension and other dimensions are weighted based on the split dimension. The splitting valley must have statistical significantly lower than the lower peak.
- For **categorical attributes**, splitting-attribute is an attribute that has significant frequency than the others within the same cluster. Split-position is a position between a pair of adjacent values whose frequencies are the most different. If split-position occurs between first and second bars, we will not split that cluster into 2 small clusters because the first value is only one outstanding member in the top- β of the splitting-attribute.

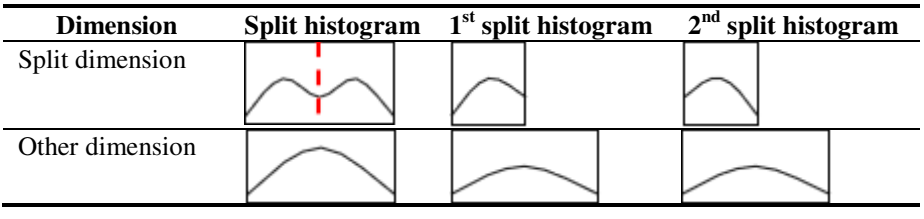


Fig. 1. Histogram management in a split dimension and other dimension of numerical

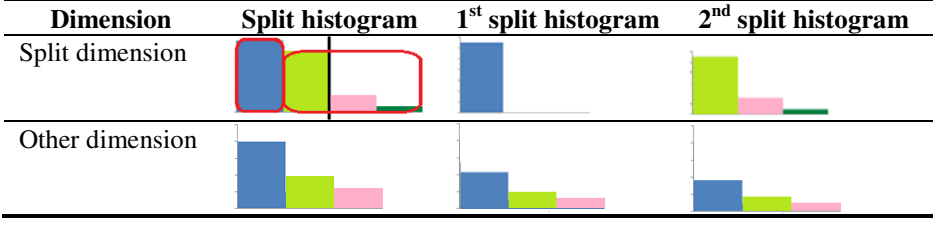


Fig. 2. Histogram management in a split dimension and other dimension of categorical

3 The Algorithm

In this section, we started discussing heterogeneous data streams with uncertainty and their evolution over time. Then, we present a new algorithm, called HUE-Stream which extended from E-Stream algorithm [7].

3.1 Overview of HUE-Stream Algorithm

This section describes HUE-Stream which is an extension of E-Stream [7] to support uncertainty in heterogeneous data stream.

Table 2 contains all the notations that are used in HUE-Stream algorithm. Pseudo-code of HUE-Stream is given in Fig. 3.

Table 1. List of notations used in HUE-Stream pseudo-code

Notation	Definition
|FCH|	Current number of clusters
FCH_i	i^{th} cluster
$FCH_i.W$	Weight of the i^{th} cluster
$FCH_i.sd$	Standard deviation of the i^{th} cluster
#isolate	Current number of isolated data
$isolate_i$	i^{th} isolated data
$minDist_{num}(FCH_a, FCH_b)$	Nearest pair of clusters when measured with numerical attributes
$minDist_{cat}(FCH_a, FCH_b)$	Nearest pair of clusters when measured with categorical attributes
$FCH_i.U$	Uncertainty of i^{th} cluster

HUE-Stream main algorithm is given in Fig. 3. HUE-Stream supports the monitoring and the change detection of clustering structures that can evolve over time. Five types of clustering structure evolution are supported which are appearance, disappearance, self-evolution, merge and split.

In line 1, the algorithm starts by retrieving a new data point. In line 2 and 3, it fades all clusters and deletes any clusters having insufficient weight. In line 4, it splits a cluster when the behavior inside the cluster is obviously separated. In line 5, it merges a pair of clusters for which the characteristics are very similar. In line 6, it checks the number of clusters and merges the closest pairs if the number of cluster exceeds the limit. In line 7, it scans cluster in the system to find out active clusters that reach sufficient weight. In line 8 to 13, it finds the closest cluster to contain the incoming data point with respect to the distance and uncertainty of that cluster. Isolate data point will be created where there is no cluster to contain the new data point. The flow of control then returns to the top of the algorithm and waits for a new data point.

```
Algorithm HUE-Stream
1 retrieve new data X_i
2 FadingAll
3 DeleteCluster
4 CheckSplit
5 MergeOverlapCluster
6 LimitMaximumCluster
7 FlagActiveCluster
8 (Uncertainty[], index[]) ← FindCandidateClosestCluster
9  if sizeof(index[]) > 0
10        index ← FindClosestCluster
11        add x_i to FCH_index
12 else
13        create new FCH from X_i
14 waiting for new data
```

Fig. 3. HUE-Stream algorithm

Procedure *MergeOverlapCluster*

```
1 for i ← 1 to |FCH|
2   for j ← i + 1 to |FCH|
3     overlap[i,j] ← dist(FCHi,FCHj)
4     m ← merge_threshold
5     if overlap[i,j] > m*(FCHi.sd+FCHj.sd)
6       if dist_cat[i,j] < m *minDist_cat(FCH_a, FCH_b)
7         if (i, j) not in S
8           merge(FCH_i, FCH_j)
```

Procedure *CheckSplit*

```
1 for i ← 1 to |FCH|
2     for j ← 1 to number of numerical attributes
3       find valley and peek
4       if chi-square test(valley,peek) > significance
5         split using numerical attribute
6     for j ← 1 to number of categorical attributes
7       find maximum different of bin_k, bin_k+1
8       if chi-square test(bin_k,bin_k+1) > significance
9         split using categorical attribute
10   if split using only numerical or categorical
11       split FCH_i
12       S ← S ∪ {(I,|FCH|)}
13   else if numerical and categorical
14       (n1,n2) ← split using numerical
15       (c1,c2) ← split using categorical
16       if max(c1,c2) > max(n1,n2)
17         split FCH_i using categorical
18       if max(c1,c2) <= max(n1,n2)
19         split FCH_i using numerical
20       S ← S ∪ {(i,|FCH|)}
```

Procedure *FindCandidateClosestCluster*

```
1 for i ← 1 to |FCH|
2   if FCH_i is active cluster
3     dist[i] ←dist(FCH_i, x_i)
4     if dist[i] < radius_factor/4
5       set_candidate ← i
6 return set_candidate
```

Procedure *FindClosestCluster*

```
1 for i ← 1 to |set_candidate|
2   index_u[i, ΔFCH_i.U] ← ΔFCH_i.U
3 index[i, ΔFCH_i.U] ← max_of_ΔFCH_i.U
4 return i
```

Fig. 4. MergeOverlapCluster, CheckSplit, FindCandidateClosestCluster and FindClosestCluster

In the following, details of each step are described.

FadingAll performs fading of all the existing clusters in the system. New data points are preferred to old data points. When a cluster in the system has insufficient weight ($FCH_i.W$ < fade_threshild), it will be deleted from the system.

LimitMaximumCluster: This procedure is used to limit the number of cluster. It checks whether the number of clusters is not greater than its "maximum_cluster". If the number of clusters exceeds then the closest pair of clusters will be merged until the number of remaining clusters is less than or equal to the threshold.

FlagActiveCluster: This procedure is used to check status of the current active cluster. When weight of any cluster is greater or equal to "active_threshold" then it will be flagged as an active cluster. Otherwise, the flag is cleared.

CheckSplit: This procedure is used to verify the splitting condition of each cluster using the histogram. If a splitting point is found in any cluster then it is splitted. This procedure is shown in Fig. 4**.** Splitting condition is verified for both numerical and categorical attribute based on weight of the resulting clusters.

MergeOverlapCluster: This procedure scans all the clusters in system for merging pairs of similar clusters. If the cluster-to-cluster distance is less than he "merge_threshold" then couple of those clusters will be merged.

FindCandidateClosestCluster and FindClosestCluster: This procedure is used to compute the distance in order to find index of candidate closest to active cluster and choose candidate the most appropriate by computing maximum uncertainty value changes $\Delta \boldsymbol{U}(\boldsymbol{C_i}) = \frac{\boldsymbol{U}(\boldsymbol{C_i})}{\boldsymbol{W_i}} - \frac{\boldsymbol{U}(\boldsymbol{C_i} \cup \boldsymbol{X})}{\mathbf{W_i+1}}$.

4 Experimental Results

In this section, we present experimental results of HUE-Stream clustering method.

We compare HUE-Stream with UMicro which is designed to tackle the challenges of clustering over uncertain data stream. We compare the two methods in terms of effectiveness (accuracy with progression of stream, accuracy with increasing level of uncertainty), sensitivity (accuracy with varying number of clusters) and efficiency (processing time and number of data points proceeded per second).

Table 2. Parameter settings of HUE-Stream and UMicro algorithms

HUE-Stream	UMicro
maximum_cluster 26	num_cluster 100
stream_speed 100	stream_speed 100
decay_rate 0.1	decay_rate 0.1
radius_factor 4	radius_factor 3
remove_threshold 0.1	
merge_threshold 1.25	
active_threshold 5	

In order to test algorithm, we make attributes uncertain by converting them into probability vectors. For numerical attributes, the converting technical described in [5] is used. For categorical attributes, the converting technical described in [6] is used.

For the dataset, we use KDD-CUP'99 (real-world dataset) which is a benchmark in UCI KDD Archive [11]. KDD-CUP'99 contains 494020 records in which each record is composed of 34 numerical attributes and 7 categorical attributes.

In this paper, all the experiments run on a PC with a 3.2 GHz Intel® Core™ i5 CPU, with the Debian GNU/Linux 5.0.8, and the memory size is 8GB. We compare the HUE-Stream algorithm with UMicro algorithm which both algorithms is implemented in C++.

Parameter settings of the two algorithms are shown in Table. 2.

4.1 Effectiveness Test

We evaluate the effectiveness of HUE-Stream and UMicro in terms of purity and f-measure. Fig. 5 and 6 show purity and f-measure of the two methods with increasing progression of the streams with time horizon = 2 and 500, respectively. In each case, the uncertainty level is set to 5%. In terms of purity, we obtain comparable results. However, HUE-Stream has purity less than UMicro for the time horizon equal to 500 within interval 100001 and 150000. This can be explained by the fact that UMicro generates too many numbers of clusters (UMicro generates 9919 clusters, while HUE-Stream generates only 1297 clusters). In terms of f-measure, experimental results show that HUE-Stream outperforms UMicro. UMicro generates too many small clusters, especially in range between 200000 and 300000, while HUE-Stream generates only one cluster. There is only one class in the actual data.

Fig. 7, shows purity and f-measure with increasing uncertainty level. It is clear that in each level, purity and f-measure reduced with increasing level of uncertainty.

4.2 Sensitivity Test

We evaluate the sensitivity of HUE-Stream and UMicro in terms of purity and f-measure. We would like to know how sensitive the method for increasing number of clusters is. To be more explicit, we increase the number of clusters by doubling its number to 13, 26, 52 and 104.

Fig. 8. shows that the purity is not sensitive to the number of clusters for the two methods. However, this is not the same for f-measure.

For UMicro, its f-measure has a tendency to decrease when increasing the number of clusters. However, HUE-Stream is not sensitive to the increasing number of clusters because its number of clusters is not fixed but depends on the behavior of data streams. As long as the maximum number of clusters is not exceeded, HUE-Stream still yields good results.

4.3 Efficiency Test

We evaluate the efficiency of HUE-Stream and UMicro in terms of number of data points proceeded per second and processing time. Fig. 9. shows that the number of data points proceeded per second by UMicro is constant. However, for HUE-Stream, this number is not constant but depends on behavior of data streams over time. Notice that in range 200000-300000, the number of data points proceeded per second is very high because the actual dataset contains only one class and merge or split operations are not necessary. Fig. 9. shows runtime of the two methods. Both algorithms exhibit

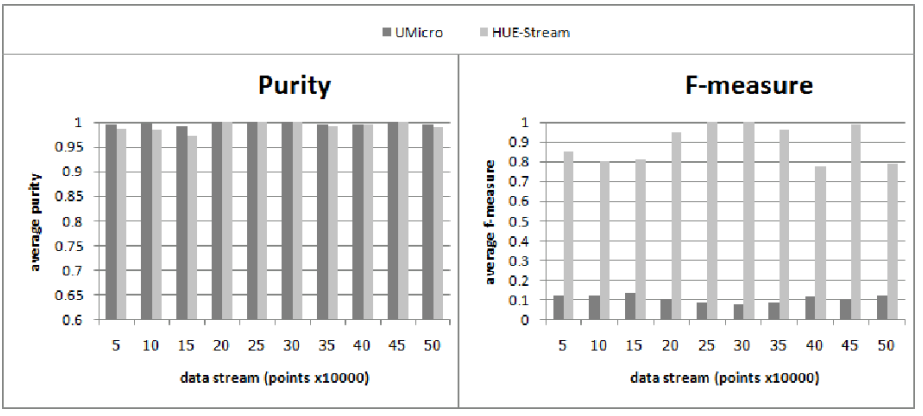


Fig. 5. Purity and f-measure with progression of stream, horizon =2

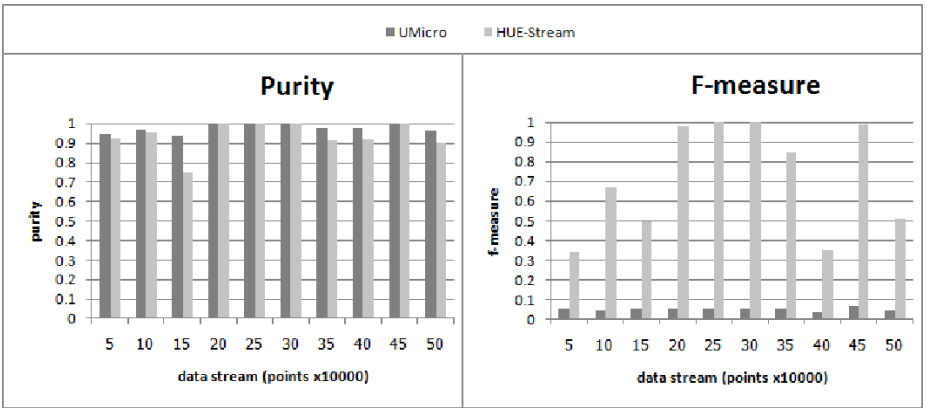


Fig. 6. Purity and f-measure with progression of stream, horizon =500

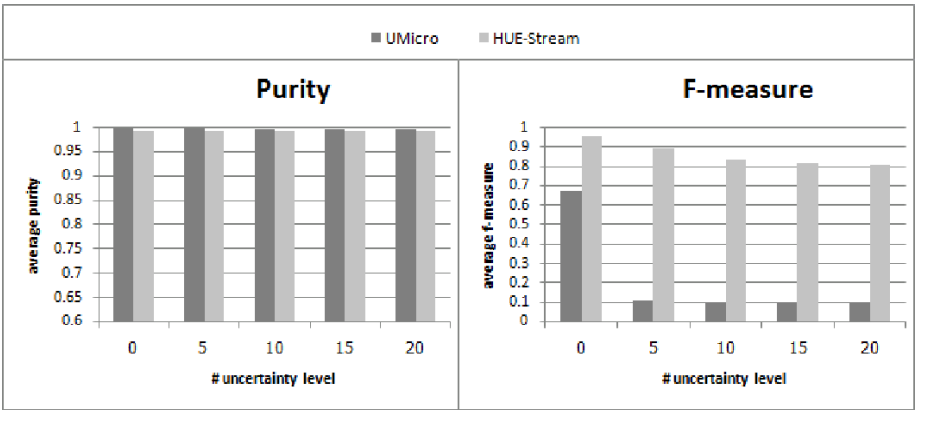


Fig. 7. Purity and f-measure with increase in uncertain level, horizon =2

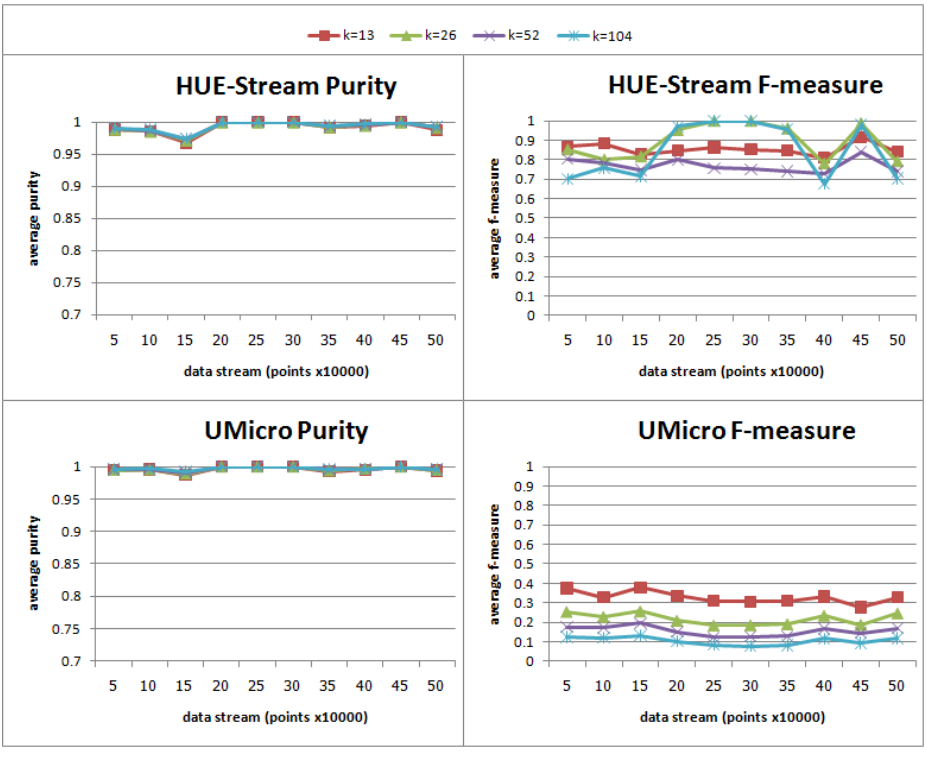


Fig. 8. Sensitivity with number of cluster, horizon =2

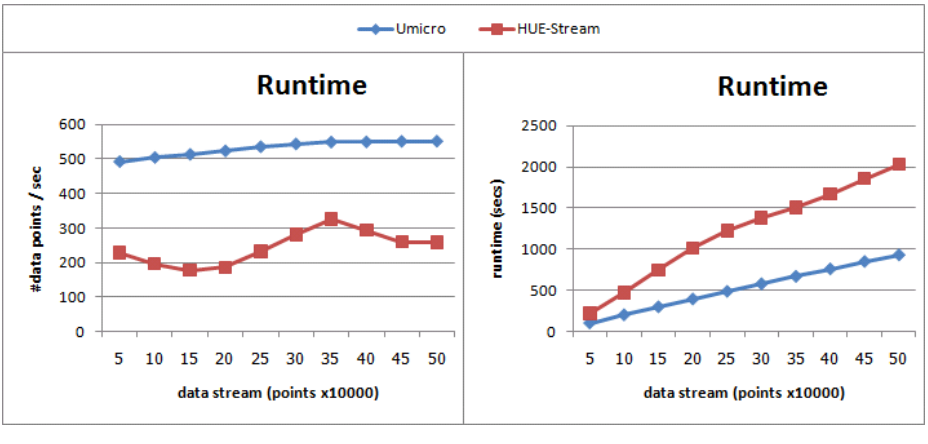


Fig. 9. Efficiency of stream clustering, horizon =2

linear runtime in number of data points. However, efficiency of UMicro is higher than that of HUE-Stream with a constant.

5 Conclusions

The uncertainty in the data stream significantly affects the clustering structure. In this paper, we proposed HUE-Stream algorithm for clustering heterogeneous data streams with uncertainty. A distance function, cluster representation and histogram management are introduced for supporting different types of clustering structure evolution which are appearance, disappearance, self-evolution, merge and split.

We compared HUE-Stream on real-world dataset against UMicro in terms effectiveness, sensitivity and efficiency. HUE-Stream algorithm outperforms UMicro in terms f-measure, and has comparable purity. While UMicro is sensitive to the input parameter number of clusters, HUE-Stream is robust and able to determine the almost exact number of clusters that suits with based on the behavior of the data stream and the number of cluster in the system. Both algorithms exhibit linear runtime in number of data points. However, efficiency of UMicro is higher than that of HUE-Stream with a constant.

References

1. Aggarwal, C.C.: On High Dimensional Projected Clustering of Uncertain Data Streams. In: IEEE 25th International Conference on Data Engineering, ICDE 2009, March 29 - April 2, pp. 1152–1154 (2009)
2. Aggarwal, C.C., Han, J., Wang, J., Yu, P.S.: A framework for clustering evolving data streams. Paper presented at the Proceedings of the 29th International Conference on Very Large Data Bases, Berlin, Germany, vol. 29 (2003)
3. Aggarwal, C.C., Han, J., Wang, J., Yu, P.S.: A framework for projected clustering of high dimensional data streams. Paper presented at the Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada, vol. 30 (2004)
4. Aggarwal, C.C., Yu, P.S.: A Framework for Clustering Uncertain Data Streams. In: IEEE 24th International Conference on Data Engineering, ICDE 2008, April 7-12, pp. 150–159 (2008)
5. Chen, Z., Ming, G., Aoying, Z.: Tracking High Quality Clusters over Uncertain Data Streams. In: IEEE 25th International Conference on Data Engineering, ICDE 2009, March 29 - April 2, pp. 1641–1648 (2009)
6. Qin, B., Xia, Y., Prabhakar, S., Tu, Y.: A Rule-Based Classification Algorithm for Uncertain Data. Paper presented at the Proceedings of the 2009 IEEE International Conference on Data Engineering (2009)
7. Udommanetanakit, K., Rakthanmanon, T., Waiyamai, K.: E-Stream: Evolution-Based Technique for Stream Clustering. In: Alhajj, R., Gao, H., Li, X., Li, J., Zaïane, O.R. (eds.) ADMA 2007. LNCS (LNAI), vol. 4632, pp. 605–615. Springer, Heidelberg (2007)
8. Kosonpothisakun, P., Kangkachit, T., Waiyamai, K.: E-Stream++: Stream clustering technique for supporting numerical and categorical data. Paper presented at the Proceedings of the 13th National Computer Science and Engineering Conference, Bangkok, Thailand (2009)
9. Yang, C., Zhou, J.: HClustream: A Novel Approach for Clustering Evolving Heterogeneous Data Stream. Paper presented at the Proceedings of the Sixth IEEE International Conference on Data Mining - Workshops (2006)
10. Huang, G.Y., Liang, D.P., Hu, C.Z., Ren, J.D.: An algorithm for clustering heterogeneous data streams with uncertainty. Paper presented at the Proceedings of International Conference on Machine Learning and Computing, Qingdao, China (2010)
11. The network intrusion detection data set, `http://archive.ics.uci.edu/ml/datasets/`

Hybrid Artificial Immune Algorithm and CMAC Neural Network Classifier for Supporting Business and Medical Decision Making

Jui-Yu Wu*

Department of Business Administration,
Lunghwa University of Science and Technology, Taiwan
jywu@mail.lhu.edu.tw

Abstract. Decision making that involves credit scoring and medical diagnosis can be considered to solve classification problems. Among the many data miming (DM) methods that have been developed to solve these classification problems are neural network (NN) and support vector machine (SVM) classifiers. Despite their successful application to classification problems, these classifiers are limited, in that users must use trial-and error to modify specific parameter settings. Fortunately, the setting of the parameters for those classifiers can be viewed as an unconstrained global optimization problem. To overcome this limitation of those classifiers, this work develops an advanced DM method that combines an artificial immune algorithm (AIA) and a MIMO cerebellar model articulation controller NN (CMAC NN) classifier (AIA-MIMO CMAC NN classifier). The AIA is a stochastic global optimization method and its parameters are easily set. The proposed CMAC NN classifier is characterized by its fast learning, reasonable generalization ability and robust noise resistance. The proposed AIA-MIMO CMAC NN classifier uses an outer AIA to optimize the parameter settings of an inner MIMO CMAC NN classifier, which is used to solve classification problems. The performance of the proposed classifier is also evaluated using a set of real-world classification problems, such as credit scoring and medical diagnosis. Moreover, this work compares the numerical results obtained using the proposed AIA-MIMO CMAC NN classifier with those obtained using published classifiers (such as SVM, SVM-based classifiers, NN classifiers and C4.5). Experimental results indicate that the classification accuracy of the proposed AIA-MIMO CMAC NN classifier is superior to those of some published classifiers. Hence, the AIA-MIMO CMAC NN classifier can be viewed an alternative DM method for supporting business and medical decision making.

Keywords: data mining, classification, artificial immune algorithm, neural network classifier, cerebellar model articulation controller.

1 Introduction

Knowledge discovery in databases (KDD) transforms low-level data into high-level knowledge. The KDD process consists of data preparation, data selection, data

* Corresponding author.

J. Tang et al. (Eds.): ADMA 2011, Part II, LNAI 7121, pp. 41–54, 2011.

cleaning, incorporation of prior knowledge, data mining (DM) and proper interpretation of the DM findings. DM is the core method in the KDD process. DM algorithms consist of the model, preference criterion and search algorithm [1]. The model includes a purpose (such as classification, regression, clustering, rule generation, discovery of an association rule, summarization, dependency modeling and time series analysis) and the technique (such as statistics-based methods, neural networks (NNs), genetic algorithms (GAs) and fuzzy logic). The criterion usually arises from some goodness-of-fit function of the model to the data. The search algorithm is used to select the models and parameters, given the data, model(s) and a preference criterion.

Zadeh [2] coined the term "soft computing", which refers to the synergistic power of two or more fused computational intelligence schemes, such as GAs, particle swarm optimization (PSO) algorithms, artificial immune algorithms (AIAs) and NNs [3]. In the past decade, the development of hybrid intelligent systems has become a promising field of research. Hybrid algorithms have some advantages [4]. For instance, hybrid algorithms outperform individual algorithms in solving certain problems, and they can solve general problems more efficiently. Many hybrid intelligent algorithms have been developed. For instance, Valdez et al. [5] presented a hybrid GA and PSO algorithm, and used fuzzy logic to combine these two methods and tune their parameters. The proposed method was used to solve ten benchmark function optimization problems. Abd-El-Wahed et al. [6] developed a combined PSO and GA to solve nonlinear optimization problems. Kao and Han [7] presented a hybrid GA and PSO algorithm for bi-level linear programming to solve a supply chain distribution problem. Huang et al. [8] developed three hybrid support vector machine (SVM)-based models to the credit scoring of applicants, indicating that the SVM-based classifiers perform similar to back-propagation NN (BPNN) and genetic programming (GP). Although hybrid algorithms are more effective than individual algorithms in solving problems, they have a critical limitation: they have a high complexity.

Decision making that involves credit scoring and medical diagnosis can be viewed as solving classification problems. Among the several DM classifiers for solving classification problems are decision tree classifiers (such as ID3, C4.5, CART), Bayesian classifiers based on statistics, instance-based learners (including case-based reasoning and minimum distance classifier), SVM classifiers, fuzzy decision tree classifiers and NN classifiers [9, 10]. For instance, many credit scoring problems have been solved using NN and SVM classifiers [11-13]. Unfortunately, SVM and BPNN classifiers have some limitations. BPNN based-classifiers have difficulty in demonstrating the predicted results owing to their lack of explanatory capability, limited generalization capacity owing to overfitting, and excessively long too time required to construct the best architecture [13]. With the best settings of the parameters, including the penalty parameter and kernel function parameters, a standard SVM classifier still require to use a trial and error method. Moreover, the multi-class classification cannot outperform binary classification since a standard SVM classifier uses approximation algorithms to reduce the computational complexity at the expense of a degrading classification performance [12].

Wu presented a MIMO cerebellar model articulation controller (CMAC) NN classifier and applied it to support medical-oriented decision making (such as diabetes and cancer), chemical analysis of glass splinters [14] and credit scoring [15] (with an Australian credit approval dataset that was collected from the UCI repository of machine learning databases [16]). The CMAC NN classifier has several advantages, such as very fast learning capability, reasonable generalization capability, flexible network topology, robust noise resistance and robust parameter settings.

In financial decision making such as credit scoring, improving the classification accuracy by even a fraction of a percent translates into significant future savings for the credit industry [17]. Moreover, finding the parameter settings of the MIMO CMAC NN classifier can be considered to be equivalent to solving an unconstrained global optimization (UGO) problem with multiple optima. The parameters quantization resolution of input vector $\mathbf{x}_i$ (Q_i), generalization size (g) and learning rate (β) are the decision variables (x_n, $n = 1, 2, 3$) of the UGO problem.

Therefore, this work develops an advanced DM method that integrates an AIA and a MIMO CMAC NN classifier (AIA-MIMO CMAC NN classifier). The AIA-MIMO CMAC NN classifier is composed of an outer AIA and an inner MIMO CMAC NN classifier. The outer AIA is used to find the optimal parameter settings (such as Q_i, g and β) of the inner CMAC NN classifier, and the inner CMAC NN classifier is applied to solve benchmark classification problems. The performance of the AIA-MIMO CMAC NN classifier is compared with that of published classifiers from the literature [8, 12, 14, 15, 18-21].

2 Related Works

2.1 MIMO CMAC NN Classifier

Figure 1 shows the network topology of the MIMO CMAC NN classifier for solving classification problems.

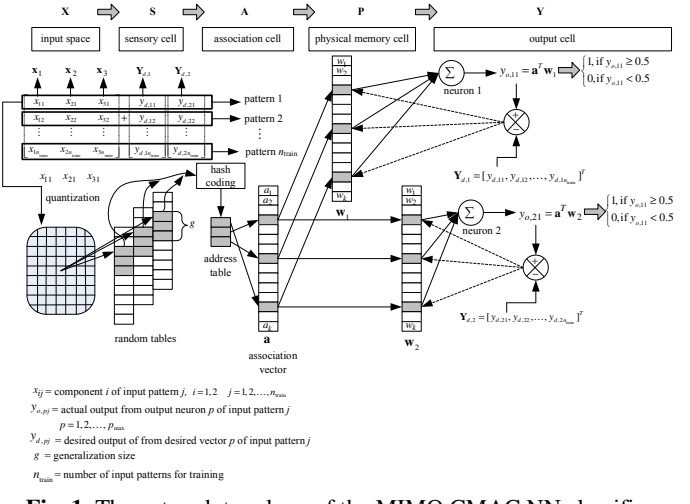


Fig. 1. The network topology of the MIMO CMAC NN classifier

The network topology of the CMAC NN classifier comprises five cells, which are an input space (**X**), a sensory cell (**S**), an association cell (**A**), a physical memory cell (**P**) and an output cell (**Y**). The CMAC NN classifier transforms input values into output values using a series of mapping operations and then updates the weights using a least mean squared (LMS) algorithm. Credit scoring and medical diagnosis can be viewed as a form of binary classification. Therefore, two weight tables **w**s and two output neurons are created. The output neuron uses a binary coding of class. For instance, the code "10" denotes class 1, while code "01" represents class 2. The CMAC NN classifier [14] is described as follows.

Step 0: Initialize paremeter settings

The size of the weight table (k), β , Q_i and g must be set.

Step 1: Quantize the input vector

Each input pattern undergoes a quantization operation to transform continuous or label (such as 1, 2, 3) attributes into discrete indexes, as follows.

$$s_{ij} = \left[\left(\frac{Q_i}{x_i^{\max} - x_i^{\min}} \right) \times \left(x_{ij} - x_i^{\min} \right) \right] - 1, \quad i = 1, 2, \ldots, i_{\max} \;\; j = 1, 2, \ldots, n_{\text{total}} \tag{1}$$

$$s_{ij} = 0, \text{ if } s_{ij} < 0 \tag{2}$$

where s_{ij} = quantization index of component x_i of input pattern j ; x_{ij} = component x_i of input pattern j ; $x_i^{\min}$ = minimum value of input vector $\mathbf{x}_i$; $x_i^{\max}$ = maximum value of input vector $\mathbf{x}_i$; $i_{\max}$ = dimension of input space; n_{total} = total number of input patterns; $[\cdot]$ = rounded number.

Step 2: Create random tables and weight tables

A sensory cell comprises random tables. The size of each random table can be calculated using

$$C_i = Q_i + g - 1, \quad i = 1, 2, \ldots, i_{\max} \tag{3}$$

where C_i = size of the random table of input vector $\mathbf{x}_i$.

Each random table i consists of uniform random numbers generated from the interval [0, k/2].

Step 3: Generate an address table

Many-into-few mapping operations are performed using a bitwise XOR operator-based hash coding, which is described in detail elsewhere [14]. An address table is then created.

Step 4: Calculate the actual output

The actual output values can be calculated using Eq. (4), as follows.

$$y_{o,pj} = \mathbf{a}^T \mathbf{w}_p, \; p = 1, 2 \;\; j = 1, 2, \ldots, n_{\text{train}} \tag{4}$$

where $y_{o,pj}$ = actual output from neuron p of input pattern j; n_{train} = number of input patterns for training;

$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_k \end{bmatrix}$, association vector in cell $\mathbf{A}$; $\mathbf{w}_p = \begin{bmatrix} w_{p1} \\ w_{p2} \\ \vdots \\ w_{pk} \end{bmatrix}$, weight table p in cell $\mathbf{P}$.

Step 5: Implement the LMS algorithm

The CMAC NN classifier modifies the weights by applying a supervised learning algorithm based on the LMS rule. The $\mathbf{w}_p$ ($p = 1, 2$) is independently updated, as follows.

$$w_{pl}(epoch+1) = w_{pl}(epoch) + \frac{\beta(y_{d,pj} - y_{o,pj})}{g}$$
$$l = 1, 2, \ldots, k \quad j = 1, 2, ..., n_{\text{train}} \quad p = 1, 2 \quad epoch = 1, 2, \ldots, epoch_{\max} \tag{5}$$

where β = learning rate; $y_{d,pj}$ = desired output from desired vector p of input pattern j; $w_{pl}(epoch)$ = value of weight l from weight table p in *epoch*; $w_{pl}(epoch+1)$ = value of weight l from weight table p in *epoch*+1; $epoch_{\max}$ = maximum epoch number.

Step 6: Evaluate the training classification error

A neuron output is classified as 1 when its original value is greater than or equal to 0.5, or as 0 when its value is less than 0.5. A training classification error (CE_{train} %) is used as a performance index, as follows.

$$CE_{\text{train}} = \left(\frac{n_{\text{train}} - n_{\text{accuracy}}}{n_{\text{train}}} \right) \times 100 \tag{6}$$

where n_{accuracy} = number of instances for determining accuracy.

According to Eq. (6), classification accuracy ($CA\%$) can be easily obtained by using $1 - CE\ \%$.

Step 7: Measure the test classification error

The input pattern j (j = n_{train}+1, n_{train}+2,..., n_{total}) is used for the test. The actual outputs of recall of the CMAC NN classifier can be calculated from the weight table $\mathbf{w}_p$ ($p = 1, 2$) that was obtained in the training stage (Steps 1–6). The generalization accuracy of the CMAC NN classifier is then evaluated using a test CE_{test} % that is obtained using Eq. (6).

2.2 Artificial Immune Algorithm

Wu [22] presented an AIA, which is a stochastic global optimization approach, and used it to solve constrained global optimization problems. The AIA consists of selection, hypermutation, receptor editing and bone marrow operations. Selection operation is used to reproduce strong antibodies (**Abs**). Hypermutation, receptor editing and bone marrow operations are employed to create diverse **Abs**.

2.2.1 Ab and Ag Representation

In human immune system, an antigen (**Ag**) has multiple epitopes (antigenic determinants), which can be recognized by various **Abs** with paratopes (recognizers), on its surface. In the AIA, a solved problem is represented by an **Ag**. The **Abs** are the candidate solutions (decision variables x_n, $n = 1, 2, ..., N$) of the solved problem. The quality of a candidate solution is measured using from an **Ab-Ag** affinity that is derived from the value of an objective function of the solved problem.

2.2.2 Selection Operation

The idiotypic network selection operation controls the number of antigen-specific **Abs**. This operation is defined according to **Ab-Ag** and **Ab-Ab** recognition information, as follows.

$$p_{r_j} = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{e^{d_{nj}}} \tag{7}$$

$$d_{nj} = \left|\frac{x_n^* - x_{nj}}{x_n^*}\right|,\ j = 1,2,\ldots,rs,\ \ n = 1,2,\ldots,N \tag{8}$$

where p_{r_j} = probability that **Ab** j recognizes $\mathbf{Ab}^*$ (the best solution); x_n^* = the best $\mathbf{Ab}^*$ with the highest **Ab-Ag** affinity; x_{nj} = decision variables x_n of **Ab** j; N = total number of decision variables x_n, rs = repertoire (population) size .

The $\mathbf{Ab}^*$ can be recognized by other **Ab** j in a current **Ab** repertoire. A large value of p_{r_j} implies that **Ab** j can effectively recognize $\mathbf{Ab}^*$. The **Ab** j with p_{r_j} that is equivalent to or larger than the threshold degree p_{rt} is reproduced to generate an intermediate **Ab** repertoire.

2.2.3 Hypermutation Operation

Multi-non-uniform mutation [23] is used as the somatic hypermutation operation, which can be expressed as follows.

$$x_{\text{trial},n} = \begin{cases} x_{\text{current},n} + (x_n^u - x_{\text{current},n})A(g),\ if\ U(0,1) < 0.5 \\ x_{\text{current},n} - (x_{\text{current},n} - x_n^l)A(g),\ if\ U(0,1) \geq 0.5 \end{cases} \tag{9}$$

where

$A(g) = \left\{ U_1(0,\ 1)\left(1 - \frac{g}{g_{\max}}\right)\right\}^2$ = perturbation factor; $x_{\text{current},n}$ = current value of decision variable x_n; $x_{\text{trial},n}$ = trial value of decision variable x_n; x_n^l = the lower boundary of decision variables x_n, x_n^u = the upper boundary of decision variables x_n, g = current generation; $g_{\max}$ = maximum generation number; $U(0,1)$ and $U_1(0,1)$ = uniform random number in the interval [0, 1].

This operation has two tasks, i.e. a uniform search and local fine-tuning.

2.2.4 Receptor Editing Operation

The standard Cauchy distribution $C(0,1)$, in which the local parameter is zero and the scale parameter is one, is used to develop a receptor editing operation. Receptor editing is performed using Cauchy random variables that are generated from $C(0,1)$ owing to their ability to provide a large jump in **Ab-Ag** affinity landscape to increase the probability of escaping from the local **Ab-Ag** affinity landscape. Cauchy receptor editing can be expressed by.

$$\mathbf{x}_{\text{trial}} = \mathbf{x}_{\text{current}} + U_2(0,\ 1)^2 \times \boldsymbol{\sigma} \tag{10}$$

where $\boldsymbol{\sigma} = [\sigma_1, \sigma_2, \ldots, \sigma_N]^T$, vector of Cauchy random variables; $U_2(0,1)$ = uniform random number in the interval [0, 1].

This operation functions in local fine-tuning and large perturbation.

2.2.5 Bone Marrow Operation

The paratope of an **Ab** can be generated by recombining gene segments $\mathbf{V_H}$ $\mathbf{D_H}$ $\mathbf{J_H}$ and $\mathbf{V_L}$ $\mathbf{J_L}$ [24]. Hence, diverse **Abs** based on this metaphor are synthesized using a bone marrow operation. The detailed implementation of the bone marrow operation can be found in the literature [22].

3 Method

Figure 2 shows the pseudo-code of the proposed AIA-MIMO CMAC NN classifier, in which the outer AIA is used to optimize the parameter settings of the inner MIMO CMAC NN classifier, which is employed to solve benchmark classification problems.

Outer AIA:

Step 1: Initialize the parameter settings

Several parameters must be predetermined. These include rs and the threshold for **Ab-Ab** recognition p_{rt}. An available **Ab** repertoire (population) is randomly generated using rs from [x_n^l, x_n^u], such as the lower and upper boundaries of Q_i [Q_i^l, Q_i^u] = [30, 50], the lower and upper boundaries of g [g_i^l, g_i^u] = [50, 120] and the lower and upper boundaries of β [β_i^l, β_i^u] = [0.05, 0.1]. Parameters Q_i and g take integer values, and parameter β is a real value.

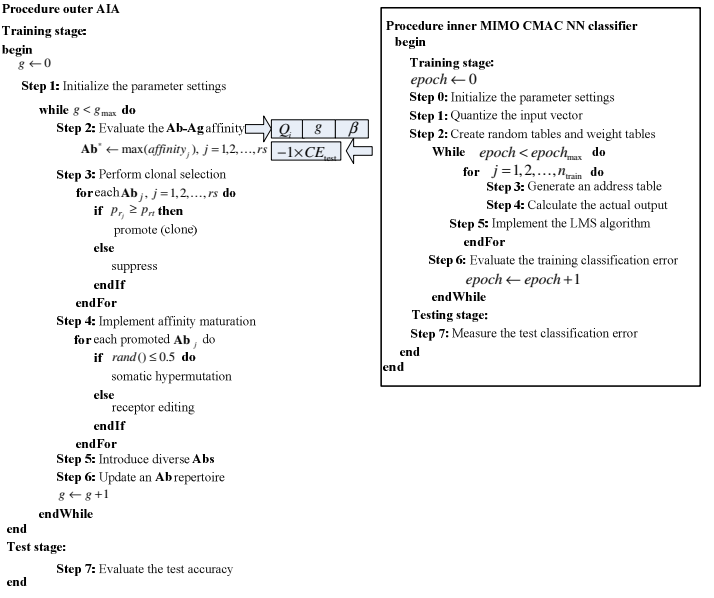


Fig. 2. The pseudo-code of the proposed AIA-MIMO CMAC NN classifier

Step 2: Evaluate the Ab-Ag affinity

Inner MIMO CMAC NN classifier:

The outer AIA offers parameter settings Q_i, g and β for the inner MIMO CMAC NN classifier. Subsequently, inner Steps 0–7 of the CMAC NN classifier, described in section 2.1, are implemented. The inner CMAC NN classifier returns the test CE_{test} % to outer AIA.

end

Consistent with the **Ab-Ag** affinity metaphor, Eq. (11) is used to determine an **Ab-Ag** affinity, as follows.

$$\max(affinity_j) = -1 \times CE_{\text{test}} \% \quad j = 1,2,...,rs \tag{11}$$

Following the evaluation of the **Ab-Ag** affinities of **Abs** in the current **Ab** repertoire, the **Ab** with the highest **Ab-Ag** affinity ($\mathbf{Ab}^*$) is chosen to undergo clonal selection in outer Step 3.

Step 3: Perform clonal selection

The idiotypic network selection operation, described in section 2.2.2, is used to select strong **Abs**.

Step 4: Implement Ab-Ag affinity maturation

The intermediate **Ab** repertoire that is created in outer Step 3 is divided into two subsets. A random number is generated from a uniform distribution for each **Ab** in the intermediate **Ab** repertoire. These **Abs** undergo somatic hypermutation when the random number equals or is less than 0.5. They suffer receptor editing when the

random number exceeds 0.5. Since parameters Q_i and g have integer values, somatic hypermutation and receptor editing operations defined by Eqs. (9) and (10), are revised to Eqs. (12) and (13), respectively.

$$x_{\text{trial},n} = \begin{cases} \left[x_{\text{current},n} + (x_n^u - x_{\text{current},n}) A(g) \right], \text{if } U_3(0,1) < 0.5 \\ \left[x_{\text{current},n} - (x_{\text{current},n} - x_n^l) A(g) \right], \text{if } U_3(0,1) \geq 0.5 \end{cases}, n = 1,2$$
$$x_{\text{trial},n} = \begin{cases} x_{\text{current},n} + (x_n^u - x_{\text{current},n}) A(g), \text{if } U_4(0,1) < 0.5 \\ x_{\text{current},n} - (x_{\text{current},n} - x_n^l) A(g), \text{if } U_4(0,1) \geq 0.5 \end{cases}, n = 3 \quad (12)$$

$$\mathbf{x}_{\text{trial}} = \left[\mathbf{x}_{\text{current}} + U_5(0,\ 1)^2 \right] \times \boldsymbol{\sigma}, n = 1,2$$
$$\mathbf{x}_{\text{trial}} = \mathbf{x}_{\text{current}} + U_5(0,\ 1)^2 \times \boldsymbol{\sigma}, n = 3 \quad (13)$$

where $[\cdot]$ = rounded number, $U_3(0,\ 1)$, $U_4(0,\ 1)$ and $U_5(0,\ 1)$ = uniform random number in the interval [0, 1].

Step 5: Introduce diverse Abs

The bone marrow operation that is introduced in section 2.2.5 is used to create diverse **Abs** to recruit the **Abs** that are suppressed in outer Step 3.

Step 6: Update an Ab repertoire

A new **Ab** repertoire is generated from outer Steps 3−5. The **Ab-Ag** affinities of the **Abs** in the generated **Ab** repertoire are evaluated. This work presents a strategy for updating the **Ab** repertoire. A situation in which the **Ab-Ag** affinity of **Ab** *j* in the new **Ab** repertoire exceeds that in the current **Ab** repertoire implies that a strong **Ab** in the new **Ab** repertoire replaces the weak **Ab** in the current **Ab** repertoire. Additionally, if the **Ab-Ag** affinity of **Ab** *j* in the new **Ab** repertoire is equal to or less than that in the current **Ab** repertoire, then the **Ab** *j* in the current **Ab** repertoire survives. In addition to maintaining the strong **Abs**, this strategy eliminates non-functional **Abs**.

Repeat outer Steps 2−6 until the termination criterion $g_{\max}$ is satisfied.

4 Results

The proposed AIA-MIMO CMAC NN classifier was coded in MATLAB programming language, and executed on a Pentium D 3.0 (GHz) personal computer. Australian credit approval credit [16] and diabetes [18] datasets are used to evaluate the performance of the proposed AIA-MIMO CMAC NN classifier. For Australian credit approval credit dataset, the ten-fold cross-validation is performed to generate random folds of a dataset. In ten-fold cross-validation, a dataset is initially partitioned into ten equally or almost equally sized folds. Ten iterations of training and validation are then implemented. Each random fold is independently used as a validation fold, those remaining nine folds are used as training set. Each test dataset comprises of a validation fold and nine training folds. The AIA-MIMO CMAC NN classifier was run independently ten times by using the ten-fold cross-validation for each combination of

settings (p_{rt} and rs) and summarizes mean CE_{train} , mean CE_{test} , mean computational CPU time (MCCT), mean Q_i , mean g and mean β . For the diabetes dataset, three permutations of the patterns that are available in the PROBEN1 are used [18]. The AIA-MIMO CMAC NN classifier was run independently 30 times by using the diabetes1, diabetes2 and diabetes3 for each combination of settings (p_{rt} and rs).

In each run, parameter p_{rt} = 0.9, parameter rs = {2, 4}, termination criterion $epoch_{\max}$= 30, $g_{\max}$= 3 and k =10000 were used.

4.1 Australian Credit Approval Credit Dataset

The Australian credit approval credit dataset that was collected from the UCI repository of machine learning databases [16] is used. Related to credit card applications, the data in the Australian credit approval dataset have a good mixture of attributes, including continuous and nominal values. Data confidentiality is ensured by transforming the attribute names into meaningless symbolic data. This dataset contains 15 attributes and one class label, + or −, as well as 690 instances, of which 307 are positive (credit approved) and 383 are negative (credit denied). Additionally, a few values are missing. The Statlog project [25, 26] modified this dataset by replacing missing values; the labels of the attributes were changed for the convenience of statistical algorithms. In the modified dataset, each instance has six nominal and eight numerical attributes, as well as one class attribute − accepted or rejected.

Table 1 lists the experimental results obtained by applying the proposed AIA-MIMO CMAC NN classifier to Australian credit approval credit dataset.

Table 1. Experimental results of the proposed AIA-MIMO CMAC NN classifier for Australian credit approval credit dataset

	p_{rt} = 0.9											
	rs = 2						rs = 4					
ten folds	mean CE_{train}	mean CE_{test}	MCCT (sec.)	mean Q_i	mean g	mean β	mean CE_{train}	mean CE_{test}	MCCT (sec.)	mean Q_i	mean g	mean β
dataset 1	5.19	15.36	254.69	38	87	0.07	5.49	15.07	466.83	40	102	0.07
dataset 2	5.22	11.74	261.64	42	94	0.08	5.46	11.88	449.18	40	96	0.08
dataset 3	5.75	8.70	269.35	38	99	0.08	5.65	8.70	453.08	38	97	0.08
dataset 4	4.98	20.58	268.13	39	98	0.08	6.36	18.84	462.51	37	104	0.06
dataset 5	5.31	12.46	269.47	40	98	0.08	4.20	11.01	445.22	43	94	0.09
dataset 6	5.68	12.46	263.00	41	96	0.08	5.15	11.59	436.15	41	90	0.08
dataset 7	6.28	6.09	273.15	39	103	0.07	6.01	5.94	457.23	42	100	0.07
dataset 8	5.25	15.22	260.90	37	95	0.07	5.06	15.65	444.09	37	94	0.08
dataset 9	5.85	10.58	269.16	41	101	0.07	6.92	10.14	474.37	37	106	0.07
dataset 10	5.85	12.32	258.00	38	90	0.08	6.10	12.17	432.94	37	89	0.08
mean	5.53	12.55	264.75	39	96	0.08	5.64	12.10	452.16	39	97	0.08
mean *CA%*	94.47	87.45					94.36	**87.90**				

It indicates that increasing rs increased the MCCT. Moreover, the mean test CA s% obtained using the $rs = \{2, 4\}$ are identical. The best mean test $CA\%$ was 87.90% and the optimal parameter settings of the MIMO CMAC NN classifier, found using the outer AIA were $Q_i = 39$, $g = 97$ and $\beta = 0.08$.

Table 2 compares the experimental results obtained by using the proposed AIA-MIMO CMAC NN classifier with those obtained using published classifiers. It shows that the $CA\%$ of the proposed AIA-MIMO CMAC NN classifier is superior to those of SVM, BPNN, C4.5, SVM-based classifiers and individual MIMO CMAC NN classifier.

Table 2. Comparsion of experimental results of the proposed AIA-MIMO CMAC NN classifier with published classifiers

Methods	$CA\%$
SVM [12]	86.07
BPNN [8]	86.83
GP [8]	87.00
C4.5 [8]	85.90
SVM+Grid search [8]	85.51
SVM+Grid search + F-score [8]	84.20
SVM+GA [8]	86.90
MIMO CMAC NN classifier [15]	86.09
this work (AIA- MIMO CMAC NN classifier)	**87.90**

4.2 Diabetes Dataset

The diabetes dataset is related to diagnoses of diabetes in the Pima Indians population. It includes eight real-valued features (number of times pregnant, plasma glucose concentration, diastolic blood pressure, triceps skin fold thickness, 2h serum insulin, body mass index, diabetes pedigree function and age), two classes (diabetes positive or diabetes negative) and 768 instances. Each dataset was divided into two parts; 75% of the data instances were used in training (with 50% as the training set and 25% as the validation set, defined in the original PROBEN1 settings), and the remaining 25% were employed for testing. However, in this work, the first 50% of a dataset was used for training and the last 25% of the dataset was used in testing.

Table 3 presents the experimental results obtained by applying the proposed AIA-MIMO CMAC NN classifier to the diabetes dataset. It indicates that increasing rs increased the MCCT. Moreover, the mean test CE_{test} s% obtained using the $rs = \{2, 4\}$ are identical for diabetes1, diabetes2 and diabetes3. For dataset diabetes1, the best mean $CE_{\text{test}}\%$ was 24.70% and the optimal parameter settings of the MIMO CMAC NN classifier, obtained using the outer AIA were $Q_i = 42$, $g = 94$ and $\beta = 0.08$. For dataset diabetes2, the best mean $CE_{\text{test}}\%$ was 26.55% and the optimal parameter settings of the MIMO CMAC NN classifier, obtained using the outer AIA were $Q_i = 40$, $g = 93$ and $\beta = 0.08$. For dataset diabetes3, the best mean $CE_{\text{test}}\%$ was 22.55% and the optimal parameter settings of the MIMO CMAC NN classifier, obtained using the outer AIA were $Q_i = 36$, $g = 94$ and $\beta = 0.08$.

Table 3. Experimental results of the proposed AIA-MIMO CMAC NN classifier for diabetes dataset

	$p_{rt} = 0.9$											
	$rs = 2$						$rs = 4$					
	mean CE_{train}	mean CE_{test}	MCCT (sec.)	mean Q_i	mean g	mean β	mean CE_{train}	mean CE_{test}	MCCT (sec.)	mean Q_i	mean g	mean β
diabetes1	10.71	25.14	110.65	40	91	0.08	10.48	**24.70**	194.20	42	94	0.08
diabetes2	10.33	26.93	111.65	38	93	0.08	10.11	**26.55**	190.85	40	93	0.08
diabetes3	11.05	22.86	113.28	37	96	0.07	10.68	**22.55**	191.02	36	94	0.08
mean	10.70	24.98	111.86	38	94	0.08	10.42	24.60	192.02	39	94	0.08

Table 4 compares the experimental results obtained using the proposed AIA-MIMO CMAC NN classifier and published classifiers. It indicates that the mean $CE_{\text{test}}\%$ of the proposed AIA-MIMO CMAC NN classifier is better than those of Zhu and Guan [21] for diabetes1. Furthermore, the mean $CE_{\text{test}}\%$ of the proposed AIA-MIMO CMAC NN classifier is better than those obtained by de Falco et al. [19] with Brameier and Banzhaf [20] for diabetes2. Moreover, the mean $CE_{\text{test}}\%$ of the proposed AIA-MIMO CMAC NN classifier is superior to those obtained by de Falco et al. [19] and Zhu and Guan [21] for diabetes3.

Table 4. Comparsion of experimental results of the proposed AIA-MIMO CMAC NN classifier with published classifiers

	de Falco *et al.* [19]	Prechelt [18]	Brameier and Banzhaf [20]	Zhu and Guan [21]	Wu [14]	this work
datasets	mean CE_{test}	mean CE_{test}	mean CE_{test}	mean CE_{test}	mean CE_{test}	mean CE_{test}
diabetes1	24.84	24.10	23.96	25.86	25.57	24.70
diabetes2	30.36	26.42	27.85	26.89	27.53	26.55
diabetes3	26.09	22.59	23.09	24.69	23.33	22.55

According to Tables 1–4, the outer AIA can obtain the optimal settings of the parameters for the inner MIMO CMAC NN classifier, which can efficiently be applied to these test datasets. Although the AIA-MIMO CMAC NN classifier increases the computational complexity, the proposed classifier has a greater $CA\%$ and reduces parameterization of the MIMO CMAC NN classifier.

5 Conclusions

This work developed an advanced DM approach that hybridizes an AIA with a MIMO CMAC NN classifier and applied the proposed classifier to two benchmark datasets – Australian credit approval and diabetes datasets. Experimental results indicate that the AIA-MIMO CMAC NN classifier outperforms some published

classifiers for test datasets. Therefore, the AIA-MIMO CMAC NN classifier can be regarded as an efficient DM scheme for business and medical decision making. Further work will apply the AIA-MIMO CMAC NN classifier to a German credit dataset that was collected from the UCI repository of machine learning databases, and cancer with glass datasets taken from PROBEN1.

Acknowledgements. The author would like to thank the National Science Council of the Republic of China, Taiwan for financially supporting this research under Contract No. NSC 100-2622-E-262-006-CC3.

References

1. Mitra, S., Pal, S.K., Mitra, P.: Data Mining in Soft Computing Framework: A Survey. IEEE Transactions on Neural Networks 13(1), 3–14 (2002)
2. Zadeh, L.A.: Fuzzy Logic, Neural Networks, and Soft Computing. Communications of the ACM 37(3), 77–84 (1994)
3. Konar, A.: Computational Intelligence-Principles, Techniques and Applications. Springer, New York (2005)
4. Poorzahedy, H., Rouhani, O.M.: Hybrid Meta-Heuristic Algorithms for Solving Network Design Problem. European Journal of Operational Research 182(2), 578–596 (2007)
5. Valdez, F., Melin, P., Castillo, O.: An Improved Evolutionary Method with Fuzzy Logic for Combining Particle Swarm Optimization and Genetic Algorithms. Applied Soft Computing 11(2), 2625–2632 (2011)
6. Abd-El-Wahed, W.F., Mousa, A.A., El-Shorbagy, M.A.: Integrating Particle Swarm Optimization with Genetic Algorithms for Solving Nonlinear Optimization Problems. Journal of Computational and Applied Mathematics 235(5), 1446–1453 (2011)
7. Kuo, R.J., Han, Y.S.: A Hybrid of Genetic Algorithm and Particle Swarm Optimization for Solving Bi-Level Linear Programming Problem - A Case Study on Supply Chain Model. Applied Mathematical Modelling 35(8), 3905–3917 (2011)
8. Huang, C.L., Chen, M.C., Wang, C.J.: Credit Scoring with a Data Mining Approach Based on Support Vector Machines. Expert Systems with Applications 33(4), 847–856 (2007)
9. Mitra, S., Acharya, T.: Data Mining—Multimedia, Soft Computing, and Bioinformatics. Wliey and Sons, New Jersey (2003)
10. Olafsson, S., Li, X., Wu, S.: Operations Research and Data Mining. European Journal of Operational Research 187(3), 1429–1448 (2008)
11. Tsai, C.F., Wu, J.W.: Using Neural Network Ensembles for Bankruptcy Prediction and Credit Scoring. Expert Systems with Applications 34(4), 2639–2649 (2008)
12. Wang, S.J., Mathew, A., Chen, Y., Xi, L.F., Ma, L., Lee, J.: Empirical Analysis of Support Vector Machine Ensemble Classifiers. Expert Systems with Applications 36(3), Part 2, 6466–6476 (2009)
13. Min, J.H., Lee, Y.C.: Bankruptcy Prediction Using Support Vector Machine with Optimal Choice of Kernel Function Parameters. Expert Systems with Applications 28(4), 603–614 (2005)
14. Wu, J.Y.: MIMO CMAC Neural Network Classifier for Solving Classification Problems. Applied Soft Computing 11(2), 2326–2333 (2011)
15. Wu, J.Y., Tseng, Y.H.: Evaluating Credit Risk via A MIMO CMAC Neural Network Classifier-Based Data Mining Approach. In: 2011 The 3rd International Conference on Machine Learning and Computing, pp. 147–151 (2011)

16. Asuncion, A., Newman, D.J.: UCI Machine Learning Repository. University of California, Department of Information and Computer Science (2007), http://www.ics.uci.edu/~mlearn/mlrepository.html
17. West, D., Dellana, S., Qian, J.: Neural Network Ensemble Strategies for Financial Decision Applications. Computers & Operations Research 32(10), 2543–2559 (2005)
18. Prechelt, L.: Proben1 - A Set of Neural Network Benchmark Problems and Benchmarking Rules. Universität Karlsruhe (1994)
19. de Falco, I., Cioppa, A.D., Tarantino, E.: Discovering Interesting Classification Rules with Genetic Programming. Applied Soft Computing 1(4), 257–269 (2002)
20. Brameier, M., Banzhaf, W.: A Comparison of Linear Genetic Programming and Neural Networks in Medical Data Mining. IEEE Transactions on Evolutionary Computation 5(1), 17–26 (2001)
21. Zhu, F., Guan, S.: Feature Selection for Modular GA-Based Classification. Applied Soft Computing 4(4), 381–393 (2004)
22. Wu, J.Y.: Solving Constrained Global Optimization via Artificial Immune System. International Journal on Artificial Intelligence Tools 20(1), 1–27 (2011)
23. Houck, C.R., Joines, J.A., Kay, M.G.: A Genetic Algorithm for Function Optimization: A Matlab Implementation. North Carolina State Univ., Raleigh (1995)
24. de Castro, L.N., Von Zuben, F.J.: Artificial Immune Systems: Part I- Basic Theory and Applications. FEEC/Univ. Campinas, Campinas, Brazil (1999), ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/tr_dca/trdca0199.pdf
25. Michie, D., Spiegelhalter, D.J., Taylor, C.C.: Machine Learning, Neural, and Statistical Classification. Ellis Horwood, London (1994)
26. Sakprasat, S., Sinclair, M.C.: Classification Rule Mining for Automatic Credit Approval Using Genetic Programming. In: IEEE Congress on Evolutionary Computation, pp. 548–555 (2007)

Improving Suffix Tree Clustering with New Ranking and Similarity Measures

Phiradit Worawitphinyo, Xiaoying Gao, and Shahida Jabeen

School of Engineering and Computer Science
Victoria University of Wellington, P.O. Box 600, Wellington, New Zealand
holypromise@hotmail.com, {xgao,shahidarao}@ecs.vuw.ac.nz

Abstract. Retrieving relevant information from web, containing enormous amount of data, is a highly complicated research area. A landmark research that contributes to this area is web clustering which efficiently organizes a large amount of web documents into a small number of meaningful and coherent groups[1, 2]. Various techniques aim at accurately categorizing the web pages into clusters automatically. Suffix Tree Clustering (STC) is a phrase-based, state-of-art algorithm for web clustering that automatically groups semantically related documents based on shared phrases. Research has shown that it has outperformed other clustering algorithms such as K-means and Buckshot due to its efficient utilization of phrases to identify the clusters. Using STC as the baseline, we introduce a new method for ranking base clusters and new similarity measures for comparing clusters. Our STHAC technique combines the Heirarchical Agglomerative clustering method with phrase based Suffix Tree clustering to improve the cluster merging process. Experimental results have shown that STHAC outperforms the original STC as well as ESTC(our precious extended version of STC) with 16% increase in F-measure. This increase in F-measure of STHAC is achieved due to its better filtering of low score clusters, better similarity measures and efficient cluster merging algorithms.

Keywords: Web Page Clustering, Suffix Tree, Vector Space Model, Single Linkage, Agglomerative Hierarchical Clustering, Similarity measure.

1 Introduction

The gigentic increase of the World Wide Web has made information access and retrieval more challanging than before. In the presence of huge volume of data,the issue of faster access to the most relevant information is becoming more and more critical. A possible solution to this problem focuses on performing a post retrieval categorization of the search results into coherent groups.Web Page Clustering is the process of organizing a large number of unstructured text into a small number of groups called clusters, having maximum intra cluster similarity and minimum inter cluster similarity thereby making the searching and browsing faster and more efficient. Diverse approaches such as Hierarchical Methods,Partitioning

J. Tang et al. (Eds.): ADMA 2011, Part II, LNAI 7121, pp. 55–68, 2011.

Methods,Density-based methods,Geometric embedding and probabilistic methods are used for web page clustering.A state-of-art clustering method is Suffix Tree Clustering which is a post retrieval linear time algorithm that categorizes huge vloume of data into coherent clusters based on shared phrases [3] . Suffix Tree Clustering gained the attention of researchers very quickly due to its efficient utilization of phrases to identify clusters and allowing clusters to overlap[4–6].

While the algorithm itself performed better than most of other clustering algorithms, various steps in the algorithm left room for further improvements. Cluster overlapping is not handled efficiently in STC creating large sized clusters with noticably poor quality. STC uses a similarity measure to identify similar clusters based on overlaps of documents sets[7]. After that, cluster merging is performed based on this score only when they have enough overlapping. Despite its fastness and simplicity, this merging scheme is not an optimal choice. This paper focuses on certain imperative issues regarding cluster merging. Another problem in STC is that the weight of the phrase is calculated from the effective length of phrase and the cluster score panelizes single words even when they are of importance thus losing much of important information. Hence the criteria of assigning weights to the phrases is not efficient. STC lacks efficient similarity measures for calculating the inter cluster as well as intra cluster similarity. In this paper we introduce a new method for ranking base clusters and new similarity measures for comparing clusters. STHAC technique combines Heirarchical Agglomerative clustering with phrase based Suffix Tree clustering to improve the cluster merging. The primary goal of this project is to improve web page clustering performance by exploring different clustering techniques and combining them to create a better web pag clustering algorithm. The research questions, that we intended to answer in this paper, are as follows:

1) How “cluster score” can affect the cluster quality and which score measure performs better to select higher quality clusters? “Cluster score” is a way to determine the importance of a word or a phrase, representing each cluster, to the overall documents collection. Generally, we only want to keep those clusters that contain more meaningful words and phrases. With such high quality clusters, we can certainley improve the overall clustering performance.

2) Which Similarity Measures are suitable for merging similar clusters when following the Agglomerative Hierarchical Clustering? Two clusters can be merged together to form a better cluster when they are considered relevant enough for merging. To do this, various statistical methods can be used to calculate the clusters “Similarity Score”. With a better similarity measure used, we can ensure that only relevant clusters, that contain documents with similar contents, can be merged together to form larger cluster and in turn this increases the chance of returning only the set of documents that are related to the user’s query[8].

3) Why adopting different “Clusters Merging” algorithms affect the overall clustering process and what are the driving factors in the cluster merging algorithms that guide the clustering process to improvement? There are many different merging algorithms that can be used to accomplish this, such as Heierarchical Agglomerative merge, which give priority to the most compatible pair and merge

them together and a single linkage merging, which merges a cluster to its nearest neighbour if certain conditions are met[9].

This paper is organized as follows. In Section 2, related work on Suffix Tree based clustering is dicussed along with the efficiency analysis of each algorithm. In Section 3, Our proposed solution(STHAC) to clustering is introduced and various states of the STHAC are illustrated in detail. Section 4 includes the discussion of a number of experiments on various datasets for two keywords 'Salsa' and 'Jaguar', and an analysis of the results is also depicted. Both the effectiveness and the efficiency of the STHAC are validated by the experimental results in comparison with a number of other most common clustering techniques including the state-of-art STC and ESTC algorithms.

2 Related Work

There are three factors that effectively influence the cluster's quality as well as clustering performance. These are namely data representation,similarity measure and the clustering technique that merges the clusters using the similarity measure[10–14]. Vector Space Model(VSM) is a data representation scheme used in most of the Web clustering techniques. In this model a document is represented as a multi dimentional vector of words where each word represents one dimension. Each word is assigned a weight based on TFIDF scheme. The similarity between two documents is usually computed using the various similarity measures such as cosine similarity, Pearson correlation coefficient or Jaccard Coefficient etc[15–17]. Clustering methods based on this scheme consider only the single word representation of documents hence ignoring most valuable word proximity information[5, 11, 18]. Moreover, clustering techniques based on vector space model do not support incremental processing[19]. Capenato et. al. stated in their survay of web clustering engines that incremental processing improves the efficiency when applied in the clustering methods [20]. The most related work that takes into account the information about proximity of words and phrase based analysis in an incremental way is Suffix Tree Clustering(STC)[21] by Zamir et. al.

STC is an incremental linear time (O(n)) algorithm[6, 18] that classifies the documents sharing phrases or the suffix of a phrase into one cluster. The group of these documents is identified by constructing a Suffix tree which represents various phrases and their all possible suffixes.A Suffix Tree is a data structure widely used in diverse applications[15, 22, 23]. A suffix Tree is a Compact trie data Structure representing all the suffixes of a text phrase[3, 6, 23], where a phrase means a sequence of words not the grammatical structure of a sentence[10]. The major advantage of STC is that it makes an efficient use of phrases rather then considering just words and it allows clusters to overlap which is quite practical in the real world as a document can contain information regarding multiple topics and can belong to more then one clusters[4].

Various efforts are made to improve the efficiency of Suffix Tree based clustering by identifying the problems in the original STC. One effort is done by

Jongkol et. al. who proposed the Semantic Suffix Tree(SSTC)[19], by combining the semantic similarity, in the original suffix Tree. They used the Wordnet to derive the semantic similarity and used it along with the string matching in the costruction of suffix Tree.Their purpose was to identify the phrases having same context but consisting of different words, for example, the context of two phrases "doctor and nurse" and "physician and nanny" is similar but Suffix Tree considers them differently. Another extension of STC is the Semantic Hierarchical Online Custering(SHOC) algorithm [24], which used Suffix Arrays to extract the frequent phrases and singular value decomposition(SVD) technique to identify the cluster contents. Lingo algorithm[25] is further extension to the SHOC algorithm that combines the common phrases with the latent semantic analysis to categorize the search results into coherent clusters. A newer version of Lingo is the Semantic Lingo Algorithm[26] that applied synonym in the result snippets to increse the cluster quality. To solve the problem of large clusters with poor quality in STC, Chim et. al. proposed NTSC algorithm. NTSC combined the vector space model with the Suffix Tree to calculate similarity between pair wise documents to increase the quality of large size clusters[12, 14]. Jianhua wang et. al. proposed a cluster merging algorithm that combined the cosine similarity with the non overlapping parts of the clusters to take into consideration the similarity between the non overlapping parts of the clusters[23]. Daniel Crabtree et. al. improved upon the STC by efficiently identifying the relation between the user query and the clusters. They used the Cluster Description Similarity alongwith the cluster overlapping in a new cluster splitting method to overcome the problem of cluster chaining [27]. To improve the clustering performance in STC, Daniel et. al. devised Extended Suffix Tree Clustering (ESTC)[4] to select a small number of clusters from a larger set of Clusters returned by the STC .They introduced a new cluster ranking function based on heuristics and a new cluster selection algorithm. Their work is based on a new cluster scoring method to select only the top ranked most relevent clusters. Archana Kale improved the cluster labeling by assigning the words of phrases (on average 3 words for each cluster) having highest frequencies to the cluster as labels[28]. Jiangning et. al. extended the STC for the Chinese web page clustering by introducing the Chinese synonyms in the suffix tree to improve cluster quality[29].

Hierarchical Clustering is prefered over non-hierarchical Clustering because in non-hierarchical clustering a central point, also called centroid, needed to be choosen randomly and the distance from that point is calculated to group documents (with less distance) in one cluster[30, 31]. Finding this central point poses a big challenge. That is why non hierarchical methods are not very popular. A comparative work on both the methods is done by [32]. Florian Beil et. al. introduced two clustering algorithms FTC(non-hierarchical) and HFTC(hierarchical) in [32], based on the concept of frequent Term Set and analysed their behaviour. They used the association rule mining to identify the frequent terms in documents to group them into clusters.

Agglomertive Hierarchical Clustering(AHC) is a widely used buttom up clustering algorithm. Many researchers have efficiently used this method in a

variety of domains ranging from Psychology[33] and Linguistics[34] to information retrieval[35] and data and Web Clustering[36]. There is a group of methods that is used for Agglomerative hierarchical Clustering. Most common of these methods are Single Linkage, Complete Linkage, Group Average Linkage and Wards Method[37]. All of these methods differ in the approach of similarity calculation that guides the selection of the most similar pair of clusters.

Recent research has developed new methods for estimating the semantic distance between two terms or words. Rudi L. Cilibrasi et. al. introduced a new similarity measure, called Normalized Google Distance(NGD) [36], to effectively capture the semantic similarity between words and phrases based on information distance and Kolmogorove Complexity. Later on, Alberto J Evangelista et. al. reviewed the work of Rudi L. Cilibrasi to improve their distance function through elimination of random data[38]. We adopted this method to estimate the similarity between two clusters rather then just two words.

3 STHAC: Suffix Tree Hierarchical Agglomerative Clustering

STHAC algorithm has used Suffix tree clustering algorithm (STC)[3] as the baseline system. The research intention behind STHAC is to improve the techniques used during various stages of suffix tree clustering so that it may result in a new phrase-based clustering algorithm that provides better results than that of the original suffix tree algorithm. The phases involved in the original suffix tree clustering algorithm can be broken down into various steps like Cleaning documents, identifying base clusters using suffix tree data structure, assigning each cluster a score based on its importance and merging similar clusters together.

The first two steps are very straight forward and are the key steps that help STC to outperform various algorithms. The third and final step of the STC algorithm however, can still be greatly improved with better techniques. For example,the cluster scoring function that STC used is too simple and it does not take into consideration the overall size of the input data collection. The merging method of STC is also not very efficient and leaves space for improvement. The modification and improvements in these steps can help a lot to get better results. Comparison of STC and STHAC is shown in figure 2.

For better selection of clusters we used the standard TFIDF[39] score and our results proved that using TFIDF was more effective then the STC's Cluster Scoring method because it takes the number of appearance of the phrase and the size of the entire data set into consideration.

In order to calculate the similarity between the base clusters, in place of STC's overlapping formula we used Normalized Google Distance [36]. The overlapping formula only considers the overlapping between the documents of each cluster without considering the importance of the phrases representing each cluster to the overall data set.

Hierarchical Agglomerative Clustering (HAC)[9] is used as an algorithm for merging base clusters in place of STC's original merging algorithm. STC used a

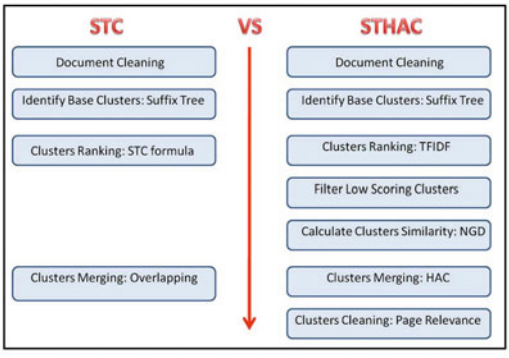


Fig. 1. STC vs. STHAC

top down merging approach. It does not take into the fact that there might be clusters that are more similar to one another and should be merged first.

One new cluster filtering step namely Cluster Cleaning, is added into the new algorithm to further improve the performance of the new clustering algorithm.The rest of this section describes STHAC, a Suffix Tree Hierarchical Agglomerative Clustering Algorithm with four key steps highlighting our key innovations compared to the original STC.

3.1 Step 1 - Document Cleaning

Document cleaning is a preliminary phase of text processing applications[40]. In this phase, a search and replace method is used to remove unwanted words from the input documents. These words usually introduce noice in the documents but are of no significance. Web page documents obtained directly from the world wide web often contain HTML and XML tags in their content. During document cleaning step these tags are also removed.

3.2 Step 2 - Base Cluster Identification

After the document cleaning,we used suffix tree data structure to help identify base clusters. A base cluster is a set of all the documents sharing a single phrase and is represented by that phrase[27]. Suffix tree data structure maps the phrase overlapping of each document with other documents and includes them into various clusters. This approach is quite efficient and different from other techniques. This is one of the reasons of better performance of STC as compared to other clustering algorithms.

Each base cluster that has been identified using suffix tree, consists of a set of documents sharing a common phrase. The importance of a phrase that represents a base cluster to the overall document collection can be an important parameter for calculating the base cluster's scores.

3.3 Step 3 - Clusters Ranking and Filtering

At the end of first step, we are left with a large amount of base clusters. This cluster collection includes all clusters that contain common words that were left out during the document cleaning phase. Because we are interested in keeping only high quality clusters before the merging process actually begins, a way of filtering out bad quality clusters is required. To solve this problem, we decided to use standard 'term frequency-inverse document frequency' (TFIDF)[39] to filter out low scoring clusters from the suffix tree. This helps a lot in reducing computation time needed to calculate similarity scores between clusters. The TFIDF score for a cluster depends on the high frequency of a phrase in the cluster's documents set and the low frequency in the total number of documents the phrase appears in the collection. Because of this nature, TFIDF is very helpful in filtering out general terms that are common in most documents, leaving only unique terms from each document. The formula used to calculate each word's frequency score is given by:

$$tf_{(i,j)} = \frac{n_{(i,j)}}{\sum_k n_{(k,j)}} \tag{1}$$

Where the numerator is the total number of appearance of a phrase in the set of documents its base cluster contains and the denominator is the total number of words in the documents. The inverse documents frequency is what we use to measure a term's general importance to the entire documents collection and is given by:

$$idf_i = \log \frac{|D|}{|\{d : t_i \in d\}|} \tag{2}$$

Where D is the total number of documents in the collection and the denominator is the total number of documents that the phrase appears in. The TFIDF score for a phrase is then given by:

$$(tfidf)_{i,j} = tf_{i,j} \times idf_i \tag{3}$$

TFIDF is used to select the top 50% cluster. This proportion is high enough to maintain a reasonable heterogenity of categories regarding a phrase and low enough to consider all the relevant information without much loss thus, maintaining a balance between relevancy and heterogenity.

3.4 Step 4 - Merging Clusters

The merging approach that we followed is based on 'Hierarchical Agglomerative Clustering' (HAC)[9]. HAC works by first creating a similarity score metric for the identified base clusters (a table showing how 'similar' or compatible 2 different clusters are to one another), the algorithm then merges the most compatible pair into a new cluster. The similarity score table then will be updated with the newly merged cluster's score, and the merging process will repeat until no more clusters can be merged.

HAC scoring metric was implemented by using single linkage clustering (also known as nearest neighbor technique) to calculate the distance between two clusters (the smaller the distance score is, the more compatible or similar, both cluster will be to each other).

A variant of "Normalized Google Distance" [36] is used to calculate the score metric. Normalized Google Distance is a very helpful and effective way of calculating a similarity distance score between 2 terms. We adopted it to calculate the distance between two clusters. It takes into consideration the number of documents each cluster contains. In other words, if the word appears in too many documents, it would be penalized. This also helps in filtering out common words. Suppose the documents in one cluster share the phrase x, in another cluster, documents share phrase y, then the distance between the clusters is calculated as

$$NGD_{(x,y)} = \frac{\max\{\log f(x), \log f(y)\} - \log f(x,y)}{\log M - \min\{\log f(x), \log f(y)\}} \tag{4}$$

Where f(x) and f(y) are the total counts of search for phrases'x'and 'y' respectively, while f(x,y) is the total counts of documents containing both terms and 'M' is the total number of documents in the collection.

3.5 Step 5 - Cluster Cleaning

A limitation of HAC is that once two clusters are merged, they can never be separated. So any early mistakes can not be recovered. After the clusters are merged using HAC, we may have noisy clusters containing a number of unrelated documents. To solve this problem,we used a method called Clusters Cleaning. Its purpose is to remove unrelated documents from each cluster. Daniel et. al used the relevance measure to compute the relevance of each document in a cluster and to remove the documents having irrelevant information[27]. The relevance of a document to its cluster is based on the number and size of its cluster's base clusters of which it is a member. The formula to calculate the relevance of each document to its cluster is as follow:

$$relevance(p,c) = \frac{\sum_{\{b|b\in base(c)\wedge p\in b\}} |b|}{\sum_{b\in base(c)} |b|} \tag{5}$$

Where 'p' is a document in cluster 'c' and 'b' is a base cluster and |b| is a document in 'b'. Page or document relevance ranges from 0 to 1, where 0 means a completely irrelevant page. Page relevance is calculated as the sum of the size of the cluster's base cluster of which it is a member, divided by the sum of the size of all of the cluster's base clusters[27].

Once a page is declared irrelevant to its cluster (ie. the relevant score is less than a relevance threshold,say 20%), it is removed from that cluster and put into another cluster with the highest compatible score (using Normalized Google Distance formula).

```
MergeThreshold mt = positive number;
FilterThreshold ft = positive number;
RelevanceThreshold rt = positive number;

function STHAC (DocumentSet ds){
        Clusters base = identify Base Cluster using text from
        each document in ds to populate suffix tree;

        FOR each tree node in base{
        CALCULATE node's score with TFIDF formula;
        }

        base = RANK clusters by their TFIDF's score;
        REMOVE ft% of total clusters in base with lowest TFIDF score;

        CREATE a score metric table sm by computing similairty score between each cluster with NGD formula;

        WHILE there are still at least a pair of cluster with NGD's score of above mt DO{
                MERGE a cluster pair with highest NGD's score together;
                UPDATE sm with newly formed cluster;
        }

        FOR each cluster that are left DO{
                FOR each document in each cluster DO{
                 CALCULATE the document relevance score to its parent cluster;
                        IF score is lower than rt THEN
                                REMOVE the document off the cluster
                }
                CREATE a new cluster from removed documents;
                ADD the new cluster into another cluster with highest NGD score btween them.
        }

}
```

Fig. 2. STHAC Algorithm

4 Evaluation

4.1 Comparison STHAC with STC and Other Algorithms

We evaluated STHAC by comparing its performance against two other algorithms; Suffix Tree Clustering (STC) and Extended Suffix Tree Clustering (ESTC) on 2 test data set collected from Google search. We used two queries, 'Jaguar' and 'Salsa'. The queries were chosen because of their ambiguous meaning and various contexts, for example, 'jaguar' could mean either a car or an animal or a game. These ambiguous meanings make them ideal queries to evaluate clustering performance.

To evaluate the performance of STHAC we used 3 performance measures namely, Precision, Recall and F-Measure[9, 41], which are three widely used statistical measures to evaluate the performance of information retrieval systems.

To avoid any bias towards the selection of test data, we choosed the data provided by[42]. The test data set that was used to evaluate STHAC's performance consisted of 800 different web page documents containing a total of 53 categories[42]. The length of each web page document varied, depending on the URL that was used to obtain them.

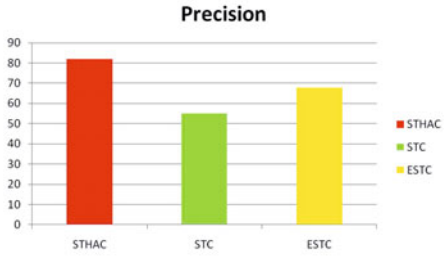


Fig. 3. Average Precision Measure

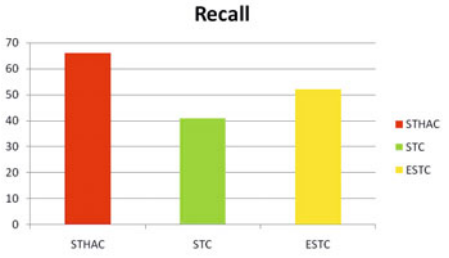


Fig. 4. Average Recall Measure

4.2 Experimental Results

On Average Precision scores, STHAC (84%) outperformed its nearest competitor, ESTC(69%) by 15% as shown in fig 4. On Average Recall score, STHAC(67%) again outperformed its nearest competitor ESTC(52%) by 14% as shown in fig 5. On average F-Measure score, which is the average performance between precision and recall, STHAC(74%) showed a 16% advantage to the second best performing clustering algorithm, ESTC(58%). Similar to STHAC, both STC and ESTC also use suffix tree data structure for identifying base clusters. With better methods of filtering out low scoring clusters and better cluster merging algorithm, STHAC outperformed the other two clustering algorithms in both Rrecision and Recall scores.

4.3 Further Experiments

In order to further investigate the contributions of each modification we used two data sets for this experiment. One data set contains 200 web page documents of a query 'Jaguar', which contains five sub-categories, 'Car', 'Game', 'Macintosh, 'Animal' and 'Maya'. Another data set contains 200 web page documents of a query 'Salsa', which contains four sub-categories, 'Music', 'Dance', 'Food', and 'Language'. For this experiment, we were only interested in analysing the effect of using different threshold values during various steps of STHAC algorithm, thus the number of documents per dataset have been reduced to save computational time.

Further Experiment Results: we selected four different modifications to analyse their effects on the overall clustering porcess:

– Effect of Cleaning Step, (fig 6): Upper graph reading is the system with cleaning step whereas the lower graph reading is the same system without cleaning step. F-measure score shows that the performance with cleaning step is much better. Results showed that the optimal value of F-measure is observerd when the value of merg Threshold is 0.05. After this, the value of F-measure degrades very quickly. So for our clustering algorithm performance we set this value equal to 0.05.

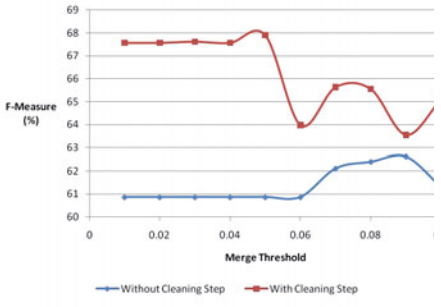


Fig. 5. With and without Cleaning Step

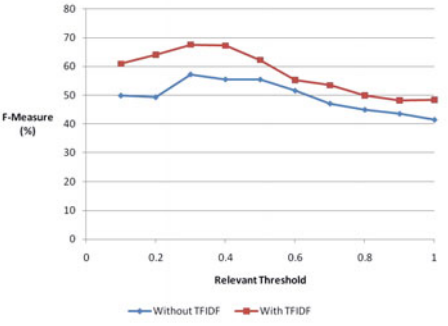


Fig. 6. With and without TFIDF

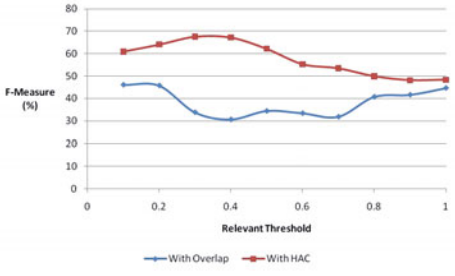


Fig. 7. With and without HAC

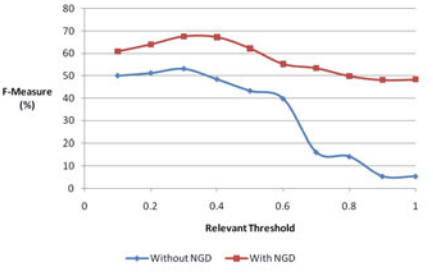


Fig. 8. With and without NGD

- Effect of HAC Merge, (fig 8): STC's original overlapping similarity measure was used in place of HAC merging method to obtain clustering results without HAC merge. The original similarity measure stated that if two clusters contain more than half of the same set of documents, then they can be merged together to form a new cluster. Clearly, the HAC algorithm performed better then STC with overlapping specially when relevance threshold is set to 0.04.
- Effect of NGD, (fig 9): For Comparison, the percentage of overlapping documents between each cluster was used to calculate clusters similarity score table (for HAC merge) instead of NGD to obtain the clustering results without NGD. Results showed that the F-measure is consistently better with NGD then that of without NGD on the relevance threshold value between 0.3 and 0.5.
- Effect of TFIDF, (fig 7): To obtain STHAC clustering results that did not used TFIDF, the original cluster scoring function of STC was used to score each cluster. The results were again better then the one without TFIDF when the relevance threshold is set to 0.4.

5 Conclusions

This paper has presented STHAC, Suffix Tree based Hierarchical and Agglomerative Clustering that has made four key contributions to the conventional

Suffix Tree Clustering to improve overall cluster quality and hence clustering performance: 1) STC's score function for filtering base clusters is replaced by standard TFIDF which resulted in more coherent clusters 2) For Cluster merging phase STC's merging process is replaced by a classic clustering algorithm HAC(Hierarchical Clustering Algorithm) which is simple as well as efficient. 3) The similarity function for comparing each pair of clusters is replaced by Normalized Google Distance which takes into consideration the number of documents contained by a cluster and helps removing common words which were left in the preprocessing phase. This helped in speeding up the clustering process and improving individual cluster's coherency. 4) An additional cluster cleaning step is introduced at the end of clustering process to filter out irrelevant documents from the low quality clusters and adding them to their relevent clusters using a relevance formula. Our experimental results have shown that each modification has made valuable contributions to the cluster quality and the overall clustering performance is significantly enhanced. However, the size of the dataset that we have used in the performance evaluation is not significantly larg. While the results are very promising, the clustering performance of our approach depends on the values of a set of thresholds which are choosen arbitrarily. Using different threshold values can result in noticably diverse clustering performances. In future, we are interested to explore and control the behaviour of these thresholds. We are also working on an algorithm that learns these thresholds based on training examples. This will be an intuitive avenue to move in future.

References

1. Syan Chen, M., Hun, J., Yu, P.S., Ibm, J.I., Ctr, W.R.: Data mining: An overview from database perspective. IEEE Transactions on Knowledge and Data Engineering 8, 866–883 (1996)
2. Sebastiani, F., Ricerche, C.N.D.: Machine learning in automated text categorization. ACM Computing Surveys 34, 1–47 (2002)
3. Zamir, O., Etzioni, O.: Web document clustering: a feasibility demonstration. In: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 1998, pp. 46–54. ACM, New York (1998)
4. Crabtree, D., Gao, X., Andreae, P.: Improving web clustering by cluster selection. In: Proceedings of 2005 IEEE/WIC/ACM International Conference on Web Intelligence, pp. 172–178 (2005)
5. Andersson, A., Larsson, N.J., Swanson, K.: Suffix trees on words. In: Hirschberg, D.S., Meyers, G. (eds.) CPM 1996. LNCS, vol. 1075, pp. 102–115. Springer, Heidelberg (1996)
6. Arne Andersson, S.N.: Efficient implementation of suffix trees. Software Practice and Experience 25, 129–141 (1995)
7. Hung Chim, X.D.: A new suffix tree similarity measure for document clustering. ACM 978-1-59593-654 (2007)
8. Lin, D.: Automatic retrieval and clustering of similar words. In: Proceedings of the 17th International Conference on Computational Linguistics, COLING 1998, vol. 2, pp. 768–774. Association for Computational Linguistics, Stroudsburg (1998)

9. Manning, C.D., Prabhakar Raghavan, H.S.: Hierarchical agglomerative clustering, http://nlp.stanford.edu/IR-book/html/htmledition/hierarchical-agglomerative-clustering-1.html (2008)
10. Hammouda, K., Kamel, M.: Efficient phrase-based document indexing for web document clustering. IEEE Transactions on Knowledge and Data Engineering 16(10), 1279–1296 (2004)
11. Hammouda, K., Kamel, M.: Phrase-based document similarity based on an index graph model. In: Proceedings of 2002 IEEE International Conference on Data Mining ICDM, pp. 203–210 (2002)
12. Chim, H., Deng, X.: A new suffix tree similarity measure for document clustering. In: Proceedings of the 16th International Conference on World Wide Web, WWW 2007, pp. 121–130. ACM, New York (2007)
13. Aas, K., Eikvil, L.: Text categorization: A survey. Technical Report 941, Norwegian Computing Center (1999)
14. Chim, H., Deng, X.: Efficient phrase-based document similarity for clustering. IEEE Transactions on Knowledge and Data Engineering 20(9), 1217–1229 (2008)
15. Salton, G., McGill, M.: Introduction to modern information retrieval. McGraw-Hill (1983)
16. Huang, A.: Similarity measures for text document clustering, pp. 49–56 (2008)
17. Joydeep, A.S., Strehl, E., Ghosh, J., Mooney, R.: Impact of similarity measures on web-page clustering. In: Workshop on Artificial Intelligence for Web Search, AAAI, pp. 58–64 (2000)
18. McCreight, E.M.: A space-economical suffix tree construction algorithm. Journal of the ACM 23(2), 262–272 (1976)
19. Janruang, J., Guha, S.: Semantic suffix tree clustering. In: First IRAST International Conference on Data Engineering and Internet Technology, DEIT (2011)
20. Carpineto, C., Osinski, S., Romano, G., Weiss, D.: A survey of web clustering engines. ACM Computing Surveys 41, 1–38 (2009)
21. Zamir, O., Etzioni, O.: Grouper: A dynamic clustering interface to web search results. In: Proceedings of the Eighth International World Wide Web Conference, pp. 283–296. Elsevier, Toronto (1999)
22. Weiner, P.: Linear pattern matching algorithms. In: IEEE Conference Record of 14th Annual Symposium on Switching and Automata Theory, SWAT 2008, pp. 1–11 (1973)
23. Wang, J., Li, R.: A New Cluster Merging Algorithm of Suffix Tree Clustering. In: Shi, Z., Shimohara, K., Feng, D. (eds.) Intelligent Information Proceesing III. IFIP AICT, vol. 228, pp. 197–203. Springer, Boston (2006)
24. Zhang, D., Dong, Y.: Semantic, Hierarchical, online Clustering of Web Search Results. In: Yu, J.X., Lin, X., Lu, H., Zhang, Y. (eds.) APWeb 2004. LNCS, vol. 3007, pp. 69–78. Springer, Heidelberg (2004)
25. Osinski, S., Weiss, D.: A concept-driven algorithm for clustering search results. IEEE Intelligent Systems 20, 48–54 (2005)
26. Sameh, A.: Semantic web search results clustering using lingo and wordnet. International Journal of Research and Reviews in Computer Science (IJRRCS) 1(2) (June 2010)
27. Crabtree, D., Andreae, P., Gao, X.: Query directed web page clustering. In: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2006, pp. 202–210. IEEE Computer Society, Washington, DC, USA (2006)

28. Kale, A., Bharambe, U., SashiKumar, M.: A new suffix tree similarity measure and labeling for web search results clustering. In: 2nd International Conference on Emerging Trends in Engineering and Technology (ICETET), pp. 856 –861 (2009)
29. Wu, J., Wang, Z.: Search results clustering in chinese context based on a new suffix tree. In: Proceedings of the 2008 IEEE 8th International Conference on Computer and Information Technology Workshops, pp. 110–115. IEEE Computer Society, Washington, DC, USA (2008)
30. Alsabti, K.: An efficient k-means clustering algorithm. In: Proceedings of IPPS/SPDP Workshop on High Performance Data Mining (1998)
31. Ng, A.Y., Jordan, M.I., Weiss, Y.: On spectral clustering: Analysis and an algorithm. In: Advances in Neural Information Processing Systems, vol. 14, pp. 849–856. MIT Press (2001)
32. Beil, F., Ester, M., Xu, X.: Frequent term-based text clustering. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2002, pp. 436–442. ACM, New York (2002)
33. Milligan, G.W., Sokol, L.: A two stage clustering algorithm with robust recovery characteristics. Educational and Psychological Measurement 40, 755–759 (1980)
34. Kessler, B.: Computational dialectology in irish gaelic. Computing Research Repository cmp-lg/950, 60–66 (1995)
35. Everitt, B.S., Landau, S., Leese, M.: Cluster analysis, 4th edn. Oxford University Press (1993)
36. Cilibrasi, R., Vitanyi, P.: The google similarity distance. IEEE Transactions on Knowledge and Data Engineering 19(3), 370–383 (2007)
37. Willett, P.: Recent trends in hierarchic document clustering: A critical review. Information Processing and Management 24(5), 577–597 (1988)
38. Kjos-hanssen, B., Evangelista, A.J.: Google distance between words. Computing Research Repository abs/0901.4 (2009)
39. Gerard Salton, C.B.: Term-weighting approaches in autumatic text retrieval. In: Information Processing and Management, vol. 24, pp. 513–523. Pergamin Press plc, Great Britain (1988)
40. Machado, D., Barbosa, T., Pais, S., Martins, B., Dias, G.: Universal Mobile Information Retrieval. In: Stephanidis, C. (ed.) UAHCI 2009, Part II. LNCS, vol. 5615, pp. 345–354. Springer, Heidelberg (2009)
41. Losee, R.M.: When information retrieval measures agree about the relative quality of document rankings. Journal of the American Society for Information Science 51(9), 834–840 (2000)
42. Crabtee, D.: Raw data (2005), http://www.danielcrabtree.com/research/wi05/rawdata.zip

Individual Doctor Recommendation Model on Medical Social Network

Jibing Gong[1,2] and Shengtao Sun[1]

[1] Department of Computer Science and Engineering, Yanshan University, Qinhuangdao, Hebei, 066004, China
gongjibing@gmail.com
[2] Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China

Abstract. It is difficult for patients to find the most appropriate doctor/physician to diagnose. In most cases, just considering Authority Degrees of Candidate Doctors(AD-CDs) cannot satisfy this need due to some objective preferences such as economic affordability of a patient, commuting distance for visiting doctors and so on. In this paper, we try to systematically investigate the problem and propose a novel method to enable patients access such intelligent medical service like this. In the method, we first mine patient-doctor relationships via Time-constraint Probability Factor Graph mode(TPFG) from a medical social network, and then extract four essential features for AD-CDs that would be subsequently sorted via Ranking SVM. At last, combining AD-CDs and patients' preferences together, we propose a novel Individual Doctor Recommendation Model, namely IDR-Model, to compute doctor recommendation success rate based on weighted average method. We conduct experiments to verify the method on a real medical data set. Experimental results show that we obtain the better accuracies of mining patient-doctor relationship from the network, AD-CDs ranking is also better than the traditional Reduced SVM method, and doctor recommendation results of IDR-Model is reasonable and satisfactory.

1 Introduction

With the rapid development of economy, people pay more and more attention to their health conditions. Due to limited medical resource, intelligent and efficient medical services are increasedly required. However, most patients have no ideas to find appropriate doctor to diagnose. It is common for patients to cure little diseases with high economic cost. Besides, patients often see wrong doctors, which results in the waste of public medical resources. One the one hand, some patients have blind faith in high level hospitals and so-called medical experts/doctors, and further cure their minor illnesses at the cost of so much money. Other patients often neglect of appropriate doctors nearby and instead seek those ones who are far away from them. Consequently, they not only spend more money but also waste their limited attention. On the other hand, doctors often confront

J. Tang et al. (Eds.): ADMA 2011, Part II, LNAI 7121, pp. 69–81, 2011.

such situations in which his/her patients' disease types are not consistent with his/her diagnosis scope. And these situations will greatly lower both medical treatment efficiency and utilization rate of insufficient medical resources. From the patient's point of view, some their own preferences, e.g. economic affordability of a patient, should be taken into consideration to find the most appropriate doctors for them. So, we can draw the conclusion that automatic and intelligent doctor recommendation service is urgently needed by all of us.

Individual doctor recommendation has already been an emerging research topic. However, main challenges/difficulties still exist as follows:

- The first is how to measure authority degree of a doctor. That is, how to mine patient-doctor relationship from the network and further measure doctors' real authorities would become one of main challenges in this work.
- The second is how to make individual doctor recommendation according to patients' preferences information and doctors' AD-CDs since traditional information retrieval models, such as Boolean Model[1] and Vector Space Model[2], all compute similarity degree between query keywords and destination doctors.
- The last is how to evaluate the recommendation accuracies of our method because it is hard to obtain solid results via evaluating AD-CDs for traditional methods. Besides, these methods themselves just are subjective processes.

To address these above challenges, we try to systematically investigate the problem of individual doctor recommendation with the following contributions:

- On the basis of our mined doctor-patient relationships via TPFG Model[4] from a medical social network, we design and extract essential features of Authority Degrees of Candidate Doctors(AD-CDs), and present a new approach based on Ranking SVM[7] to calculate AD-CDs.
- We systematically analyze what preferences patients would have when they are going to find doctors, and formalize them. Combining AD-CDs and these preferences, we propose a novel weighted-average-based approach, namely IDR-Model, for solving the problem of individual doctor recommendation.
- We propose a novel and integrated recommendation method to help patients find the most appropriate doctor to diagnose. The method is a broad topic that involves many techniques, and it also is practical technology for efficient and intelligent medical information service.

2 The Method

Our proposed method includes the following layers from bottom to top: 1) The first layer is data source layer which can be also called medical social network layer. It is comprised of a real disease/illness cases dataset with 256 size and 219 valid questionnaires from patients. The feedback information includes Attitude toward Patients, Reasonable Price, Diagnosis Efficiency, Medical Technical Level, and Curative Effect. These information is used to compute Satisfaction

Degree. 2)In the second layer, we utilize TPFG Model[4] to mine patient-doctor relationships from the above dataset. 3) The four essential features are extracted from the mined relationships in the third layer.They will be later used in our individual doctor recommendation model. 4) In the forth layer, we use Ranking SVM[7] to train the weights of all features and compute AD-CDs. 5) In the last layer, combining these above sorting results and patients' preferences together, we propose individual doctor recommendation model(IDR-Model) based on weighted average method to help a patient find the most appropriate doctor.

2.1 Doctor-Patient Relationships Mining via TPFG

Patient-doctor relationship mining is the basis of accurately extracting features to compute Authority Degree of Candidate Doctors(AD-CDs). The problem can be formalized as: input time-correlated cooperation relationship network $G = (V = V^p \cup V^a)$, where $V^p = \{p_1, p_2, \cdots, p_{n_p}\}$ denotes the set of disease case, the diagnosis time of p_i is expressed by t_i, $V^a = \{a_1, a_2, \cdots, a_{n_a}\}$ stands for the set of all participants during treatment, as well as E is the edge set(that is, $e_{ji} \in E$). Here, e_{ji} links p_i and a_j together, and denotes a_j is one of all participants in the disease case p_i. The output of this model is a directed acyclic graph $H = (V', E'_s, (r_{ij}, st_{ij}, ed_{ij})_{(i,j)\in E'_s})$, where H is a subgraph of G and $E'_s \in E'$. In the edge-correlation information $(r_{ij}, st_{ij}, ed_{ij})$, r_{ij} indicates the probability that participant a_j is doctor a_i, as well as st_{ij} and ed_{ij} denote the starting time and end time of patient-doctor relationship duration, respectively.

This paper applies Time-constraint Probability Factor Graph Model (shortly TPFG)[4] to mine patient-doctor relationships. In this model, for each node a_i, three variables, y_i, st_i and ed_i, need to be solved. Given a region feature function $g(y_i, st_i, ed_i)$, to reflect all joint probabilities in the relational graph, we define the joint probability as the product of all region feature functions, as in (1).

$$P(\{y_i, st_i, ed_i\}_{a_i \in V^a}) = \frac{1}{Z} \prod_{a_i \in V^a} g(y_i, st_i, ed_i) \tag{1}$$

where $1/Z$ indicates the normalization factor of the joint probability, and there two basic assumptions exist: (1) $\forall 1 \leq x \leq n_a, py^1_{yx} < py^1_x$ indicates that a patient knows dianosis results later than his/her doctor. (2) $\forall a_i \in V^a, ed_{yi} < st_i < ed_i$ tells that a patient obtains diagnosis result/information later than his/her doctor. In this equation, to obtain most probable values of all unknown factors, the joint probability need to be maximized. A great deal of unknown parameters would lead to too large-scale solution space if the joint probability problem above wasn't simplified before being solved. Therefore, we simplify the joint probability problem into the following equation: Suppose if a patient a_i and his/her doctor y_i are determined, so we can obtain $\{st_i, ed_i\} = \arg \max_{st_i < ed_i} g(y_i, st_i, ed_i)$, that is, st_i and ed_i can be well solved.

Before working out this joint probability, we first compute st_i and ed_i which are contained in every possible pair of patient-doctor relationship, and then we can get joint probability formula with simplified parameters, as in (2).

$$P(y_1, y_2, \cdots, y_{n_a}) = \frac{1}{Z} \prod_{i=1}^{n_a} f_i(y_i | \{y_x | x \in Y_i^{-1}) \tag{2}$$

where $f_i(y_i|\{y_x|x \in Y_i^{-1})$ is the product of $g(y_i, st_{ij}, ed_{ij}$ and $\hbar$ is a continuous product(That is, $\prod_{x \in Y_i^{-1}} \hbar$). If $y_x \neq i \vee ed_{ij} < st_{xi}$, $\hbar = 1$, otherwise $\hbar = 0$. After simplifying formula (1), we can use probability factor graph[5] to solve formula (2). The graph mapped by formula (2) contains two types of nodes in the factor graph: variable nodes and function nodes. Variable nodes are corresponding to hidden variables $\{y_i\}_{i=0}^{n_a}$. Every variable node links one function node $f_i(y_i|\{y_x|x \in Y_i^{-1}\})$, which indicates $f_i(y_i|\{y_x|x \in Y_i^{-1}\})$ is determined by y_i. Besides, the probability factor graph includes one kind of dependence relationship between variables and functions.

In the graph, a patient-doctor relationship is expressed by the probability of any two cooperators (that is, a patient and a doctor), the starting time of the relationship and the end time of the relationship. To compute these factors, we introduce two usual merits, **Kulc**(Kulczinski measuree[15]) and **IR** (Imbalance Ratio).**Kulc** is used to measure the closeness degree of two cooperators, and **IR** means the imbalance degree between the probability of a_i being the cooperator of a_j and the probability of a_j being the cooperator of a_i. To mine this kind of relationships, we consider the time, when a doctor diagnose a patient for the first time, as the starting time st_{ij} of the patient-doctor cooperation relationship. And then we define the time, when the value of **Kulc** begins to decline or greatly changes, as the end time ed_{ij}. After finalizing st_{ij} and ed_{ij}, the probabilities of the patient-doctor cooperation relationships can be computed by formula (3).

$$p_{ij} = \frac{\sum_{st_{ij} \leq t \leq ed_{ij}} (kulc_{ij}^t + IR_{ij}^t)}{2(ed_{ij} - st_{ij} + 1)} \tag{3}$$

2.2 Features Fxtraction for Calculating AD-CDs

To measure AD-CDs and build an individual doctor recommendation model, this paper designs the following four features and extracts them from mined patient-doctor relationships.

1. The feature *DomainRel*, describes matching degree between doctors' diagnosis scope and disease types of all patients. It is measured from the doctor's point of view and computed using ACT Model[6].
2. The feature *M-index* indicates influence index of a doctor in healthcare community. Specifically, the *M-index* value is equal to m, if one medical technology of a doctor had been used by other doctors for more than m times in no less than m disease cases, and instead, use times of all other technologies were less than m.
3. The feature *Uptrend* states the uptrend index of a doctor's diagnosis achievements[9]. It is calculated by both formula (4) and (5).

$$Uptrend(D) = Avg(AoT(U(D))) - Curve(U(D)) * Avg(c_i) \tag{4}$$

$$Curve(D) = \frac{\sum_{i=1}^{N}(c_i * AoT(U_{ty-N+i}(D))) - N\overline{c} * \overline{AoT(U(D))}}{\sum_{i=1}^{N}(AoT(AoT(U_{ty-N+i}(D)))^2) - N\overline{AoT(U(D))}^2} \quad (5)$$

where $U(D)$ denotes the set of successfully cured cases from one doctor D, $Curve(D)$ indicates a fitted curve which is generated using Least Square Method from all successfully cured cases in latest N years, c indicates the increment of years number, U_{ty} means the set of all successfully cured cases in this year, as well as $\overline{AoT(U(D))}$ states the average value of overall evaluation for $U(D)$ given doctor D.

4. The feature *Activity* is the activity index of latest disease cases performed by a doctor. It is computed by the formula (6).

$$Activity(D) = \sum_{latestNyears} AoT(U_n) * w(n) \quad (6)$$

where U_n denotes the set of all disease cases in the Nth year, $w(n)$ is the weight value of the nth year, and N indicates the latest N years.

2.3 Authority Degrees Sorting via Ranking SVM

We select classic learning ranking method, Ranking SVM[7,8], as basic framework for AD-CDs sorting. To address the problem, Ranking SVM creates a new instance $(x_i^a - x_i^b, z_i)$ for $(x_i^a - x_i^b)$ which is the instance of $(y_i^a - y_i^b)$ with two different ranking levels in query keywords. z_i satisfies: $z_i = +1$ if $y_i^a > y_i^b$, else $z_i = -1$. After building a new training set $\Gamma' = \{(x_i^a - x_i^b, z_i)\}_{i=1}^n$, it is feasible that classic Ranking SVM(Support Vector Machine) is employed to solve the ranking function $f =< \overrightarrow{w}^*, \overrightarrow{x}' >$. That is, sub-optimization problem needs to be solved, as in (7).

$$\min_{\overrightarrow{w}} M\overrightarrow{w} = \frac{1}{2}\|\overrightarrow{w}\| + C\sum_{i=1}^{1}\xi_i \quad (7)$$

where ξ_i indicates the ranking error of a ranking function, S means the set of new instances inputted, x' indicates a new input ($x' \in S$), C stands for user-defined parameters of SVM, every x denotes one instance, (x^a, x^b) is a pair of instances, (x_i^a, x_i^b) is the ith pair of instances, $x^{(1)} - x^{(2)}$ is the pair of instances which is comprised of two adjacent instances, y indicates the ranking level of every instance, (y^a, y^b) states ranking relationships between any two instances, (y_i^a, y_i^b) means the ranking relationship of the ith pair of instances, as well as rank relationships may be aliased as Z, $z = 1$ means the ranking level of x^a is higher than x^b, and inversely, $z = -1$ means the ranking level of x^a is lower than x^b. Through training SVM, weight vectors $\overrightarrow{w}^*$, which are corresponding to all feature values in the model shown in (7), might be worked out. Their ranking scores, Authority Degrees of doctors, would be computed using this model. And further, we can sort these authority degrees successfully.

2.4 Individual Doctor Recommendation via Weighted Average Method

Our sorting model based on Ranking SVM can compute and sort Authority Degree of Doctors(AD-CDs) according to query keywords inputted by patients. However, this model is still not an individual doctor recommendation model because patients' preference information is not considered yet. Therefore, it is important to compute success rate of finding the most appropriate doctor according to both patients' preference information and doctors' features. We present a novel Individual Doctor Recommendation Model(IDR-Model) based on weighted average method. IDR-Model computes success rate of finding the most appropriate doctor using probabilities of four preferences/factors including Economy Matching Degree, Medical Domain Matching Degree, Recommender Influence, and Region Reference. The model is defined as in (8).

$$\begin{aligned} success\_rate(P,D) = (\ & a\_score(P,D) \times a\_weight \\ & + m\_score(P,D) \times m\_weight \\ & + c\_score(P,D) \times c\_weight \\ & + l\_score(P,D) \times l\_weight) \\ & / \ (a\_weight + m\_weight + c\_weight + l\_weight) \end{aligned} \tag{8}$$

where initial values of all weights are set to 1. With receiving feedbacks from patients, the model would use feedback data to optimize these weight values. These four preferences are computed using different methods. They are completely described below.

1. ***Economy Matching Degree(EMD)*** is used to measure matching degree between a patient's economic affordability and treatment expenditure. Computing EMD preference needs to map all kinds of information from both patients and doctors into one same number interval so that comparisons might be made at one same order of magnitude. Therefore, *Sigmoid* function is used to map all inputs of real number field into the interval [0,1]. And then rank fraction function is applied to map ranking information into the interval [0,1]. They are defined as in (9) and (10), respectively.

$$sigmoid : S(t) = \frac{1}{1+e^{-t}} \tag{9}$$

$$rank\_score : R(r,c) = 1 - \log_{10}(1 + 9 \times \frac{r-1}{c-1}),\ c > 1 \tag{10}$$

We first adjust input errors of patients, and then carry out formatting and normalization processes. At last, combining doctors' information, we compute *gpa_score*—the percentage transferred from monthly salary, *i_score*—the index transferred by overall income of patients, *s_rank*—the salary grade of patients, *s_count*—the number of salary grades, *a_rank*— the authority degree ranking of doctors, *a_count*—the number of authority degrees, *hosp_rank*—the hospital rank, and *hosp_count*—the number of hospital ratings. The final formula for computing EMD preference is given, as in (11).

$$\begin{aligned} a\_score(P,D) = P(&P's\ gpa\_score + P's\ i\_score \\ &+ R(P's\ s\_rank,\ s\_count) \\ &- R(D's\ a\_rank, a\_count) \\ &- R(D's\ hosp\_rank, hosp\_count)) \end{aligned} \tag{11}$$

where $a\_count = 12, hosp\_count = 9$, P means a patient, D indicates a doctor, and $i\_score = \log_{10}(1 + num\_of\_months) + \sum \log_{10}(1 + gpa\_score)$.

2. ***Medical Domain Matching Degree(MDMD)*** states that the more a patient's disease type fit with his doctor's diagnosis scope, the more success rate finding the most appropriate doctor will have. In contrast with the $DomainRel$ described above, $MDMD$ is measured from the perspective of the patient. And it is computed using formula (12).

$$m\_score(P,D) = \frac{\overrightarrow{q}_P \cdot \overrightarrow{q}_D}{\|\overrightarrow{q}_P\| \|\overrightarrow{q}_D\|} \tag{12}$$

where $\overrightarrow{q}_P$ is the vector of disease types of patients and $\overrightarrow{q}_D$ is the vector of diagnosis scopes of doctors.

3. ***Recommender Influence(RI)*** indicates that the influence degree of a recommender is undoubtedly important to find the most appropriate doctor. This preference tells such a fact that the success rate will be significantly improved if a recommender was a client or colleague of one candidate doctor. The value of RI is computed using formula (13).

$$c\_score(R,D) = \begin{cases} 1 - spw(R,D), & \text{if } spw(R,D) < 1 \\ 1, & \text{if } spw(R,D) \geq 1 \end{cases} \tag{13}$$

where R denotes a recommender, D a dooctor, and spw means the weight of shortest path between R and D.

4. ***Region Reference(RR)*** reflects such a common sense that a patient is inclined to visiting those doctors whose working place are not far away from them. RR is formalized as formula (14).

$$l\_score(R,D) = \begin{cases} \frac{\#D's\ ps\ of\ P's\ locality}{\#D's\ patients}, & \text{if } P's\ locality \in Na \\ 0, & \text{if } P's\ locality \notin Na \end{cases} \tag{14}$$

where ps indicates patients and $Na = \{Fuxinmen, Guanganmen, Jishuitan, Zhongguancun, Chongwenmen, Chichengqu, Dongchengqu\}$ means the set of regions where patients live.

3 Experiments and Evaluations

It is difficult to prepare training and testing data for studying the topic of individual doctor recommendation. So, we evaluate the Ranking-SVM-based algorithm for sorting AD-CDs and perform individual doctor recommendation model in

a real-world medical dataset with 256 size. It involves seven kinds of disease, twelve doctors, important information for curing or treatment, patients' personal information, and patients' evaluation information on doctors. Specifically, we choose 219 valid ones to calculate Satisfication Degree of Patients (SDP) among 256 questionnaires. In training our Ranking SVM model, we selecte 156 of 256 disease cases as training dataset and 100 ones as testing dataset.

3.1 Mining Patient-Doctor Relationships via TPFG

Although the evaluation for our TPFG-based method can not be done in a real-world large-scale dataset of diagnosis and treatment, we take the preprocessing problem into considerations. The preprocessing procedure mainly is to filter those connections that don't stand for doctor-patient cooperation relationships, and accordingly reduce the time and space cost performed on the TPFG-based method. Steps in this procedure are listed below.

1. During the treatment/cooperation between a_i and a_j, there exists $IR_{ij}^t < 0$ in the time serial $\{IR_{ij}^t\}_t$ of IR value.
2. During the treatment/cooperation between a_i and a_j, the length of $\{kulc_{ij}^t\}_t$ serial has no change.
3. the duration of the treatment/cooperation between a_i and a_j last more than ten days. As we know, a period of treatment often is 10 days.
4. $py_x^1 + 1 > py_{yx}^1$, that is, a patient knows the diagnosis results one day later than his/her doctor.

In this experiment, we utilize TPFG-based method to mine patient-doctor relationships. The input of the model include the twelve doctor's information in our medical dataset, the information of 256 disease cases, diagnosis times, symptom information during treatment and so on.If a pair of patient and doctor complies with the rules(given in section 2.1), we would create one edge from a_i to a_j in the patient-doctor cooperation sub-graph H', and then compute the starting time and the end time between a_i and a_j according to formula (3). After building H', we calculate the probability of every edge in H' using TPFG model. We select symbol θ as the threshold telling whether one patient-doctor relationship is true or not. The greater the value of the threshold θ is, the higher the mining accuracies is and the lower the rate of recalls is. In the experiment, we select $\theta = 0.7$ and extract totally 120 patient-doctor relationships. The mining accuracies of patient-doctor relationships extracted by our TPFG-based method are relatively low since it is just 68.5% when choosing $\theta = 0.7$.

3.2 Effectiveness of Ranking-SVM-Based Algorithm

In the training process for Ranking SVM, this paper employs a open-source SVM tool, SVMlight[10], to implement. The model takes users' standard answers as training dataset, and then generates training data through feature extraction procedure. After training the data, we finally obtain the ranking function $f =< \overrightarrow{w}^*, \overrightarrow{x}' >$ where every feature has the following weight values, as in Table 1.

Table 1. All feature values given by Ranking SVM

Feature	*M-index*	*DomainRel*	*Activity*	*Uptrend*
Value	4.8209	2.5474	1.6076	-0.5391

From Table 1, the feature *M-index* has the greatest effect on our ranking model for computing Authority Degree of doctors, which indicates a good doctor must be a medical expert to a great degree. Generally, $DomainRel = 0$ indicates matching degree(shortly **Mdegree**) between patients' disease type and doctor's diagnosis scope have no effect on Authority Degrees(AD-CDs). When $DomainRel > 0$, the higher value of *DomainRel* has, the more it help the doctor's AD-CD move to top. Here, $DomainRel = 2,5474$ means that patients are inclined to giving high scores to those doctors whose diagnosis scope is consistent with patients' disease types. $Activity = 1.6076(> 1)$ states patients prefer to visit those doctors who have higher frequency of diagnosis or treatment activities than others. This is because patients think they will have higher probabilities of making appointments with those doctors. Among these features, the value of $Uptrend(= -0.5391)$ is negative just states those medical experts are often senior doctors, and junior doctors often have less achievements than senior ones in both diagnosis achievements and treatment experiences.

To improve the speed of ranking, the model would off-line compute all features, except *DomainRel*, of every doctor, and store them into one relational database. Besides, Taking a query of Authority Degrees into consideration, the system doesn't sort AD-CDs of all doctors in the database, and instead, it first obtains the sub-dataset of disease case obtained by doctor queries, and then resorts them using our ranking model. Compared with sorting operation in the whole dataset, this strategy not only improves the efficiency of on-line computation, but also gains the ranking results with a slighter error.

3.3 AD-CDs Sorting Evaluation

In this evaluation experiment, we adopted four indexes, $P@5$,$P@10$,$P@15$, and MAP, to evaluate Ranking algorithm for authority degrees of doctors. $P@k$ indicates precision rates of the first k results that the system outputs towards inputted query keywords, and it is defined as formula (15).

$$P@k = \frac{number\ of\ disease\ cases\ in\ the\ first\ k\ results}{k} \tag{15}$$

MAP denotes average values of accuracies which are corresponding to query keywords of every disease. Specifically, given an existing query keyword, average values of precision rates will be computed according to precision rates of the first k results. Namely, MAP is the average value of AP in the whole testing dataset, where AP is described as formula (16).

$$AP = \frac{\sum\limits_{k\ is\ relavent} P@k}{data\ of\ relavent\ disease\ cases} \tag{16}$$

This paper utilizes feedback information of patients' questionnaires as testing dataset to compare with traditional Reduced SVM(RSVM)[14] method. The comparative evaluation results are shown in Fig. 1. From this Figure, we can find accuracies of our Ranking-SVM-based algorithm are better than RSVM method. Also, the comparisons means sorting effects of the most relevant results using our algorithm are better.

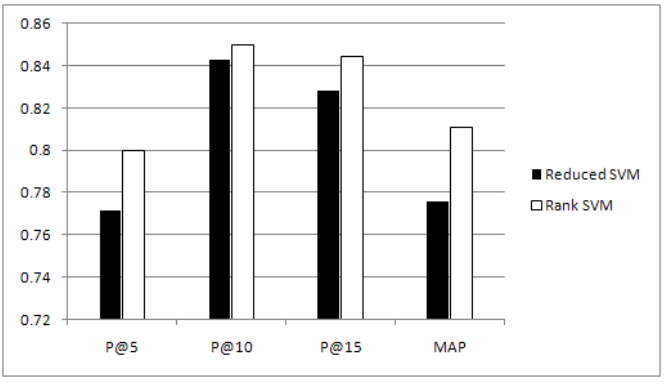


Fig. 1. Comparative evaluation results between RSVM and Ranking SVM

3.4 Recommendation Performance

In the evaluation experiment, we divided the Authority Degree of Doctor(AD-CD) results into five different levels from $1st$ to $5th$ according to sorting effects in testing dataset. In every medical domain, there are five doctors with $5th$ level, five doctors with $4th$ level, two doctors with $2nd$ level and zero doctors with other levels. Fig. 2 shows recommendation results and Authority Degrees of Candidate Doctors(AD-CDs) of the first twelve doctors when inputted keywords is 'Cirrhosis Disease' and 'Coronary Heart Disease'.

From instances analyses shown in Fig. 2, recall rates of the most relevant doctors using our IDR-Model are highest. For example, the first six results in analyses to Coronary Heart Disease almost involve all doctors with greater than 3rd level, and also analysis to Cirrhosis disease includes three doctors with greater than $5th$ level. However, our IDR-Model has some disadvantages. For instance, it could not well distinguish different recommendation degrees of doctors. Another example to illustrate it is sorting/ranking results didn't reflect the differences between AD-CDs of the testing dataset. We select one random patient and five doctors from our real medical dataset with 256 size. Here, the first three doctors are attending Coronary Heart Disease and the last two treats Cirrhosis Disease. One patient's personal and diagnosis information from our real medical dataset is illustrated in Table 2.

We compute success rates and four preferences according to formulas (8), (11)∼(14), and then display them in Table 3. From this table, Doctor **Xiaojun**

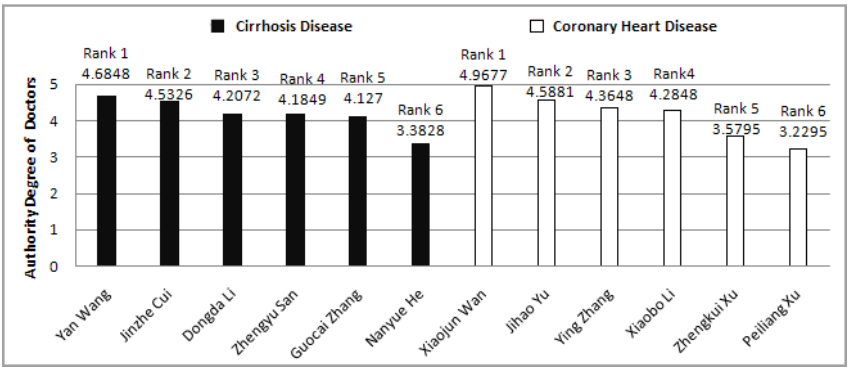


Fig. 2. Instances analyses of IDR-Model

Table 2. Illustration of personal and diagnosis information of one patient

Patient No.	*A73_519*	**Full name**	*Fen Li*	**Nationality**	*Han*
Age	*22*	**Birth place**	*Hunan*	**CHS**	*H1,H2*
Gender	*Female*	**Position**	*Student*	**Affiliation**	*LA73*
Height	*164cm*	**Weight**	*49kg*	**HBP**	*100 kPa*
LBT	*35.5 °C*	**RBT**	*35.5 °C*	**LBP**	*65 kPa*
Sleep quality	*Normal*	**Appetite**	*Normal*	**Tongue condition**	*TC1,TC2*
Right Chi	*Deep pulse*	**Left Guan**	*Float pulse*	**Urine condition**	*Normal*
Left Chi	*Deep pulse*	**Right Guan**	*Deep pulse*	**Stool condition**	$1/(3 \sim 4)$
Disease history	*Null*	**FDH**	*Null*	**Pulse condition**	*SS*
Doctor No.	*1001*	**NSDS**	*219*	**Symptoms**	*S1,S2*
EoD	*3*	**AD-CD**	*Level 5*	**Hospital location**	*Na[4]*

Wang has higher recommendation success rate than Doctor **Zhengkui Xu** and Doctor **Jihao Yu**. This is because his AD-CD and EMD values are the highest among these doctors. Simultaneously, the patient's disease type, Coronary Heart disease, is consistent with the three doctors'. Doctor **Wang** is an authoritative expert in the disease scope. This point has been already shown in Fig. 2. Therefore, it is reasonable that our model recommends him to the patient. Besides, comparing Doctor **Zhengkui Xu** with Doctor **Jihao Yu**, we find that Economy Matching Degree(EMD) and Region Reference(RR) have crucial influences on recommendation results. The two values of AD-CDs are almost large between Doctor **Xu** and Doctor **Yu**. And both of them are medical expert in the 'Coronary Heart Disease' medical field. However, Doctor **Xu** is on duty in the big hospital, and Doctor **Yu** is now working for in a medium one. Obviously, the treatment fee which the former charge will be higher than the latter. The patient's economic situation is poor because she is a student career. At the same time, the commuting distance for the big hospital is longer than the medium one. That is, the RR value of the former($RR = 0.328$) is smaller than the

latter's ($RR = 0.746$). Therefore, Doctor **Yu** is stronger to be recommended to the patient than Doctor **Xu**. At last, the patient's disease type is not consistent with the diagnosis scopes of Doctor **Guocai Zhang** and **Nanyue He**. So, whatever other preferences vary, their *success_rate* values are too low to be recommended.

Table 3. The values of patients' four preferences computed by IDR-Model

	Xiaojun Wang	Zhengkui Xu	Jihao Yu	Guocai Zhang	Nanyue He
EMD	0.854	0.492	0.733	0.6067	0.7230
MDMD	0.725	0.8603	0.8722	0	0
RI	0.710	0.872	0.709	0.5030	0.4219
RR	0.725	0.328	0.746	0.7854	0.6243
success_rate	83%	59%	68%	13%	11%

4 Related Work

It is difficult for most patients to find the most appropriate doctor/physician to diagnose. In most cases, just considering Authority Degrees of Candidate Doctors(AD-CDs) cannot meet this need due to some objective constraints such as economic affordability, commuting distance for visiting doctors and so on. In this paper, we try to systematically investigate the problem and propose a novel method to enable patients access such intelligent medical service like this.

To address the issue, some researches would be important and meaningful. Specifically, due to limited medical resource in our society, intelligent and efficient medical services are increasedly required. So it would become more and more important researches that can intelligently help patients find the most appropriate doctor to diagnose. In existing work, most focuses on expertise search such as Craig Macdonald and Iadh Ounis's expertise search based on candidate vote[11], expertise mining from social netwok of Jing Zhang and Jie Tang et al.[12], as well as Zi Yang and Jie Tang's research on transfer learning issue from expertise search to Bole search[3].

Few researches about healthcare and medical treatment focus on social network applications and Medical informatics. Instead, they concentrate on Non-invasive diagnosis and decision support, Early detection and/or prevention of diseases, patient-Doctor relationship, early detection and management of pandemic diseases as well as public health policy formulation. To address the problem of 'information rich' but 'knowledge poor' in healthcare researches, Ref. [13] discusses the issue how e-Patient utilize health-related knowledge. However, this paper focuses on data mining issue about healthcare data, but not on medical social network. To the best of our knowledge, researches about individual doctor recommendation on patient-doctor social network were not explored yet.

5 Conclusions

In this paper, we try to systematically investigate the problem of individual doctor recommendation. We first mines patient-doctor relationships in a real medical network via TPFG Model, then proposes a Ranking-SVM-based algorithm for computing Authority Degrees of Doctors(AD-CDs), and finally, presents a novel individual doctor recommendation model, namely IDR-Model based on weighted average method. Compared with the existing Reduced SVM method, our IDR-Model has better recommendation accuracies. Experimental results show that our method can help patients find the most appropriate doctors to diagnose, and it is practical technology for efficient and intelligent medical information service.

Acknowledgments. This work is supported by Natural Science Foundation of HeBei Province under Grant No. F2010001298 and No. F2011203092.

References

1. Baeza-Yates, R., Ribeiro-Neto, B.: Modern information retrieval. ACM Press (1999)
2. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Communications of the ACM 18(11), 613–620 (1975)
3. Yang, Z., Tang, J., Wang, B., et al.: Expert2Bole: from expert finding to Bole search (Demo Paper). In: SIGKDD 2009 (2009)
4. Wang, C., Han, J.W., Jia, Y.T., Tang, J., et al.: Mining advisor-advisee relationships from research publication networks. In: SIGKDD 2010 (2010)
5. Kschischang, F.R., Member, S., Frey, B.J., et al.: Factor graphs and the sum-product algorithm. IEEE Transactions on Information Theory 47, 498–519 (2001)
6. Tang, J., Zhang, J., Yao, L.M., et al.: ArnetMiner: extraction and mining of academic social networks. In: SIGKDD 2008 (2008)
7. Herbrich, R., Graepel, T., Obermayer, K.: Large margin rank boundaries for ordinal regression. MIT Press, Cambridge (2000)
8. Joachims, T.: Optimizing search engines using clickthrough data. In: Proceedings of KDD 2002, Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 133–140. ACM Press, New York (2002)
9. Wang, Z., Tang, J., Gao, B.: How we calculate academic statistics for an expert, http://www.arnetminer.com/workshop.do?id=21
10. Joachims, T.: Training linear SVMs in linear time. In: Proceedings of the ACM Conference on Knowledge Discovery and Data Mining, KDD (2006)
11. Macdonald, C., Ounis, I.: Voting for candidates: adapting data fusion techniques. In: CIKM 2006 (2006)
12. Zhang, J., Tang, J., Li, J.: Expert Finding in a Social Network (poster). In: Kotagiri, R., Radha Krishna, P., Mohania, M., Nantajeewarawat, E. (eds.) DASFAA 2007. LNCS, vol. 4443, pp. 1066–1069. Springer, Heidelberg (2007)
13. Healthcare Data Mining - Prediction Inpatient Length of Stay.pdf
14. Lee, Y.-J., Huang, S.-Y.: Reduced Support Vector Machines: a statistical theory. IEEE Transactions on Neural Networks 18(1), 1–13 (2007)
15. Wu, T., Chen, Y., Han, J.: Re-examination of interestingness measures in pattern mining: A unified framework. Data Mining and Knowledge Discovery (2010)

Influence Maximizing and Local Influenced Community Detection Based on Multiple Spread Model

Qiuling Yan, Shaosong Guo, and Dongqing Yang

School of EECS, Peking University, Beijing 100871, China
yqlpku@gmail.com, guoshaosong@163.com, dqyang@pku.edu.cn

Abstract. In independent cascade model, an active node has only one chance to activate its neighbors, while in reality an active node has many chances to activate its neighbors. We propose an influence diffusion model called multiple spread model, in which an active node has many activation chances. We prove that influence maximizing problem with the proposed model is submodular and monotone, which means greedy algorithm provides (1-1/e) approximation to optimal solution. However, computation time costs much due to Monte Carlo simulation in greedy algorithm. We propose a two-phase method which leverages community information to find seeds. In order to evaluate influence of a particular node, we also propose a definition of local influenced community as well as an algorithm called LICD to detect local influenced community. Experiments show that the proposed model and algorithms are both efficient and effective in problems of influence maximizing and local influenced community detection.

Keywords: Influence, independent cascade model, multiple spread model, local influenced community.

1 Introduction

Social network plays an important role in the spread of information and influence. Examples of social networks include co-authorship network, paper citation network, Wikipedia networks. Maximizing influence [4] and community detection [2, 17] are two popular research issues of social network analysis. Given a network, the goal of maximizing influence problem is to find out a subset that maximizes influence diffusing through the network, while community detection issue targets at clustering vertexes according to some quality function.

Maximizing influence has concrete application. Viral marketing [18] is a typical example. The intuition is that people perhaps buy a product because of others' influence. In order to raise popularity and sales volume of a product, retailers provide free products or preferential price to influential people, hoping that they influence other people to buy the product and to influence others, and so on. Another example is sensor location in a city water distribution network, which delivers water to households via pipes. We want to locate sensors in some locations, in order to detect accidental contamination as quickly as possible. The problem lies on location

J. Tang et al. (Eds.): ADMA 2011, Part II, LNAI 7121, pp. 82–95, 2011.

selection strategy. Formally, given a parameter k, the problem of maximizing influence is to find a k-node subset, where influence is diffused according to an influence cascade model.

Independent cascade model [1] is a popular model in influence maximizing. In independent cascade model, a vertex is either active or inactive. An active vertex v has only one chance to activate each neighbor w with probability $p_{v,w}$. If v succeeds at time step t, w becomes active at time step $t+1$. The activation process goes on until no more vertexes can be activated. If a vertex is activated, we called it influenced. Note that in independent cascade model, an active vertex has only one chance to activate its neighbor, while in reality, a vertex can influence others repeatedly. For example, in viral marketing, the case that an influencer fails to influence a person does not mean the person refuses a product. As long as the influencer keeps trying to influence him, it is possible for him to accept it later on. As a result, a person perhaps decides to buy a product because his friend appraises it many times, despite his friend fails for the first time. For this reason, we propose a multiple spread model.

In multiple spread model, an active vertex v has multiple chances to activate its inactive neighbor w. At each time step, v activates w with different probability. The probability attenuates along time, which conforms to the fact that influence effect weakens gradually. We prove theoretically that the problem of maximizing influence based on multiple spread model is submodular and monotone, which guarantees (1-1/e) approximation to optimal solution with greedy algorithm.

Computation cost in greedy algorithm is intolerable, since Monte Carlo simulation targets at the whole network. When the network size is large enough, it would take a long time to run greedy algorithm. Intuitively, influencers should scatter across the whole network, rather than meet in the same community. As a result, we propose a two-phase method that combines community detection and maximizing influence. In the first phase, we partition the network into k communities. In the second phase, greedy algorithm is applied to each community to find an influencer from each community.

Community detection is another issue in this paper, which also has wide applications in computer science, biology, physics, and politics and so on. A publicly accepted definition of community is that there are dense connection inside a community while sparse connection between communities. In biology, a community represents a functional module. In social network, a community can be a group of people sharing common interest or similar political opinion.

The task of community detection is to cluster vertexes in the network while optimizing some quality function. There exist numerous algorithms to mine communities in networks, most of which are to optimize a global quality function, such as modularity [3]. However, when network size is too large to evaluate a quality function or parts of network is missing, community detection becomes challenging. In this case, mining communities based on a global quality function is infeasible. Local community detection that leverages local network structure emerges for this case.

Given a query vertex, the goal of local community detection is to mine a local community starting from the vertex. Several local methods were proposed [10-14]. Common framework of them is given a query vertex, to add a vertex each time based

on some quality function. Our work is related to [14] in that both works use activation starting from a query vertex. Our difference from [14] is that formation of local community is based on influence diffusion model, which is the reason that we call local influenced community. We proposed a local influenced community detection algorithm based on multiple spread model, called LICD. In LICD, a particular query vertex propagates its influence through local networks. When a vertex is activated directly or indirectly, it joins the local community. Initially, the vertex tries to activate its neighbors, when it succeeds, its neighbors is directly influenced by it and try to activate their neighbors. When other vertexes are activated, we call them influenced indirectly. Influence of the query vertex diffuses like this. Local community investigated by LICD exposes the influencing range of the query vertex. To our best knowledge, we are the first to detect local community based on influence.

In summary, our contribution consists of three parts:

1. We propose an influence diffusion model, called multiple spread model, in which an active vertex has multiple chances to activate its neighbors.
2. We propose a two-phase method that combines community detection and greedy algorithm to find influencers.
3. A definition of local influenced community detection is proposed, along with a detection algorithm called LICD.

We apply the proposed model to both problems of influence maximizing and local influenced community detection in realistic network. Experiments show that it outperforms independent cascade model in effectiveness. Meanwhile, greedy algorithm combining with community information reduces computation time greatly while influence size approximates naïve greedy algorithm.

The paper is organized as follows. Related work is reviewed in section 2. Section 3 describes details of multiple spread model. In section 4, a method of maximizing influence is proposed, which combines community detection and greedy algorithm. Section 5 details local influenced community detection based on multiple spread model. We report experiment performance in section 6. Finally, we offer conclusions and future work in section 7.

2 Related Works

2.1 Independent Cascade Model (ICM)

Among many diffusion models, ICM [1] is most investigated. In ICM, given an initial active set A, the diffusion process unfolds according to the following course. If a node v is active at time step t, it gets only one chance to activate each inactive neighbor w. v succeeds with probability $p_{v,w}$. If it succeeds, w will become active in time step $(t+1)$. If w has many newly active neighbors, they try to activate w in an arbitrary sequence order. Whenever v succeeds or not, it does not have another chance to activate w anymore. The process continues until there is not any new activation. The problem of maximizing influence with ICM first proposed by Kempe [4] and is widely researched

after then. Kempe [4] shows that the influence maximization problem with ICM is NP-hard. Since the influence function is submodular and monotone, approximation solution with greedy algorithm is (1-1/e) of optimal solution.

Due to Monte Carlo simulations in greedy algorithm, running speed is limited greatly. Greedy algorithm is not suitable to large scale network as well. So there emerged a lot of work aiming at resolving the limit of greedy algorithm. Kimura [5] proposes two special case of ICM: Shortest-Path Model and SP1 Model. In these two models, a node *v* is activated only through shortest path from active nodes. Leskovec [6] exploits the submodular property of influence maximizing problem and proposed CELF algorithm, which is 700 times faster than naïve greedy algorithm. Chen [7] finds out those edges that do not participate in diffusion process and deletes them to get another graph G'. Two algorithms NewGreedyIC and MixGreedyIC are proposed based on the graph G'. Narayanam [8] considers nodes as players in coalition games and treats information diffusion process as coalition construction process. Narayanam proposes SPIN algorithm that leveraging games theory. Nodes are ranked according to their shapley values. Chen [9] proposes an IC-N model based on ICM to model negative influence. In influence maximizing problem based on IC-N model, monotone and submodular property still hold. Chen assumes that influence spreads only through arborescence area of nodes and apply greedy algorithm only in the arborescence area.

2.2 Local Community Detection

Given a graph *G* and a local community detection algorithm initiated with a query vertex *v*, each vertex *w* in *G* has one of three relations with *v*:

1. *w* is in the local community *C*.
2. *w* is in the neighborhood *N* of community *C*.
3. *w* is in unexplored area *U*.

We use symbols *C*, *N*, *U* to represent local community, neighborhood area and unexplored area, respectively. We use *B* to represent the boundary of *C*, i.e. *B* consisting of each node in *C* that has at least one edge connecting to a node in *N*. Many local community detection algorithms choose one vertex from *N* at each step to add into *C*. Their difference lies on choosing strategy. In [10], Clauset defines the local modularity *R* to be $R = I/T$, where *I* is the number of edges with endpoints in $C \cup N$, *T* is the number of edges with one or more endpoints in *B*. The intuition behind *R* is that *R* is "directly proportional to sharpness of the boundary given by *B*". Luo et al. proposed the definition of modularity of a sub-graph *S* in [11]: $M = ind(S)/outd(S)$, ratio of number of edges connecting nodes both from *C*, to number of edges connecting one node from *C* and another node outside *C*. Bagrow [12] proposed a selection criterion "*outwardness*" of a node *v*, Ω_v. $\Omega_v = (k_v^{out} - k_v^{in})/k_v$, where k_v^{in} is the number of edges from *v* to nodes in *C*, k_v^{out} is the number of edges from *v* to nodes in $N \cup U$, $k_v$is the degree of node *v*. At each step, the node with lowest *outwardness* is moved into *C*. Chen et al. proposed a metric of community connection density in [13]: $L = L_{in}/L_{ex}$, where $L_{in}$is the average internal degree of nodes in *C*, L_{ex} is the average external degree of nodes in *B*. Those algorithms

mentioned above are all based on measuring connection among nodes around the given query vertex. Instead, Branting [14] concentrates on the spreading activation of the query vertex. Given a query vertex *v*, activation spreads along links from *v*. At each stage, the vertex with maximal activation joins in the community.

Our work is related to [14] in that we assume the given query vertex is an influencer that spreads influence along links. The difference from [14] is that we compute influence activation based on an influence diffusion model . As a result, local community detected by our algorithm is an influenced community whose initial influencer is the query vertex, which is the reason we call it local influenced community.

3 Multiple Spread Model (MSM)

In multiple spread model, an active node *v* has many chances to activate its inactive neighbors. Suppose an active node *v* has *T* chances to activate each neighbor. At the first time step, *v* activates its inactive neighbor *w* with probability $p_{v,w}(1)$. If it fails, it will continue to activate *w* with probability $p_{v,w}(t)$ at time step *t,* as long as *t* is not larger than *T*. If *w* has many active neighbor nodes, they attempt to activate *w* in arbitrary order. In MSM, inactive nodes can become active, but not vice versa.

We assume that the probability $p_{v,w}(t)$ at time step *t* is a function of time step *t*. The value of $p_{v,w}(t)$ decreases as *t* becomes larger. Given an active node *v* and its inactive neighbor *w* along with initial activation probability $p_{v,w}$, the probability of activating *w* successfully is

$$g(t, p_{v,w}) = \begin{cases} (1-p_{v,w})(1-p_{v,w}(2)) \dots (1-p_{v,w}(\mathrm{t}-1))p_{v,w}(t) \,, t = 2, \dots, T \\ p_{v,w} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad , \;\; t = 1 \end{cases} \tag{1}$$

$g(t, p_{v,w})$ is different from geometric distribution in that at each time step probability keeps changing. A simple expression of $p_{v,w}(t)$ is $p_{v,w}^t$. In this case, the activation probability is

$$g(t, p_{v,w}) = \begin{cases} (1-p_{v,w})(1-p_{v,w}^2) \dots (1-p_{v,w}^{t-1})p_{v,w}^t \;\; , \;\; t = 2, \dots, T \\ p_{v,w} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad , \;\; t = 1 \end{cases} \tag{2}$$

Obviously, ICM is a special case of MSM, when *T* is set 1.Comparatively, in ICM, an active node *v* has an only chance to activate its inactive neighbors. This constraint does not conform to the actual situation. For example, a person buys a product due to his many friends' influence. It does not mean that he buy it as soon as his friends buy it. The most probable case is that he accepts the product gradually due to continuous influence from his friends. Meanwhile, the influence is weakened along time and after some time the influence does not work any longer.

4 Influence Maximizing

According to multiple spread model, given an initial active node set *A* of size *k*, the diffusion process unfolds along with time steps. If a node *v* is activated in time step t_0,

it activates each of its inactive neighbors w with probability $p_{v,w}$ at time step t_0+1. If it fails, it still has chances at time step (t_0+t) (t=2, 3 …). The success probability in time step (t_0+t) is $p_{v,w}(t)$. Following is the definition of influence maximizing problem.

Definition 1. Influence Maximizing Problem: Assuming that diffusion process unfolds according to multiple spread model, given set size k, how to choose the initial set A with size k to be active, such that the number of active nodes $\pi(A)$ is largest at the end of diffusion?

In this section, we prove that influence maximization problem with MSM is monotone and submodular.

4.1 Approximation Guarantees

Theorem 1. The influence maximization problem is NP-hard for MSM.

Proof. Consider the NP-complete Set cover problem, given m subsets $\{S_1, S_2, \ldots, S_m\}$ of the universal set $\{u_1, u_2, \ldots, u_n\}$, we wonder that whether there exist k sets whose union is the universal set. We show that the set cover problem can be considered as a special case of influence maximization problem for MSM. We define a corresponding bipartite graph with $m+n$ nodes. Each subset or element from the universal set corresponds to a node. Whenever element u_i is in subset s_j, there is a directed edge from node j to node i in the bipartite graph with activation probability $p_{j,i}=1$. The set cover problem is equivalent to deciding if there is a set A of k nodes in the bipartite graph with $\pi(A) \geq n + k$. If there is any set A with $\pi(A) \geq n + k$, the set cover problem must be solvable.

Theorem 2. For an arbitrary instance of MSM, the influence result function $\pi(\cdot)$ is monotone and submodular.

Given a set function f on vertices of graph G=(V,E), f: $2^V \rightarrow R$, function f is monotone if $f(S) \leq f(D)$ for any set $S \subseteq D$. Function f is submodular if $f(S \cup \{v\}) - f(S) \geq f(D \cup \{v\}) - f(D)$ for any set $S \subseteq D$.

Proof of Theorem 4.2. The monotone property is obvious. Adding a node into an active seed set S can not decrease the resulting influence number.

Proof for submodular: In MSM, an active node v has many times to activate its inactive neighbor w. The probability of successful activation in the time step t is given by equation 1, which is

$$g(t, p_{v,w}) = \begin{cases} (1 - p_{v,w})(1 - p_{v,w}(2)) \ldots (1 - p_{v,w}(\mathrm{t} - 1))p_{v,w}(t), t = 2, \ldots, T \\ p_{v,w}, \quad t = 1 \end{cases}.$$

This is equivalent to the case that there are at most T coins between v and w. The activation process corresponds to flipping coins. We sequence those coins from 1 to T. The bias of the t-th coin is $g(t, p_{v,w})$.

We can consider the diffusion process as flipping T coins for each edge (v,w) sequentially. As long as there is at least one coin that succeeds, we label the corresponding edge as live. Since pre-flipping coins indicates the same result to the

moment that v becomes active, we pre-flip all coins. So given a pre-flipping result and an initial active set A, the number of activation is definite, which means that $\pi(A)$ is a deterministic quantity. Only if there is a path consisting of live edges from A to node v($v \in V\text{-}A$), the state of node v is active at the end of diffusion process. The subsequent proof part is equivalent to [4]. So the submodular property holds for influence maximization problem based on MSM.

4.2 Greedy Algorithm

Since the influencing maximization problem for MSM is monotone and submodular, the greedy algorithm provides (1-1/e) approximation for optimal solution [15]. The greedy algorithm shows in figure 1. The algorithm starts from an empty set (step 1), each time adding the node which provides the maximum marginal gain (from step 2 to step 11). The function *influence(A)* in step 6 decides activation number of A. We use Monte Carlo simulation to estimate marginal gain of each candidate node (from step 5 to step 8). Since the diffusion process is stochastic, we simulate R times to compute the marginal gain for each vertex $v \in V\text{-}A$. For each edge there are T chances to decide the live state of the edge. The complexity of algorithm 1 is $O(kmnRT)$. So the overall computation cost is very large. We can control parameter T to trade the computation cost for activation number.

Although there is theoretical guarantee with greedy algorithm, the computation costs much. Moreover, greedy algorithm is not scalable as well. It even takes several days to run on large data sets. Note that the most costing part of greedy algorithm is Monte Carlo simulations. As a result, we target decreasing the total number of Monte Carlo simulations. So we consider some heuristic methods, such as degree centrality, or splitting a network into communities and applying greedy algorithm in each community.

Algorithm 1: Greedy algorithm

Input: network $G=(V, E), p, T, k,$ where p is the initial probability for each edge, T is the total number of time steps, k is the seed number.

Output: seed set A consisting of k nodes, where $\pi(A)$ is the largest among all subsets with size k.

1: Initialize $A = \phi$
2: for i=1 to k do
3: for each vertex $v \in V\text{-}A$ do
4: s_v=0
5: for j=1 to R do
6: s_v+=*influence*($A \cup \{v\}$)- *influence*(A)
7: end for
8: $s_v=s_v/R$
9: end for
10: $A=A \cup \{argmax_{v \in V-A}\{s_v\}\}$
11: end for

Fig. 1. Greedy algorithm

4.3 Greedy Algorithm Based on Community Detection

Given an influential node, its influence spreads along paths to other nodes, whereas the influence probability decreases gradually. In this case, the influence of the node can not reach nodes far away from it. Meanwhile, seeds should scatter across the whole network, rather than in the same community.

Our heuristic method consists of two phases. In the first phase, we use kernel k-means algorithm [16] to partition graph nodes into k clusters, with the parameter k equivalent to the number of influential nodes. In the second phase, we apply greedy algorithm to each cluster and find an influencer in each cluster.

k-means method is one of the most popular clustering methods. Given a set of objects o_1, $o_2 \ldots o_n$, k-means clusters the objects into k parts $C_1, C_2, \ldots C_k$, while minimizing the objective function $\mathrm{D}\left(\{C_i\}_{i=1}^{k}\right) = \sum_{i=1}^{k} \sum_{n_j \in C_i} ||n_j - m_i||^2$, where $m_i = \left(\sum_{n_j \in C_i} n_j\right) / |C_i|$. There are two main drawbacks of k-means method. One is that it is infeasible to nonlinearly separable situation in input space. The other one is that k-means usually produce local optimal outcomes. Several approaches have emerged to tackle both drawbacks, among which kernel method and local search are used to handle the two drawbacks, respectively. Before clustering, kernel method map data objects to a higher-dimensional space using a nonlinear function. After then, the data points in higher-dimensional space are linearly separable for k-means. We apply kernel k-means method to partition networks into communities and optimize results with local search strategy.

Since the first phase of clustering costs little time and is not influenced so much by the number of clusters, the whole computation cost depends mostly on greedy algorithm. Since each cluster is smaller than original graph, total running time would reduce a lot, even though there are k clusters to exploit.

5 Local Influenced Community Detection

Most community detection algorithms leverage global property which sometimes is impossible to get when the network size is large enough or part of network is missing. Local community detection aims at resolving this problem. Given a query vertex, the task is to find out community spreading from the vertex. Common framework of existing relevant works [10-14] is to define a metric and at each step to add a vertex that optimize the metric. In this work, we assume that influence spreads from an influential vertex to other vertexes and define local influenced community as follows:

Given a query vertex v and an influencing spread model M, local influenced community of v consists of nodes influenced directly or indirectly by v based on M, as well as corresponding edges.

Local influenced community detection has concrete application. For example, in viral marketing, finding out community influenced by an influencer can help discovering common interest among influenced people or review tendency about a product, which is useful to adjust marketing strategy or to improve the product. In politics, we can detect what topics are talking about a politician in his local influenced community, as well as common profiles of people that are influenced by the

politician. In biology, local influenced community is helpful to examine how a functional cell influences others.

Given an influencer *v*, *v* spreads its influence along edges in graph *G*. We propose an algorithm called LICD to detect local influenced community of a particular node. Figure 2 exhibits details. At each time step *t*, an active node *u* tries to influence each neighbor *w* with probability $g(t, p_{u,w})$ (step 4), where $p_{u,w}$ is the initial probability along with edge *(u, w)*. If *u* succeeds at time *t*, *w* becomes influenced at time step *t*+1 and joins local influenced community C (step 5, 6, 7). Then *w* tries to influence its non-influenced neighbors and the process goes on likely. When there is no vertex that can be influenced or number of members in community already meets demands, the process terminates. We determine whether a vertex use up its chances by the flag attribute "isChecked" (step 1, 2, 7, 9).

Algorithm 2: local influenced community detection algorithm (LICD)
Input: network G, a query vertex v, probability p with each edge.
Output: local influenced community C.
1. C ={v},v.isChecked=false,
2. for each $u \in C$ and *u*.isChecked=false
3. for each *w*∈*neighbor(u)*- C
4. u spreads influence to *w* with probability
$$g(t, p_{u,w}) = \begin{cases} (1-p_{u,w})(1-p_{u,w}(2)) \dots (1-p_{u,w}(t-1))p_{u,w}(t), & t = 2, \dots, T \\ p_{u,w}, & t = 1 \end{cases}$$
5. If u succeeds
6. $C = C \cup \{w\}$
7. *w*.isChecked=false
8. end for
9. *u*.isChecked=true
10. end for
11. Return C

Fig. 2. LICD algorithm

LICD contains two level loops. Supposed k is the size of *C* when LICD terminates. Then the running time of LICD is $O(k^2dT)$, where d is the mean degree of *G*. when graph *G* is sparse and k>>T, the time complexity is roughly $O(k^2)$. Note that the influencing probability at time *t* is

$$\begin{aligned} g(t,p) &= (1-p)(1-p^2)\dots(1-p^{t-1})p^t \\ &= \frac{g(t-1,p)(1-p^{t-1})p^t}{p^{t-1}} \\ &= g(t-1,p)(1-p^{t-1})p \end{aligned}$$

So we only need reserve $g(t-1,p)$ for computing $g(t,p)$.

6 Experiments

In this section, we evaluate the efficiency and effectiveness of the proposed approaches. All the experiments were conducted on a PC with Intel® Core™ 2 Duo processor (3.0GHz) and 3G memory.

6.1 Data Sets

We use three data sets: a collaboration network, a citation network and a wiki-vote network. The collaboration network is collaboration network of Arxiv general Relativity, which is undirected. Nodes represent authors and edges represent collaboration. It contains 5242 nodes and 14490 edges. The citation network contains 27770 nodes and 352807 edges. Nodes represent papers and edges represent citation relationship among papers. The Wiki-vote network contains 7115 nodes and 103689 edges. Nodes represent Wikipedia users and an edge from node *i* to node *j* represents that user *i* voted on user *j*. Both the citation network and wiki-vote network are directed. We use the collaboration network for influence maximizing problem, the citation network and the wiki-vote network for local influenced community problem.

6.2 Influence Maximizing

The data set used here is the collaboration network. We compare greedy algorithm between MSM and ICM, as well as with high degree heuristic method and random method. For simplicity, probability along with each edge is uniformly set 10%. We set parameter T as 1, 5, and 10, respectively. When *T* is set 1, MSM degrades to ICM. In all experiments, we set parameter R of Monte Carlo simulation as 10000.

Figure 3, 4, 5 shows effectiveness of MSM. In figure 3 and figure 4, we compare MSM with ICM, degree method, random method with different *T* values. Degree method means nodes are ranked according to degree, and top-k nodes are chosen as seeds. Then we apply greedy algorithm based on MSM with those seeds. In random method, we choose k seeds randomly. Both figures show that MSM performs best. When k is 1, degree method outperforms ICM. When k becomes larger, the curve of degree method rises more slowly than that of ICM. The probable reason is that high degree nodes tend to group closely and their influencing area is partly overlapped. The random method performs least.

Figure 5 lists MSM performance with different *T* values. It shows that the larger *T* is, the more influence size MSM performs. On the other hand, performance of MSM improves more when *T* becomes from 1 to 5 than when *T* becomes from 5 to 10. The reason is that when *T* becomes larger, activation probability becomes smaller.

We use the influence size of all seeds as the performance of one algorithm. Figure 6 compares the performance of naïve greedy algorithm and greedy with community detection. We apply the proposed two-phase method (Section 4.3) to select one seed from each community. Then we apply MSM with those k seeds on the whole network

to evaluate the influence size of the two-phase method. In this experiment, we set k as 5. Figure 6 shows that the performance of the two-phase method is approximate to the naïve greedy algorithm with different *T* value. Since the time of community detection is trivial when compared with greedy algorithm, we take no account of it.

Figure 7 lists computation time of the two-phase method and naïve greedy algorithm. Parameters *T* and k are the same to those in figure 6. In figure 7, time cost of the two-phase method reduces dramatically. Obviously, the two-phase method is a better alternative compared to naïve greedy algorithm.

We also check out the distribution of k seeds selected by naïve greedy algorithm. First of all we partition the network into communities by kernel k-means method. And then we find that no matter *T* is 5 or 10, there exists more than one seed in the same community, which means that seeds selected by naïve greedy algorithm are not scattered enough. Intuitively, if more than one seed is clustered into the same community, their influence scope overlaps, which weakens the whole influence effect.

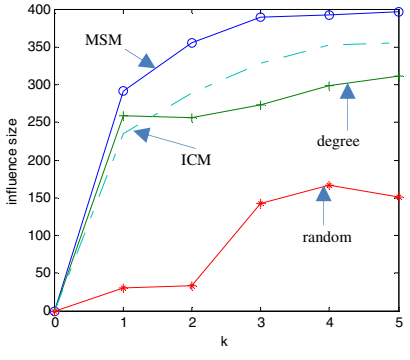


Fig. 3. Objective functions with T=10

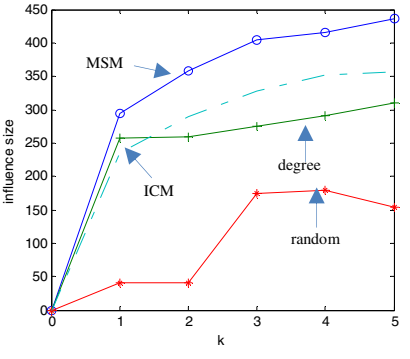


Fig. 4. Objective functions with T=5

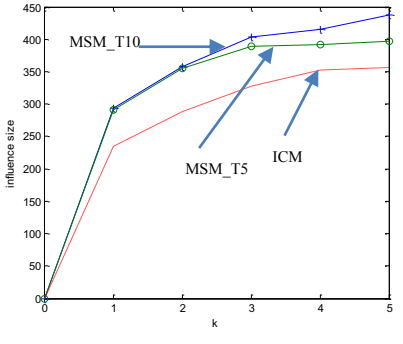


Fig. 5. Greedy algorithm with different *T*

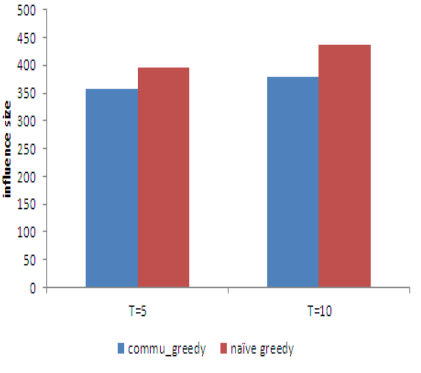


Fig. 6. Combine greedy with community: influence size

6.3 Local Influenced Community

We use the citation network and the wiki-vote network for LICD algorithm. In the citation network, an edge from node *i* to node *j* means that node *i* influences node *j* in research topics. The average clustering coefficient is 0.32951. In the wiki-vote network, an edge from node *i* to node *j* shows node *i* voted on node *j*, which means node *j* influences node *i*. In preprocessing, we reverse direction of each edge in the wiki-vote network. The average clustering coefficient of wiki-vote network is 0.20891. We randomly choose a node in each network as the query node. The baseline is original local network around the query node, with radius 10. If there is a path from the query node to a node, we assume the node is influenced by the query node. As a result, nodes inside 10-hops from the query node are included in baseline community. We run greedy algorithm for 15 times with ICM and MSM, respectively. Whenever a node is activated, it joins local influenced community. We compare the average size of local influenced community with original network and compute the ratio.

Table 1 lists the results. The results show that the MSM results approximate more with baseline than ICM results. Note that the ratios in the wiki-vote network are larger than those in the citation network. The reason lies on that wiki-vote network has smaller average clustering coefficient than the citation network. Influence is not so dispersed when average clustering coefficient is small.

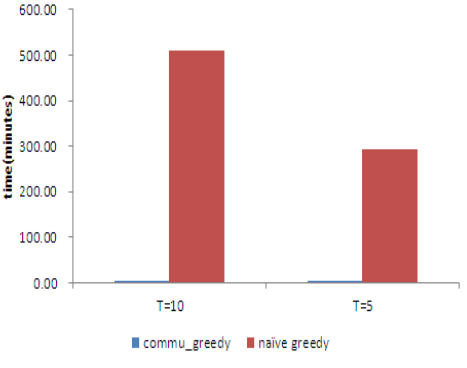


Fig. 7. Combine greedy with community: time

Table 1. Local Influenced Community size ratio

influence diffusion model	citation network	wiki-vote network
ICM	0.79%	17.44%
MSM_T_5	5.91%	17.92%
MSM_T_10	**6.23%**	**19.91%**

7 Conclusion and Future Work

In this paper, we propose an influence diffusion model, called multiple spread model, in which an active node has many chances to activate its neighbors. We prove that influence maximizing problem with multiple spread model is submodular and monotone. This guarantees that greedy algorithm approximates (1-1/e) of optimal solution. Since computation time costs much in naïve greedy algorithm, we propose a

two-phase method that combines community detection and naïve greedy algorithm. Experiments show the performance of the method is approximate to naïve greedy algorithm while its computation time costs much less. In order to evaluate influence of a particular node, we also define local influenced community and propose an algorithm named LICD to detect local influenced community. We applied MSM and ICM to detect local influenced community. Experiment results show that MSM performs better in size of local influenced community. In short, MSM is more corresponding to influence diffusion in realistic network than ICM.

In our experiments, for simplicity, we set probabilities and activation time step uniformly. However, in some situations, importance is different among nodes or edges, especially in weighted networks. In such case, each node and each edge should be treated differently, which demands distinguishing probabilities and activation time step. This will complicate the situation and make problems more challenging. Our first future work is to study problems with distinguishing probabilities and activation time step.

The second future work targets at studying alternative approaches, since greedy algorithm consumes large computation. For example, we can exploit local network structure and adopt some heuristic methods.

References

1. Goldenberg, B.L.J., Muller, E.: Talk of the network: A complex systems look at the underlying process of word-of-mouth. Marketing Letters 12(3), 211–223 (2001)
2. Fortunato, S.: Community detection in graphs. Physics Reports 486, 75–174 (2010)
3. Newman, M.E.J.: Fast algorithm for detecting community structure in networks. Physical Review (2004)
4. David Kempe, J.K., Tardos, E.: Maximizing the Spread of Influence through a Social Network. In: KDD, Washington, DC, USA (2003)
5. Kimura, M., Saito, K.: Tractable Models for Information Diffusion in Social Networks. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS (LNAI), vol. 4213, pp. 259–271. Springer, Heidelberg (2006)
6. Jure Leskovec, A.K., Guestrin, C., Faloutsos, C., VanBriesen, J., Glance, N.: Cost-effective Outbreak Detection in Networks. In: KDD (2007)
7. Wei Chen, Y.W., Yang, S.: Efficient Influence Maximization in Social Networks. In: KDD, Paris, France (2009)
8. Narayanam, Y.N.R.: A shapley value based approach to discover influential nodes in social networks. IEEE Transactions on Automation Science and Engineering (2010)
9. Wei Chen, A.C., Cummings, R., Ke, T., Liu, Z., Rincon, D., Sun, X., Wang, Y., Wei, W., Yuan, Y.: Influence Maximization in Social Networks When Negative Opinions May Emerge and Propagate. Microsoft Research Technical Report (2010)
10. Clauset, A.: Finding local community structure in networks. Physical Review 72(2), 26132 (2005)
11. Luo, J.W.F., Promislow, E.: Exploring local community structures in large networks. Web Intelligence and Agent System 6(4), 387–400 (2006)
12. Bagrow, J.: Evaluating local community methods in networks. Journal Statistical Mechanics 2008(05), 5001 (2008)

13. Zaïane, O., Chen, J., Goebel, R.: Local community identification in social networks. In: Proceedings of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM), Athens, Greece (2009)
14. Branting, L.K.: Incremental detection of local community structure. In: International Conference on Advances in Social Network Analysis and Mining, Los Alamitos, CA, USA (2010)
15. Nemhauser, L.W.G., Fisher, M.: An analysis of the approximations for maximizing submodular set functions. Mathematical Programming 14, 265–294 (1978)
16. Inderjit, Y.G., Dhillon, S., Kulis, B.: Kernel kmeans, Spectral Clustering and Normalized Cuts. In: KDD (2004)
17. Lancichinetti, A., Fortunato, S.: Community detection algorithms: a comparative analysis. Physcal. Review. E 80, 56117 (2009)
18. Ma, H., Yang, H., Lyu, M.R., King, I.: Mining social networks using heat diffusion processes for marketing candidates selection. In: Intelligent Hypertext (2008)

Interactive Predicate Suggestion for Keyword Search on RDF Graphs

Mengxia Jiang[1,2], Yueguo Chen[2], Jinchuan Chen[2], and Xiaoyong Du[1,2]

[1] School of Information, Renmin University of China, Beijing, China
[2] Key Laboratory of Data Engineering and Knowledge Engineering, MOE, China
{jmx,chenyueguo,jcchen,duyong}@ruc.edu.cn

Abstract. With the rapid growth of RDF data set, searching RDF data has recently received much attention. So far, structural languages such as SPARQL support to search RDF data efficiently and accurately. Unfortunately it requires users to have the prior knowledge of the underlying schema and query syntax of RDF data set. On the other hand, keyword search over graphs outperforms SPARQL queries in terms of usability. However, the predicate information is ignored in keyword queries, which results in the huge searching space and generates ambiguous interpretation of queries. In this paper, we design an interactive process of keyword search, which allows users to reduce the ambiguity of keywords by selecting some predicates to constrain the semantics of query keywords. We propose an efficient and effective algorithm to online discover a small number of predicates for users to choose. Experiments on the YAGO data set demonstrate the effectiveness and efficiency of our method.

1 Introduction

There is an emerging trend to apply keyword search on the graph-shaped data extracted from the Web. As a W3C-endorsed data model that builds on graphs, RDF is gaining prevalence for resource description and large-scale RDF collections such as DBpedia [4] or YAGO [19], which arouses intensive studies for the efficient storage and querying of RDF data [1,17]. Although SPARQL is a powerful language that search RDF data like SQL on relational database, it only suits for professional users because: firstly, the complex syntax structure of SPARQL [17] prevents ordinary users to learn and handle it easily; secondly, users must know at least part of the RDF schema information (like table name or column name in SQL query) as a prior knowledge of writing SPARQL query, which is usually nontrivial.

From another perspective, an RDF collection can be modelled as a graph, where nodes are subjects/objects, and edge labels are predicates. Keyword search on graph data has attracted much attention for its easy portal to assist users to access the desired information. As introduced in [5], given a directed graph G in which each node is associated with a plain text label, keyword search on graphs concerns with querying G by a set of keywords. The expected answer is a ranked list of substructures which contain the query keywords in their leaf nodes.

J. Tang et al. (Eds.): ADMA 2011, Part II, LNAI 7121, pp. 96–109, 2011.

However, RDF is not merely a directed labeled-graph, which makes keyword search on RDF graphs somehow different. In particular, there are two main differences between RDF graphs and common labeled-graphs:

- The edges in RDF graphs are labeled. An edge label (i.e., a predicate) usually describes the relationship between the corresponding two nodes. Generally, predicates are predefined and specified in certain format. They typically do not support keyword search directly.
- The literal nodes in traditional labeled graphs are typically short labels. However, many RDF labels are long texts that may contain dozens of words. As a result, a large number of subgraphs potentially contain query keywords, which leads to the ambiguity of keyword search.

Predicates can help us to reduce the disambiguation of keywords. For example, "Apple" with the predicate "hasCEO" obviously means a company. The authors of [20] and [6] have observed this and they proposed that predicates should be taken into accounts for keyword search over RDF graphs. However, it is not clear how the predicates are involved in keyword search process. To the best of our knowledge, there are few studies on predicate suggestion for keyword search over RDF graphs. Traditional methods of keyword search on graphs [12,9,13] mainly focus on the labeling literals associated with nodes, without edge information (predicates). The motivation of this paper is, in real scenarios, users who present the keyword query generally have an expectation on the semantics of the keywords. Thus, it will be helpful if a "smart" search engine can suggest a short ranked list of predicates to interpret the user's intention, so that even the user does not know any knowledge about the underlying schema of an RDF data set, he can find the desired information easily by inputting some keywords and associating some predicates to constrain the semantics of keywords.

In this paper, we propose an interactive predicate suggestion for keyword search on RDF graphs. We use an example to illustrate the main idea. Suppose a user inputs a keyword query "Ulm Nobel_Prize", our system will suggest a ranked list of predicates for each keyword to capture the user's intention at moment m_0, e.g., "isCityOf" and "bornIn" for "Ulm", "isCalled" and "hasWon" for "Nobel_Prize". If the user selects "bornIn" for "Ulm", the system will be aware of user's intention by suggesting "hasWon" for "Nobel_Prize" to help user find a person who was born in Ulm and has won Nobel prize. Once the predicate selection is completed, our system will rank all the possible results that contain the issued keywords and the associated predicates.

As a new problem, searching RDF data with predicate suggestion is a challenging work. As an online process, predicate suggestion should be able to efficiently capture users' intention as accurate as possible, even facing RDF graphs with billions of triples. In this paper, we exploit both the semantics and structure of RDF graphs for predicate suggestion. We summarize the main contributions: (1) we shed light on easing keyword search on RDF graphs by interactive predicate suggestion; (2) we propose the prioritized propagation and aggregation (PPA) algorithm to capture user's intention with enhanced performance; (3) we test our method on 17.6 millions YAGO triples to demonstrate the effectiveness and efficiency.

In the rest, Sections 2 and Section 3 give formal problem statement and the framework of our solution. In Section 4, we elaborate the MPA and PPA algorithms for effective and efficient predicate suggestion. Section 5 provides experimental study, followed by the related work in Section 6 and the conclusion in Section 7.

2 Problem Statement

Interactive predicate suggestion for keyword search over RDF data is based on the following two intuitions:

Intuition 1. *Generally, users who input the keywords know the exact semantics of these keywords. However, they do not know how the semantics is presented in RDF graphs because they often do not know the predicates used in RDF data sets. Simply providing more keywords for clarifying the query intention may cause false negatives of matching subgraphs.*

Intuition 2. *The desired results are substructures with a very small radius. In other words, all the querying keywords must appear in literal nodes close to each other in terms of the graph distance of RDF data sets. A small radius guarantees the strong semantic correlations among the mentioned nodes of querying keywords.*

Intuition 1 provides a supporting evidence for interactive predicate suggestion. The query intention of users can be interpreted by keywords associated with predicates, which are obtained from user feedbacks. Intuition 2 reveals the substructures with small radius are preferred, which forms the theoretical basis of the prioritized propagation and aggregation (PPA) approach we propose in the predicate suggestion process.

RDF Graph Formulation. An RDF statement is a subject-predicate-object triple and represented by (S, P, O), which corresponds to an edge from a node S to a node O with edge type P in the RDF graph. An RDF dataset can be abstracted as a graph G. As a general discrimination, vertices in G are partitioned into two disjoint sets: V_L (representing literal nodes) and V_E (entities, typically represented as URIs). A literal node must link to an entity node. For the RDF graph example shown in Fig. 3(a), $p1$ is an entity node, "2010" and "*Paper*" are literal nodes. Note that keywords can only appeared in literal nodes because entity nodes are only URIs, which typically are not meaningful in semantics. We formalize an RDF data graph as follows.

Definition 1 (RDF data graph). *An RDF graph G can be represented as a tuple $(V, L, \phi_V, E, P, \phi_E)$, where*

- *V is a set of vertices and $V = V_L \cup V_E$;*
- *L is a set of node labels;*
- *ϕ_V is a mapping function which assigns each node in V_L by a label in L;*
- *E is a set of edges with form $e(v_1, v_2)$, where $v_1, v_2 \in V$;*
- *P is a set of predicates;*
- *ϕ_E is a mapping function which assigns each edge in E by a predicate in P.*

Input/Output. The interactive predicate suggestion process is partitioned into a number of moments. The initial input at moment m_0 is a set of keywords denoted as $Q_0 = \{K_1, K_2, \cdots, K_n\}$ if the keyword size $|Q| = n$. At each moment, the user will select a predicate from the suggestion list for an unpaired keyword. As time goes by, more and more keywords will be paired with predicates due to user's selection. For example, suppose Q_0={mining,ICDM}. At moment m_1 the user selects "confName" for "ICDM" to identity "ICDM" is a conference name, we get Q_1={mining,(ICDM,confName)} as the input of moment m_1. We formalize the input at any moment m_t as follows:

Definition 2. *Supposing $Q_0 = \{K_1, \cdots, K_n\}$, the input of interactive predicate suggestion at moment m_t is formalized as $Q_t = (Q_0, P_t, \phi_t)$, where*

- *P_t is a list of associated predicates and the size $|P_t| \leq n$;*
- *ϕ_t is a mapping function to associate keywords with corresponding predicates, and*
  - *$\phi_t(K_i) = p \in P_t$ means K_i is associated with a predicate p;*
  - *$\phi_t(K_i) = *$ means K_i is not associated with any predicates.*

The output at moment m_t is a suggestion list of predicates for each un-associated keyword, and we use $S_t(K_i)$ to denote the output for the keyword K_i. For example, *author* and *title* are suggested to keyword *mining* at moment m_1. S_t will be presented to users for predicate selection. It is important to note that at each moment, predicates are finely evaluated and quickly updated to capture the user's intention for un-associated keywords. This interactive suggestion process is the main focus of this paper. It is up to the user to abort the interactive selection phase at any moment, so after the completion of predicate suggestion, it is not necessary for every keyword to be associated with a predicate. In the extreme case, users may reject any predicate suggestion and submit Q_0 as the final input. Once the users finish the process predicate selection at some moment m_t, Q_t will be sent to query execution engine and a list of ranked substructures that fit Q_t will be ranked and returned.

Problem. In interactive predicate suggestion, supposing the input is Q_t at moment m_t, we are concerned with the problem of efficiently updating top-ranked predicates for each keyword that is un-associated with predicates in Q_t to capture user's intention. Let's present this problem in a formal way.

Problem 1. At moment m_t, given the input Q_t, suggest the top-k ranked predicate list $S_t(K_i)$ for any keyword K_i if $\phi_t(K_i) = *$.

Finally, major notations used in this paper are shown in Fig. 1.

3 The Framework

3.1 The Structures of Search Results

In traditional approaches ([5,9,12,10,14]) of keyword search on graphs, the answers are typically tree-shaped substructures (in particular, Steiner trees [5])

$\phi_E(e)$	the predicate on edge $e(v_1, v_2)$
$\phi_V(v)$	the label on node v
$V(K_i)$	keyword nodes that matches K_i
T_{K_i}	keyword node that matches K_i in subtree T
$M(T_{K_i}, K_i)$	matching degree between T_{K_i} and K_i
$\phi_t(K_i)$	The predicate associated with K_i at m_t, or * if not associated

Fig. 1. Notations

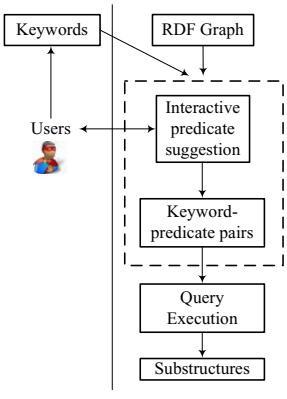


Fig. 2. The framework

where the keyword nodes correspond to leaves. With directed paths from the root to leaves, subtree patterns are meaningful in domains such as XML and relational DB. However, in RDF graphs, we claim the restriction for "directed path" is not necessary. The main reason is, edge directions are generally invertible in RDF statements, e.g., (Alice, eat, Apple) can also be expressed as (Apple,isEatenBy,Alice), making the edge direction in RDF graphs less meaningful. Thus, we consider the resulting substructures as undirected subtrees. In particular, two kinds of nodes are distinguished in an undirected subtrees:

1) **Keyword nodes**, which contain query keywords;
2) **Spanning nodes**, which connect all keyword nodes together.

Recalling Definition 1, keyword nodes are literal nodes, and spanning nodes can only be entities (represented using URIs). The root node is a special spanning node and generally considered as the answer node for the keyword query. According to Intuition 2, the resulting subtree should be compact with small radius. We therefore define the paths from the root to leaves as the shortest paths of the nodes. This guarantees that the resulting subtree to be a minimum spanning tree (MST). If a spanning node is removed, the subtree will be disconnected. Two examples of answering subtrees for query "ICDM mining" are shown in Fig. 3(c).

3.2 Framework of Interactive Predicate Suggestion

Next we explain the framework process of interactive predicate suggestion for keyword search on RDF graphs which is showed in Fig. 2. The initial input of users at moment m_0 is a set of keywords. Immediately after that, the predicate suggestion module will be invoked. For each keyword, a ranked list of predicates will be suggested to interpret the semantics of this keyword. Then at moment m_1, users can select a predicate for a keyword that mostly captures his intention for that keyword. This selection will be reflected in the updating of predicate lists

for the remaining un-associated keywords by the predicate suggestion module. Basically, the predicate suggestion module will be invoked for each moment. This interactive process goes on until all keywords are associated with predicates, or users decide to finish this process at any moment. Finally, keywords and selected predicates will be submitted to the query execution engine. The output of the execution engine is a ranked list of connected subtree which contains all query terms in its node labels.

3.3 Relevance Evaluation

Now we discuss the relevance heuristics for the resulting substructure. In recent works, Many heuristics [8,5,13,10] have been proposed to evaluate the relevance scores of matching subtrees for keyword search over graphs. Based on them, we design three general components to evaluate the relevance of answers in RDF graphs: (1) the similarity between keywords and the literal of their matched keyword nodes; (2) the importance of root node T_r and (3) the distance from the root to keyword nodes. To integrate these components together, we use the idea of match propagation and aggregation (will be explained in Section 4) and consider the score of a subtree as the sum of matching degree propagating from leaf nodes to the root, decayed by a factor $0 < C < 1$ and weighted by the importance of root node. That is,

$$Score(T) = I_{T_r} \cdot \sum_{K_i \in K} M(T_{K_i}, K_i) C^{T_{p_i}} \tag{1}$$

where T_{K_i} is the keyword node that matches K_i in T and $M(T_{K_i}, K_i)$ is the similarity of keyword match, quantified by popular similarity measures such as Jaccard Coefficient [15]. T_{p_i} is the distance from T_r to T_{K_i}. I_{T_r} reflects the importance of root T_r in the whole graph, which can be evaluated by popular authority ranking methods like PageRank [16].

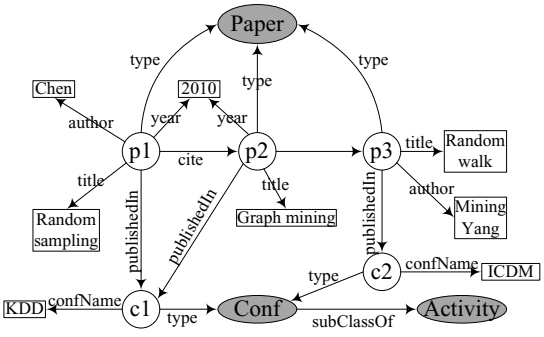
4 Prioritized Propagation and Aggregation

In this section, we first propose the *match propagation and aggregation* (MPA) algorithm, which provides a basic methodology for predicate suggestion. To make MPA more practical in real applications, we further propose an alternative of MPA, called the *prioritized propagation and aggregation* (PPA) algorithm, to leverage the performance of MPA.

4.1 MPA

To illustrate the main idea, we start from an example.

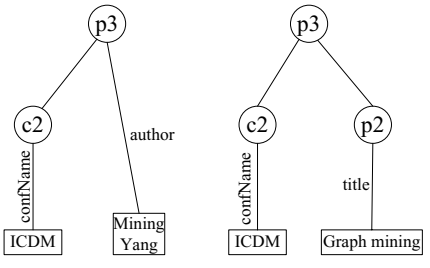
Example 1. In Fig. 3(a), suppose the query is "mining random" and at moment m_1, 'random" is associated with predicate "title". How can we suggest predicates for keyword "mining"? Notice that "mining" may be a part of an author name or title name.



(a)

2010	1
Activity	2
Chen	3
Conf	4
Graph mining	5
ICDM	6
KDD	7
Paper	8
Random walk	9
Stream mining	10
Mining Yang	11
c1	12
c2	13
p1	14
p2	15
p3	16

(b)



(c)

	...	12	13	14	15	16
1	...	2		1	1	
2	...	2	2			
3	...	2		1	2	
4	...	1	1	2	2	2
5	...	2		2	1	2
6	...		1			2
7	...	1		2	2	
8	...	2	2	1	1	1
9	...		2		2	1
10	...	2		1	2	
11	...		2		2	1
12	...		2	1	1	
13	...	2			2	1
14	...	1			1	2
15	...	1	2	1		1
16	...	2	1	2	1	

(d)

Fig. 3. (a) An example of RDF graph. (b) Sorted node numbering. (c) Two answering subtrees for query "ICDM mining". (d) A fragment of shortest distance matrix D.

Intuitively, the keyword matching similarity and the distance of nodes should be both considered in the predicate ranking. However, how to combine them adequately is a problem. In this paper, we rank predicates in a uniform framework called match propagation and aggregation. In simple words, we consider each keyword node has an authority since it matches the keyword, and this matching degree will propagate to other nodes via the selected predicate edges. A predicate of an un-associated keyword will be ranked by the matching traffic that goes through this predicate. In particular, during the predicate suggestion process, keyword nodes have two kinds of roles, as explained below.

Propagator: At moment m_t, if a keyword K_i is associated with predicate p, keyword nodes that match K_i are propagators and the matching degree can be propagated to other nodes via edge e with $\phi_E(e) = p$.

Aggregator: At moment m_t, suppose keyword K_j is not associated with any predicate, i.e., $\phi_t(K_j) = *$, keyword nodes that match K_j are aggregators and will receive the matching degrees from their neighbors.

Basically, at the node level, matching degree will flow from propagators to aggreagators. At the predicate level, the aggregated matching degree on aggregators

via predicate p will be taken as the basis to rank p. In Example 1, nodes matching "random" are propagators and nodes matching "mining" will receive the matching degree from propagators via edge type "title".

If K_j is un-associated and p is one of its predicate, now we compute the matching degree traffic of p and let $M_p(K_j)$ to denote it. Recalling that T_{p_j} is the shortest distance from the root to leaf node that matches K_j, we have

$$M_p(K_j) = \sum_{T'} \sum_{\phi_t(K_i) \neq *} M(T'_{K_i}, K_i) \cdot M(T'_{K_j}, K_j) \cdot C^{T'_{p_i} + T'_{p_j}} \tag{2}$$

where T' is the subtree T that satisfies T_{K_j} is connected by edge with predicate p. $\phi_t(K_i) \neq *$ actually indicates the propagators that are associated in Q_t.

Why $M_p(K_j)$ works to rank predicate p of keyword K_j? The reason is, $M_p(K_j)$ captures the matching degree traffic from all propagators to T_{K_j} through edges with predicate p. Higher $M_p(K_j)$ means that the matching degree of keywords is higher and the paths from the root to leaf nodes are shorter, which results in a subtree ranked higher by scoring function shown in Equation 1.

4.2 PPA

MPA is a kind of random walk algorithms [3], where a random surfer starts from the propagators and visits their neighboring entities recursively. If an entity is visited, it will become active and can influence its neighbors at the next step. Further, we use "frontier" to describe the set of entities that are activated at current step. For example, {p2,c2} is the frontier of keyword node "random walk" at 2 steps in Fig. 3(a). Recalling that the output subtrees are MSTs, the frontier $F(i)$ at step i is actually the nodes reached from propagators by shortest paths with exact i steps, and we denote it as $F(i) = \{x | D(v, x) = i\}$, where v is a propagator and D is the shortest distance matrix. A fragment of D with maximal distance $d_{\max} = 2$ is shown in Fig. 3(d).

However, MPA may face performance issues in real RDF graphs, since some nodes are of very high degree and the number of visited nodes increases exponentially during the flooding process. Also, if the amount of propagators are huge, the time and space cost will become unaffordable for the predicate suggestion process. $|F(i)|$ may easily go up to hundreds of thousands. Thus, it becomes necessary to prioritize the nodes in the same frontier and only propagate the nodes with higher authority. In detail, for a node v in $F(i)$, the propagators that reach v on the i-th step can be denoted as a set $S(v) = \{s | D(s, v) = i\}$. Thus, all nodes in the frontier $F(i)$ like v can be ranked by

$$PPA(v) = \sum_{s \in S(v)} M(T_{K_s}, K_s) \cdot C^i \tag{3}$$

After that, only nodes of top-k highest scores will be further propagated in $F(i)$. An important problem is how to decide the parameter k. The most simple way is setting k by a threshold, which is either a fixed number or a fixed ratio. However, the fixed number (e.g., top 100) cannot reflect the size of the frontier. The fixed ratio (e.g., top 30%) may exceeds the total time budget if the size

of frontier is extremely huge. Thus, we propose a more flexible but efficient strategy by combining them together using a piecewise linear function with factor (*min*,*max*,*ratio*), where *min* and *max* are the lower and upper bound of k and *ratio* decides the portion of k out of all candidates. Formally, let $|F(i)|$ be the size of the frontier, and we decide k by

$$k = \begin{cases} |F(i)|, \text{ if } |F(i)| \leq min \\ min, \text{ if } |F(i)| > min \text{ and } |F(i)| \cdot ratio < min \\ |F(i)| \cdot ratio, \text{ if } min \leq |F(i)| \cdot ratio < max \\ max, \text{ if } |F(i)| \cdot ratio > max \end{cases} \tag{4}$$

4.3 Interactive Predicate Suggestion

The steps of interactive predicate suggestion is summarized in Algorithm 1. Line 1-12 are the process of initialization, which involves mapping the keywords to nodes in RDF graph and finding entities within $d_{\max}$ steps based on prioritized propagation and aggregation. Line 13-18 are the first predicate suggestion to users at moment m_0. Since no keywords are associated with predicates at this moment, the suggestion of predicate p is basically a statistics of p occurring in subtrees. Line 19-27 are steps for predicate suggestion at moment m_t with $t \geq 1$. Here, some of keywords are associated with predicates and serve as propagators, whose matching degree will be propagated to other keywords that are un-associated. The predicates associated with aggregators will be ranked by the matching degree traffic flowing through them, as computed in Line 25.

For time and space complexity, supposing the keyword node set size is N and the average degree is $\frac{2|E|}{|V|}$ (for undirected graphs), the visited node size is $O(nN(\frac{2|E|}{|V|})^{d_{\max}})$ for MPA. MPA faces performance problems when N or edge degree are unusually huge. Thus, it is our recommendation to replace MPA by PPA, which limits the upper bound of the visited node size as $O(n \cdot \max)$.

5 Experimental Study

In this section, we instantiate the interactive predicate suggestion on YAGO dataset to evaluate the effectiveness and performance. Without loss of justice, our method is compared with other existing methods like traditional keyword search on graphs. All experiments are implemented by Java, and running on a computer with 1.86GHz CPU and 4GB main memory.

5.1 Data Set and Query Set

As a huge collection of RDF facts, YAGO(1) dataset[1] contains 17,631,348 statements, 8,979,923 unique subjects and 4,654,016 unique objects. We map all RDF statements into edges and finally builds an RDF graph with 10,796,073 nodes and 92 different kinds of edge types (predicates). To index these data and respond

[1] http://www.mpi-inf.mpg.de/yago-naga/yago/downloads.html

Algorithm 1. Interactive Predicate Suggestion

Input: RDF graph $G, C, d_{\max}$, query $\{K_1, \cdots, K_n\}$
Output: Suggestion lists S_t and the answer subtrees
`// Offline Part:`
for *each keyword* K_i **do**
  find keyword node set $V(K_i)$ that matches K_i in $V_A \cup V_C$;
  $D(K_i) = \emptyset$;
  for *each keyword node* $v \in V(K_i)$ **do**
    for *step* $i = 1 : d_{\max}$ **do**
      if $D(v, v') = i$ *and* $v' \in V_E$ **then** add v' into $D(K_i)$;
      only expand top k entities in $D(K_i)$ according to Equation 4;
$Root = D(K_1) \cap \cdots \cap D(K_n)$;
$TreeSet = \emptyset$;
for *each entity* $r \in Root$ **do**
  generate subtrees rooted by r and having leaf nodes in $V(K_1), \cdots, V(K_n)$;
  add these subtrees into $TreeSet$;
`// Online Part:`
set $t := 0$;
for *each keyword* $K_i \in \{K_1, \cdots, K_n\}$ **do**
  for *each predicate* p *associated with* K_i **do**
    $S_0(K_i, p) = \sum_T M(T_{K_i}, K_i) \cdot C^{T_{K_i}}$ where $T \in TreeSet$ and T_{K_i} is connected by edge with p;
  $S_0(K_i) =$ the ranked list of $S_0(K_i, p)$;
present S_0 to the user;
while *not interrupted by the user and* $t < n$ **do**
  $t := t + 1$;
  select a predicate for a keyword and build input Q_t;
  remove subtrees not consistent with Q_t from $TreeSet$;
  for *each unpaired keyword* $K_i \in Q_t$ **do**
    for *each predicate* p *associated with* K_i **do**
      $M_p(K_j) = \sum_{T'} \sum_{\phi_t(K_i) \neq *} M(T'_{K_i}, K_i) \cdot M(T'_{K_i}, K_i) \cdot C^{T'_{p_i} + T'_{p_j}}$ where $T' \in TreeSet$ and T'_{K_i} is connected by edge with p;
    $S_t(K_i) =$ the ranked list of $S_t(K_i, p)$;
  present S_t to the user;
rank subtrees that are consistent with Q_t in $TreeSet$ and return;

fast for user queries, we use Compressed Row Storage (CRS, [18]) to store them in a compact way, where a matrix is transformed into three vectors: non-zero values, column indices and row starting indices. Terms are obtained by splitting node labels by "_" or blank, and a total # of 3,430,600 terms are included.

Two distinctly different keyword query sets QS_1 and QS_2 are used: QS_1 contains popular keywords with average keyword node size $|V(K_i)| = 4772$ (example: "movie Italian") while QS_2 contains less popular keywords with average keyword node size $|V(K_i)| = 63$ (example: "Deborah Exterminator"). How do we get these query sets? Firstly, we randomly produce many queries. After getting these queries, we calculate two measures: the number of keyword nodes, and the number of neighbors for keywords in each query. With the information we filter unsatisfied queries and separate the remaining into two groups,that is QS_1 and QS_2. QS_3 is produced similarly. Each query in QS_3 contains 4 keywords. The average keyword node size of QS_3 is 5987, and average root size is 40.

5.2 Quality Evaluation

Because keywords may be matched to a huge number of literal nodes (called keyword nodes) and some nodes may have very high degree, the propagation and aggregation will be extremely expensive. In these cases, we propose to use PPA instead of MPA for reducing the computational cost. The generated subtrees by PPA are actually a subset of subtrees by MPA. Therefore, every resulting subtree in PPA is an correct match to the keyword query. Thus, we only measure the recall of subtrees generated by PPA. To evaluate the quality of predicate lists suggested by our method, we further use intention index, which is defined as follows: for a keyword K_i, supposing the predicate in user's mind is p for K_i, we examine the average ranking position of p in the suggested list. The smaller the intention index of p, the better this suggestion list is.

We fix $ratio = 50\%$ in top-k factor $(min, max, ratio)$. $min - max$ pair is set in different scales to reflect k's influence on the recall. The details are shown in Fig. 4(a). As we can see, with the enlargement of $min - max$ pairs, the recall on both QS_1 and QS_2 increases steadily. But obviously, QS_2 has a significantly higher recall than QS_1, and the reason is: given that in MPA the average root number in QS_1 and QS_2 are 26276 and 21 respectively, the truncation error in PPA by the same top k has a heavier damage to the recall in QS_1. Thus, max tends to be set as a high number (typically $max = 10000$) but a too high max is not necessary. As showed in Fig. 4(a), $max = 10000$, even the recall is only 0.118, the average returned root number in QS_1 is more than 3000. Users only pay attention to a small number of subtrees ranked in the top of results.

We test the intention index of predicate lists suggested by MPA, PPA ($min = 1000, max = 10000$) and frequency-based statistics which rank predicates only based on their appearance. The predicate in user's mind for a keyword is judged by human testers (an average of 5 testers in our experiments). Fig. 4(b) shows PPA has a slightly worse intention index than MPA esp. on QS_1, which accords with our assumption that PPA is an approximation of MPA, but with an enhanced performance.

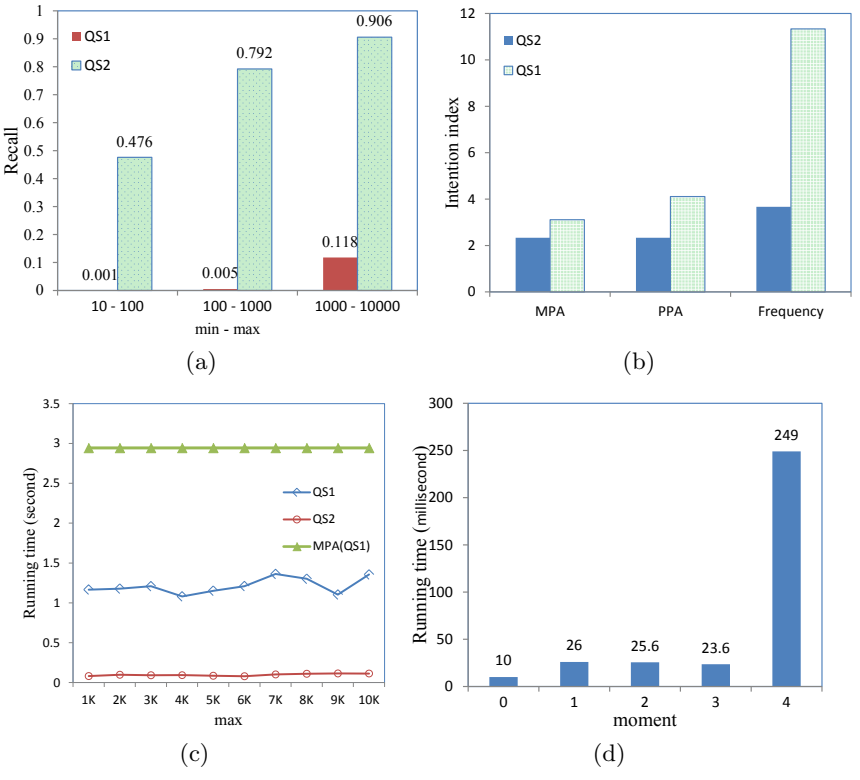


Fig. 4. (a) Recall based on different top k scales. (b) Intention indices of predicate suggestions by MPA, PPA and frequency. (c) The offline running time on different top k scales with $min = 0.1max$. (d) The running time of QS_3 at different moments.

5.3 Performance Evaluation

The predicate suggestion in our method can be divided into offline part and online part. Offline part includes the construction of inverted index between keywords and nodes in the RDF graph, and also the routing of nearby neighbors by shortest distance paths. The performance of PPA using different top k prioritizing strategies and query sets is shown in Fig. 4(c). We only show the timeconsuming of MPA for QS_1 because it is more expensive when there are a huge number of keyword nodes, and therefore suitable for applying PPA. Although going up slightly with fluctuation, the offline running time for QS_1 and QS_2 is around 1 and 0.1 second respectively under different scales of max ($min = 0.1max$).

For online part, we evaluate the performance of predicate suggestion on each moment. QS_3 is used. The results are shown in Fig. 4(d) with setting $max = 10000$. At moment m_0, because no keywords are associated with predicates, PPA is not necessary and a quick frequency-based suggestion is performed. At moment m_1 to m_3, the number of associated keywords increases one by one, which means that #propagators is increasing and #aggreagators is decreasing.

The trend of time cost is slightly descending over these moments. At moment m_4, all the resulting subtrees are generated and ranked, with a time cost relatively higher than the suggestion phases. Overall, the response of predicate suggestion by PPA is rapid even on huge datasets like YAGO and sufficient for online usage.

6 Related Work

RDF is a W3C-endorsed data model that gains popularity in various applications. More and more large-scale RDF collections become available on the Internet, such as DBpedia [4] or YAGO [19], and attract much attention in the academia and industry. Generally, there are two different categories of methods for searching desired information in RDF data. The first category is from the view of database, where RDF statements are stored in SPO triples. The work [1] proposed an impressive RDF storage by vertical partitioning. SPARQL is the W3C standard language for querying RDF. Similar to SQL, it is a structural language that requires users to know the knowledge about the schema and query syntax. Another category of methods tries to avoid this drawback, where users simply need to input some keywords. These methods are based on the view that an RDF dataset can be considered as a graph with each statement corresponding to an edge. In the academia, keyword search on graphs is extensively studied [9,13,12,7,2]. While effective in graphs where edges have no types, it is problematic in the case of RDF graphs, because each edge in the RDF graph is associated with a predicate to indicate the semantics of entities and nodes.

We try to adopt both the simplicity of inputs in keyword search and the awareness of predicates in SPARQL query in this paper. We achieve this goal by adding a predicate selection process, and for each keyword, the suggested predicates will be dynamically ranked to reflect user's intention. Predicate suggestion is not a well-studied area. To achieve a better accuracy and efficiency, we suggest predicates by prioritized propagation and aggregation techniques, which is inspired by authority-based search [11] and bidirectional search [12].

7 Conclusion

In this paper, we propose an effective searching method to answer keyword queries over RDF graphs. Our method takes the advantage of keyword search (easy usage) and that of SPARQL query (better semantic discrimination) by associating predicates with keywords. This is achieved by allowing users to select predicates based on the suggested predicates of keywords. To make predicate suggestion more precisely and efficiently, we adopted the idea of propagation and aggregation of users' preferences. We also examine the issues of keyword search by taking both semantic and structure information into accounts.

Acknowledgments. This work is partially supported by HGJ PROJECT 2010ZX01042-002-002-03, the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University of China under grant No. 20100303614.

References

1. Abadi, D.J., Marcus, A., Madden, S., Hollenbach, K.J.: Scalable semantic web data management using vertical partitioning. In: VLDB, pp. 411–422 (2007)
2. Achiezra, H., Golenberg, K., Kimelfeld, B., Sagiv, Y.: Exploratory keyword search on data graphs. In: SIGMOD Conference, pp. 1163–1166 (2010)
3. Aldous, D., Fill, J.: Reversible markov chains and random walks on graphs. Materials, http://www.stat.berkeley.edu/aldous/RWG/book.html
4. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.G.: DBpedia: A Nucleus for a Web of Open Data. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007)
5. Bhalotia, G., Hulgeri, A., Nakhe, C., Chakrabarti, S., Sudarshan, S.: Keyword searching and browsing in databases using banks. In: ICDE, pp. 431–440 (2002)
6. Elbassuoni, S., Ramanath, M., Schenkel, R., Weikum, G.: Searching rdf graphs with sparql and keywords. IEEE Data Eng. Bull. 33(1), 16–24 (2010)
7. Golenberg, K., Kimelfeld, B., Sagiv, Y.: Keyword proximity search in complex data graphs. In: SIGMOD Conference, pp. 927–940 (2008)
8. Guo, L., Shao, F., Botev, C., Shanmugasundaram, J.: Xrank: Ranked keyword search over xml documents. In: SIGMOD Conference, pp. 16–27 (2003)
9. He, H., Wang, H., Yang, J., Yu, P.S.: Blinks: ranked keyword searches on graphs. In: SIGMOD Conference, pp. 305–316 (2007)
10. Hristidis, V., Gravano, L., Papakonstantinou, Y.: Efficient ir-style keyword search over relational databases. In: VLDB, pp. 850–861 (2003)
11. Hristidis, V., Hwang, H., Papakonstantinou, Y.: Authority-based keyword search in databases. ACM Trans. Database Syst., 33(1) (2008)
12. Kacholia, V., Pandit, S., Chakrabarti, S., Sudarshan, S., Desai, R., Karambelkar, H.: Bidirectional expansion for keyword search on graph databases. In: VLDB, pp. 505–516 (2005)
13. Li, G., Ooi, B.C., Feng, J., Wang, J., Zhou, L.: Ease: an effective 3-in-1 keyword search method for unstructured, semi-structured and structured data. In: SIGMOD Conference, pp. 903–914 (2008)
14. Liu, F., Yu, C.T., Meng, W., Chowdhury, A.: Effective keyword search in relational databases. In: SIGMOD Conference, pp. 563–574 (2006)
15. Manning, C.D., Raghavan, P., Schtze, H.: Introduction to Information Retrieval. Cambridge University Press (2008)
16. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab (1999)
17. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of sparql. ACM Trans. Database Syst., 34(3) (2009)
18. Saad, Y.: Iterative Methods for Sparse Linear Systems. Society for Industrial and Applied Mathematics (2003)
19. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: WWW, pp. 697–706 (2007)
20. Tran, T., Wang, H., Rudolph, S., Cimiano, P.: Top-k exploration of query candidates for efficient keyword search on graph-shaped (rdf) data. In: ICDE, pp. 405–416 (2009)

Intrinsic Dimension Induced Similarity Measure for Clustering

Yu Xiao, Jian Yu, and Shu Gong

Beijing Jiaotong University, Beijing 100044, China
{07112059,jianyu,05112059}@bjtu.edu.cn

Abstract. The goal of clustering is to partition the data points into clusters, such that the data points in the same cluster are similar. Therefore, similarity measure is one of the most critical issues for clustering. In this paper, we present a novel similarity measure based on intrinsic dimension, where the local intrinsic dimension of each data point is considered as a new feature to describe the data points, leading to a new type of similarity measure combining the new feature and original features. The main idea is that the data points in the same cluster are expected to have the same intrinsic dimension while they have similar values of the traditional features. The proposed method is evaluated on some artificial data sets and the experiment results illustrate the effectiveness of the proposed similarity measure. Moreover, the segmentation results of natural images based on the proposed similarity measure show that the intrinsic dimension is worthy of being considered as a new feature of the data points in more applications.

Keywords: Similarity measure, intrinsic dimension, clustering.

1 Introduction

Cluster analysis is one of the most important methods of unsupervised learning, which aims to discover the natural clusters from the given data without any prior knowledge about the class labels. In the literature, a large variety of clustering algorithms have been proposed for many different application fields such as image segmentation, pattern recognition, information retrieval, etc., see [1,2,3,4],ect. In fact, cluster analysis has always been a hot issue, and it is being used everywhere.

Clustering assigns a set of data points into clusters, such that data points within a cluster are more similar to each other than to data points in the different clusters. Therefore, how to define and compute similarity is very crucial for clustering. Although there have been many discussions about similarity measure in the literature [5,6,7,8,9,10,11], unfortunately, it is still impossible for us to find a uniform rule about the definition of similarity measure since the notion of similarity is closely related to the purpose of the study, and different requirements demand different definitions of similarity. There are two key factors in similarity measure: feature selection and mathematic formulation of similarity measure. Sometimes, good features can make the clustering problem become much easier

J. Tang et al. (Eds.): ADMA 2011, Part II, LNAI 7121, pp. 110–123, 2011.

regardless of the mathematic formulation of similarity measure. The most commonly used features are the known dimensions in the feature space, for example, a data point $x \in \mathbb{R}^D$ has D obvious features. Figure 1(a) shows a set of n data points $X = \{x_1, \ldots, x_n\} \in \mathbb{R}^2$ and the expected clustering result of this data set is illustrated in Fig. 1(b). Generally, based on the two features (position information), the '$*$' marked point is more similar the '$\circ$' marked point than it to the '$\diamond$' marked point, which is not consistent with the intension of clustering. Usually, it is difficult to obtain good clustering results, especially for the clustering algorithms based on similarity matrix such as hierarchical clustering. But it is easy for us to get the expected clusters since the points have well separated topology structure. The left cluster and the right cluster are distributed in plane while the other cluster between them has line structure. In this paper, in order to solve this problem, we use intrinsic dimension as an additional feature to describe the data points, leading to a more appropriate similarity measure. Adding the intrinsic dimension feature of each data point, the problem becomes much easier since the intrinsic dimension of points on the line structure is 1 while the intrinsic dimension of the left data points is 2.

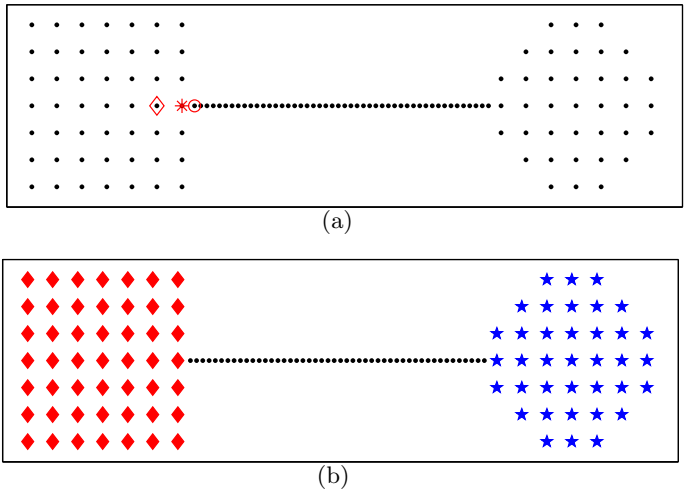


Fig. 1. (a) A toy data set on a 2D space; (b) The expected clustering result

Intrinsic dimension estimation is a key step in dimension reduction problem, which aims to solve the curse of dimensionality since most of the learning algorithms perform poorly in high dimensions. Till now, a large variety of intrinsic dimension estimation algorithms have been proposed, see [12] for a comprehensive review on dimension estimation algorithms. The existing approaches to estimating the intrinsic dimension can be roughly divided into two groups: global approaches and local approaches. The global approaches aim to gain a global estimation directly based on the whole data set, such as [13,14] and the recent

reference [15,16]. The local ones usually use local subsets of the data to form a global estimation of dimension. Fukunaga and Olsen's [17], Pettis et al.'s [18], as well as more recent work [19],[20], and [21] belong to this category.

In this paper, we focus on the application of the similarity measure based on the intrinsic dimension feature. Thanks to the various intrinsic dimension estimation algorithms referred above, the intrinsic dimension of each data point can be gained by using local intrinsic dimension estimation methods directly or applying global methods over a small neighborhood around each data point as a local dimension approach. A brief review on some commonly used intrinsic dimension estimation methods is presented in section 2. Section 3 presents the proposed framework of clustering based on the new similarity measure combining the traditional features and the intrinsic dimension feature. Section 4 illustrates numerical experiments on both artificial data sets and natural images. Finally, section 5 concludes this paper.

2 Intrinsic Dimension Estimation

As discussed above, the existing methods to estimating the intrinsic dimension can be classified into two groups: the global methods and the local methods. In this paper, we aim to get the intrinsic dimension of each data point, called point intrinsic dimension, instead of the global intrinsic dimension of the whole data set by using the existed approaches. Although both local algorithms and global algorithms are available for point intrinsic dimension estimation when considering the local subset as a whole data set, we prefer to use local methods rather than global ones since the global ones usually fail when the size of the data set is small. Next, we review three local intrinsic dimension estimation methods: the local PCA (principal components analysis) algorithm [17],the k-nearest neighbor algorithm [19] and the maximum likelihood estimation (MLE) algorithm [20].

2.1 Dimension Estimation Based on Local PCA

The local PCA algorithm for dimension estimation was proposed by Fukunga and Olsen, which uses PCA locally on many small subregions to estimate local dimension, and then gain a global estimation of dimension by averaging the values of local dimension estimates. Since the global PCA can not get satisfactory results on nonlinear manifolds, Fukunga and Olsen apply PCA to estimate the dimension of small subregion which can be considered as a linear sub-manifold.

Let $X = \{x_1, \ldots, x_n\} \in \mathbb{R}^D$ be a data set of n data samples. A small subset $M = \{x_m^1, \ldots, x_m^k\} \subset X$ lies on a small submanifold, and the intrinsic dimension of this subset can be estimated by PCA as follows.

– Calculate the covariance matrix for each subset M:

$$\mathrm{C} = 1/k \sum_{i=1}^{k} \left(x_m^i - \mu_m\right) \left(x_m^i - \mu_m\right)^T \tag{1}$$

where $\mu_m = 1/k \sum_{i=1}^{k} x_m^i$ is the mean vector.

– Calculate the eigenvalues of C:$\lambda_m^1 \geq \cdots \geq \lambda_m^D$. The intrinsic dimension d of M is the number of dominant eigenvalues among all gained eigenvalues, which can be obtained as

$$d = \arg\min_j \{j \,|x_m^{j+1}/x_m^j < \varepsilon\} \tag{2}$$

and the threshold ε is a small value between 0 and 1.

The global intrinsic dimension of the data set X can be gained based on the local intrinsic dimensions of the subsets, such as the mean of the gained local intrinsic dimensions.

2.2 Dimension Estimation Based on *k*-Nearest Neighbor Graphs

Costa et al. [19] proposed a local intrinsic dimension method based on a global dimension estimator [14], which estimates the intrinsic dimension by calculating the length of the k-nearest neighbor graph. Let $\mathcal{N}_{k,i}$ be the set of k-nearest neighbors of x_i in X, which is obtained based on the usual Euclidean (L_2) distance in $\mathbb{R}^D$, leading to a k-NN graph where there is an edge between each point in X and its k-nearest neighbors. The total edge length of the k-NN graph is defined as

$$L_{\gamma,k} = \sum_{i=1}^{n} \sum_{x \in \mathcal{N}_{k,i}} |x - x_i|^\gamma \tag{3}$$

where $\gamma > 0$ is a power weighting constant. In [14], Costa et al. got a relationship between $L_{\gamma,k}$ and the intrinsic dimension d:

$$\log L_{\gamma,k} = a \log n + b + \varepsilon_n \tag{4}$$

where $a = (d - \gamma)/d$, b is an unknown constant with respect to a and ε_n is an error residual that goes to zero with probability 1 as $n \longrightarrow \infty$. The estimator works on subsets $X_1, X_2, \ldots, X_Q$ gained using the nonoverlapping block bootstrapping method. Defining $l = [\log L_1, \ldots, \log L_Q]^T$ and $A = \begin{bmatrix} \log n_1 & \ldots & \log n_Q \\ 1 & \ldots & 1 \end{bmatrix}^T$ where n_i is the cardinality of the subset X_i and L_i is the total length of the k-NN graph for X_i. Finally, the linear least square strategy is used to solve the following linear vector model to get $\hat{a}$ and $\hat{d} = round\{\gamma/1 - \hat{a}\}$.

$$l = A \begin{bmatrix} a \\ b \end{bmatrix} + \varepsilon \tag{5}$$

The proposed local intrinsic estimation method applied the above global estimator to estimate the intrinsic dimension for each data point based on its local k-NN graph.

2.3 Dimension Estimation Based on Maximum Likelihood

In [20], Levina and Bickel proposed a maximum likelihood estimator (MLE) of intrinsic dimension derived by applying the principle of maximum likelihood to the distances between close neighbors. The experiments in their work illustrated that the MLE method performance good on a range of artificial and real data sets. Similar to the above k-NN estimator, the MLE method estimate the intrinsic dimension of data points based on their k nearest neighbors and the intrinsic dimension of each data point is formulated as follows.

$$\hat{d}_k(x_i) = \left[\frac{1}{k-1} \sum_{j=1}^{k-1} \log \frac{T_k(x_i)}{T_j(x_i)} \right]^{-1} \tag{6}$$

where $T_k(x_i)$ is the distance from x_i to its kth nearest neighbor. The global intrinsic dimension for the data set is simply calculated by averaging the local intrinsic dimension of all data points.

$$\hat{d}_k = \frac{1}{n} \sum_{i=1}^{n} \hat{d}_k(x_i) \tag{7}$$

In order to reduce the effect of the choice of k, the final intrinsic dimension can be calculated as the average over a range of small to moderate values $k = k_1, \ldots, k_2$.

$$\hat{d} = \frac{1}{k_2 - k_1 + 1} \sum_{k=k_1}^{k_2} \hat{d}_k \tag{8}$$

A revised version of the MLE method was presented by MacKay and Ghahramani [22], which suggest that it is more sensible to average the inverses $\hat{d}_k{}^{-1}(x_i)$ rather than averaging the estimators $\hat{d}_k(x_i)$.

$$\hat{d}_k{}^{-1} = \frac{1}{n} \sum_{i=1}^{n} \hat{d}_k{}^{-1}(x_i) \tag{9}$$

3 New Similarity Measure for Clustering

There are two important parts for clustering algorithm: similarity measure and the clustering algorithm. In this paper, we focus on the study of the first part. A new similarity measure based on the intrinsic dimension is presented in this section.

3.1 Intrinsic Dimension Feature

Seen from Fig. 1, we observe that the intrinsic dimension can reflect the local topology structure, and we propose to use intrinsic dimension as a new feature

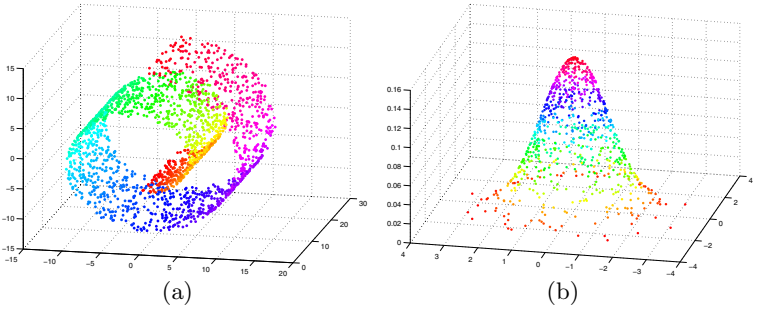


Fig. 2. (a) The Swissroll data set of 2000 points; (b) The Gauss data set of 800 points

that take into account of the topology structure. In order to choose a related optimal estimator to estimate the intrinsic dimension of all data points, we make a simple comparison experiment on two commonly used data sets with three local algorithms reviewed in section 2 and a global method called incising ball algorithm [15]. For Local PCA algorithm and incising ball algorithm, we form a small data set $N_{k,i}$ of k-nearest neighbors of x_i in X, and apply the estimators on the neighborhood data set to estimate the intrinsic dimension of each data point. Figure 2 shows the data sets and Figure 3 presents the estimate of the intrinsic dimension. The true intrinsic dimension of the two data sets is 2.

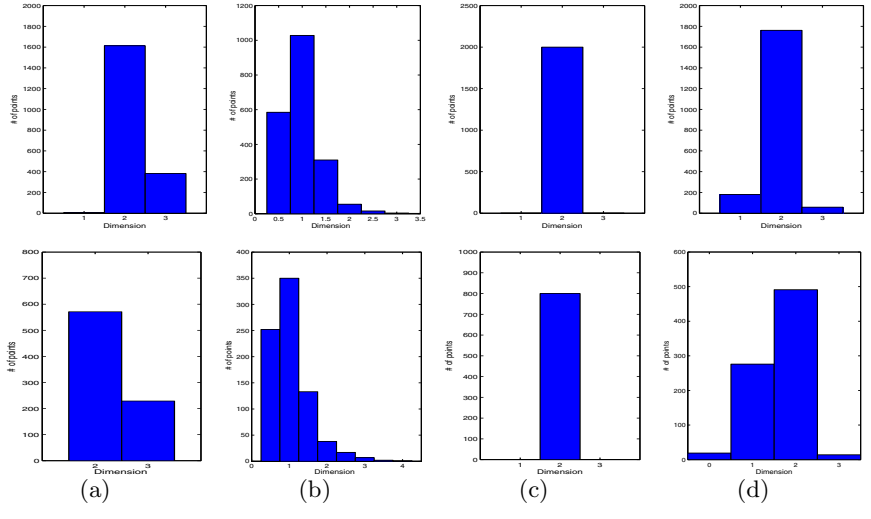


Fig. 3. Histogram of dimension estimates for Swissroll data set and Gauss data set according to the first row and the second row respectively. From (a) to (d) in column: k-nearest neighbor algorithm, MLE algorithm, local PCA algorithm, and the incising ball algorithm.

Observed from Fig. 3, local PCA outperforms the other two methods. In this paper, we use local PCA for estimation of intrinsic dimension for each data point, which has been used in [23] for manifold clustering. Usually, the neighborhoods are defined through Euclidean distance, which intent to form spherical regions around each data point. Here, we give a new method to choose the neighborhoods by considering both the Euclidean distance and geodesic distance[24], which can not only ensure that the point is close to its neighbor points in Euclidean space but also better reflect the local topology structure than only using Euclidean distance. Figure 4 gives an illustration of the effectiveness of the proposed method for choosing neighborhoods.

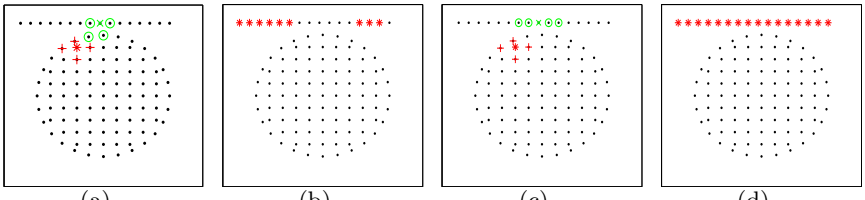


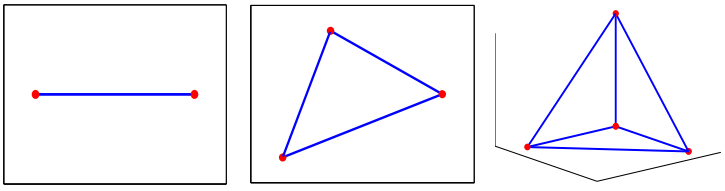
Fig. 4. (a) The selected 4 neighbors for two data points through Euclidean space; (b)The gained intrinsic dimension of (a) by local PCA algorithm, '$*$' denotes that the intrinsic dimension is 1 while '$\bullet$' denotes that the intrinsic dimension is 2. (c)The selected neighbors for two data points through our method; (d)The gained intrinsic dimension of (c) by local PCA algorithm. Note: data points marked '+' are the selected neighbors for point marked '$*$', and data points marked '$\circ$' denotes the selected neighbors for point marked '$\times$'.

Given a set of data points $X = \{x_i \in \mathbb{R}^D, i = 1, 2, \ldots, n\}$. For each data point x_i, the k nearest neighbors are calculated by the following two steps.

1. Calculate the $k^{'}$ nearest neighbors of data point x_i through Euclidean distance, where $k^{'} > k$ and we set $k^{'} = 2k$.
2. Sort the $k^{'}$ neighbors according to the geodesic distance between each neighbor point and x_i: $x_{i1}, \ldots, x_{ik'}$, the first x_{i1} has shortest geodesic distance from point x_i while the last $x_{ik'}$ is the $k^{'}$th neighbor according to the geodesic distance. The first k points in the sorted $k^{'}$ neighbors $\{x_{i1}, \ldots, x_{ik}\}$ form the small subset $\mathcal{N}_{k,i}$ used for the intrinsic dimension estimation of the data point x_i.

After we get the set $\mathcal{N}_{k,i}$, the local PCA algorithm is applied to estimate the intrinsic dimension and the gained intrinsic dimension will be considered as a new feature to get a better description of data points. For local PCA algorithm, how to choose the parameter k is still an open problem. Next, we discuss about how to set the number of neighbors.

Seen from Fig. 5, the intrinsic dimension of the small data set of two data points is 1 in the left sub-figure of Fig. 5, and the intrinsic dimension of the other



(a) $d = 1$ (b) $d = 2$ (c) $d = 3$

Fig. 5. An illustration of data sets with different intrinsic dimensions

two small data sets are respectively 2 and 3. This simple illustration shows that a data set should contain at least $d + 1$ data points if the intrinsic dimension of this data set is d. Generally speaking, it only needs $d + 1$ neighbors to estimate whether d is the intrinsic dimension of the data point. However, the number $d+1$ is usually too small in practice, which is also dependent on the density of the data set. Low density data sets need few neighbors while high density data sets need more neighbors since the neighbors may too close to each other to estimate the intrinsic dimension correctly by PCA. In this paper, we set k between $D+1$ and $2D$, where D is the original dimension of the data points. Whether there is a relationship between the size of the neighborhood and the corresponding data point will be our future work.

After finding the neighbors of the data points, most of the existed estimation methods directly estimate the intrinsic dimension of the corresponding data points, which do not consider the effect of the outliers in the data set. Obviously, the neighbor set of each outlier can not reflect the true topology structure of the outlier. In order to avoid the effect of outliers, we suggest to detect the outliers first and propose a simple method to detect the outliers as follows. For these detected outliers, the similarity measure does not consider the new feature of intrinsic dimension.

1. Calculate the k nearest neighbors of data point x_i through Euclidean distance, denoted as $\mathcal{N}_{ik}$.
2. Compute the *mean* μ and *variance* σ of $\mathcal{N}_{ik}$. The data x_i is considered as a outlier if $\|x_i - \mu\|_2 > t\sigma$, and $t = 3$ in this paper.

3.2 Similarity Measure

Based on the new feature derived from intrinsic dimension, a new similarity measure combining new feature and the original features is proposed in this section. The final goal of this work is to define a new similarity measure to improve the clustering results.

Generally speaking, there are two types of similarity definition: the feature is directly used as a indicator function, i.e. points with the same value of the feature are clustered into the same cluster; the other type is to combine different features to compute the similarity through some mathematics formulations.

The mathematical formulation of the first type of similarity definition can be expressed as follows.

$$S(i,j) = \mathbf{I}(d_i, d_j) \tag{10}$$

where d_i is the intrinsic dimension of point x_i and $\mathbf{I}(d_i, d_j)$ is equal to 1 when $d_i = d_j$ while $\mathbf{I}(d_i, d_j)$ is equal to 0 when $d_i \neq d_j$. Similarity based on (10) usually can not get good clustering results in the following cases.

1. The intrinsic dimension of all data points in the data set is equal to each other, when it can not partition the clusters in the data set only considering the intrinsic dimension.
2. Any other property of the data points are very different to each other even if they have the same intrinsic dimension, see Fig. 1 for example, two data points (one is from the left cluster and the other is from the right cluster) are far from each other in Euclidean space can not be clustered into one cluster in spite of the same intrinsic dimension of the two data points.
3. The clustering results are heavily dependent on the intrinsic dimension estimation and so it fails to get satisfactory clustering results based on (10) when the dimension estimation methods give wrong estimation of the intrinsic dimension.

In order to overcome the above three cases, we try to use the second type of similarity definition, i.e., combining the intrinsic dimension feature and other features.There are two similarity models to combine different features: additive model and multiplicative model.

$$\text{Additive model:} \quad S_{AM}(i,j) = \frac{\sum_p w_p s_{ij}^p}{\sum_p w_p} \tag{11}$$

$$\text{Multiplicative model:} \quad S_{MM}(i,j) = \prod\nolimits_p s_{ij}^p \tag{12}$$

where p denotes a feature in the feature set and w_p is the weight of feature p.

Different applications prefer to use different similarity models. One commonly used representative of the multiplicative model is the similarity measure based on Shepard's similarity function [5] defined as follows.

$$S^P(i,j) = exp(-\alpha(\|x_i - x_j\|_2)^\beta) \tag{13}$$

where P denotes some feature or a set of features, and α is the scaling parameter. The Shepard's similarity function becomes Gaussian kernel similarity function when $\beta = 2$, which is commonly used in clustering algorithms based on similarity matrix, such as spectral clustering [25,26,27].

4 Experiments

In this section, numerical experiments are presented to illustrate the effectiveness of the new similarity measure based on the intrinsic dimension feature. Due to

the limited space within this paper, only Gaussian kernel similarity function based on combined features is tested on both artificial data sets and images using spectral clustering algorithm [25].

4.1 Artificial Data Sets

Figure 6 shows the artificial data sets and the clustering results based on the new similarity measure considering the intrinsic dimension. The first data set consists of 800 points sampled on the "swissroll" manifold and 348 points on a three-dimensional hypersphere which lies in the "swissroll". Two view angles of this three-dimensional data set are shown in the first two rows of Figure 6.

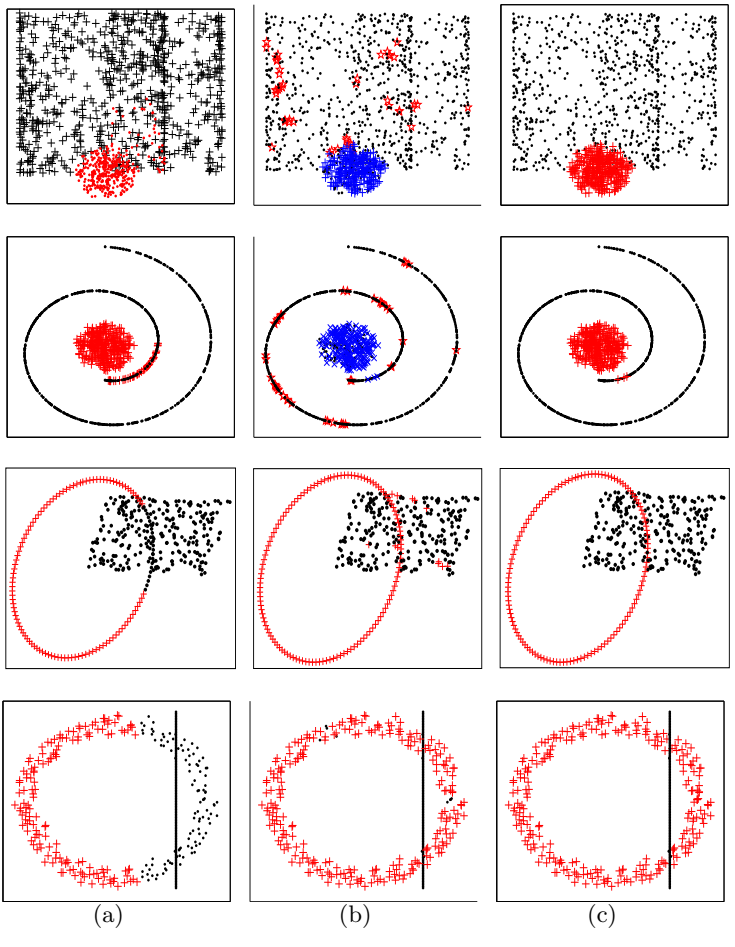


Fig. 6. Clustering results based on different similarity measures for three data sets. From (a) to (c) in column: similarity based on original features, similarity based on intrinsic dimension, new similarity defined as (14).

Given a data set $X = \{x_i \in \mathbb{R}^D, i = 1, \ldots, n\}$, the main steps of the framework for clustering can be expressed as follows.

- Calculate the intrinsic dimension for all data points, denoted as d_i for x_i.
- The similarity s_{ij} is defined as:

$$s_{ij} = \exp\left(-\alpha_1 \left\|x_i - x_j\right\|_2^2\right) * \exp\left(-\alpha_2 \left(1 - \mathbf{I}(d_i, d_j)\right)\right) \tag{14}$$

- Apply the spectral clustering algorithm [25] based on the gained similarity matrix.

where $\alpha_1 = \frac{1}{\sigma_i \sigma_j}$, and σ_i (σ_j) is a local scaling parameter defined as the distance between x_i (x_j) and its 5th neighbor in our experiment [27]. We use $\alpha_2 = 6$ for both of the two data sets. The new similarity measure is compared with other two similarity measures: similarity based on original features, $s_{ij} = \exp\left(-\frac{1}{\sigma_i \sigma_j} \left\|x_i - x_j\right\|_2^2\right)$; similarity based on intrinsic dimension (10).

Seen from Fig. 6, we notice that the clustering algorithm based on the traditional similarity measure can not get satisfactory results for the three complex data sets: one data set with two clusters close to each other and the other two data sets with intersecting clusters, and neither the similarity measure based on intrinsic dimension can do a good job for all of the three data sets. The proposed similarity measure outperforms the other two similarity measures.

4.2 Image Segmentation

We now apply the proposed similarity for image segmentation. The intrinsic dimension has been used for the description of image regions, see [28],[29],etc. Different texture regions in images usually has different intrinsic dimensions and the intrinsic dimension can be considered as an indicator variable to partition the images into disjoint regions with patches in the same region have equal intrinsic dimension. As we discussed above, the segmentation results are heavily dependent on the estimation of intrinsic dimension, which may fail for natural images. The natural gray images used in our experiments are from the Berkeley Segmentation Datasets, in which each image contains 154401 (321×481 or 481×321) pixels. To use local dimension estimation, the data sample x_i is a 25-dimensional vector representing a rasterized 5×5 patch of the image. The similarity between patches x_i and x_j is defined as:

$$s_{ij} = \mathrm{e}^{-\sigma_I \|I_i - I_j\|_2^2} * e^{-\sigma_d (1 - I(d_i, d_j))} * \begin{cases} \mathrm{e}^{-\sigma_l \|l_i - l_j\|_2^2} & \text{if } \|l_i - l_j\|_2^2 \leq r \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

where I_i is the gray level of the patch i, which is the mean value of all pixels' gray value in the patch. l_i is the location of the patch i in the image, which can be considered as a superpixel with the gray level I_i. We take $r = 2$ in all experiments for image segmentation in this paper i.e. 8-neighbors of each patch,

and the location scaling parameter is fixed as $\sigma_l = 0.001$, which has little effect on the segmentation results. The value of the color scaling parameter σ_I varies with the images, the value of which is around 0.005 and the dimension scaling parameter σ_d is around 5.

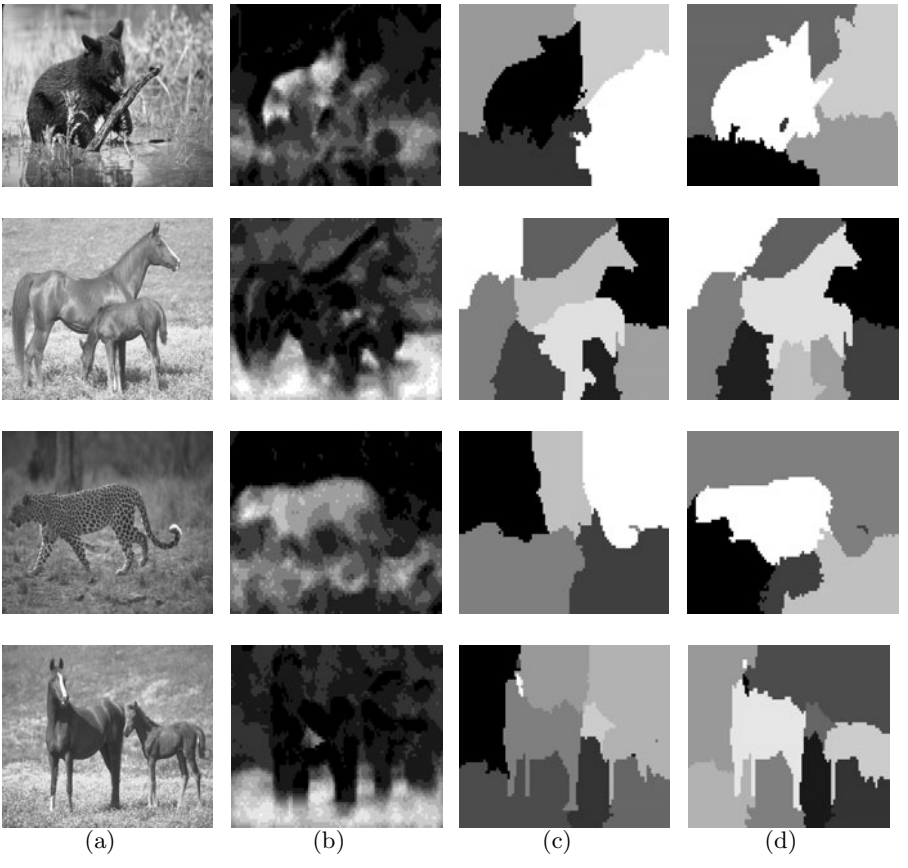


Fig. 7. Segmentation results based on different similarity measures. From (a) to (d) in column: original image, intrinsic dimension, similarity based on original features, new similarity defined as (15).

Figure 7 shows the images and the segmentation results. The proposed similarity works better than the traditional used features for partitioning the texture regions and non-texture regions. Please note that the goal of the experiment for image segmentation here is to show that the proposed similarity measure has the potential to give better results than traditional similarity based on original features, rather than to propose a new segmentation algorithm to get very good segmentation results, which is also a very challenging topic we are concentrating on.

5 Conclusion

In this paper, we have proposed a novel similarity measure based on intrinsic dimension. The local intrinsic dimension of each data points is considered as a new feature to describe the data points, leading to a new type of similarity measures combining the new feature and original features. In order to better estimate the intrinsic dimension of each data point, we present a new method to choose the neighbors of data points which is propitious to discover the true topology structure. The proposed similarity measure has been compared with the traditional similarity measures on artificial data sets and images, and the experiment results demonstrated the effectiveness of the intrinsic dimension induced similarity measure.

Although the proposed similarity measure has improved the clustering results in some degree, there is still big room to ameliorate the similarity measure based on intrinsic dimension, such as, how to make a better estimation of the intrinsic dimension using local dimension estimation methods; how to combine different features together effectively, which is still our future research topic.

Acknowledgments. The research was partially supported by the 973 Project under Grant no. 2007CB311002, and by the National Natural Science Foundation under Grant no. 60875031, 90820013, 61033013.

References

1. Baraldi, A., Blonda, P.: A Survey of Fuzzy Clustering Algorithms for Pattern Recognition-Part I and II. IEEE Transaction on Systems, Man, and Cybernetics, Part B 29, 778–801 (1999)
2. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers (2000)
3. Jain, A.K.: Data clustering: 50 years beyond K-means. Pattern Recognition Letters 31, 651–666 (2010)
4. Filippone, M., Camastra, F., Masulli, F., Rovetta, S.: A survey of kernel and spectral methods for clustering. Pattern Recognition 41, 176–190 (2008)
5. Shepard, R.N.: Toward a universal law of generalization for psychological science. Science 237, 1317–1323 (1987)
6. Tversky, A.: Features of similarity. Psychological Review 84, 327–352 (1977)
7. Santini, S., Jain, R.: Similarity measures. IEEE Transaction on Pattern Analysis and Machine Intelligence 21, 871–883 (1999)
8. Navarro, D.J., Perfors, A.F.: Similarity, feature discovery, and the size principle. Acta Psychologica 133, 256–268 (2010)
9. Cheng, H., Liu, Z., Yang, J.: Sparsity Induced Similarity Measure for Label Propagation. In: 12th IEEE International Conference on Computer Vision, pp. 317–324. IEEE Press, Kyoto (2009)
10. Jiang, H., Ngo, C.W., Tan, H.K.: Gestalt-based feature similarity measure in trademark database. Pattern Recognition 39, 988–1001 (2006)
11. Liu, C.: The Bayes Decision Rule Induced Similarity Measures. IEEE Transaction on Pattern Analysis and Machine Intelligence 29, 1086–1090 (2007)

12. Camastra, F.: Data dimensionality estimation methods: A survey. Pattern Recognition 36, 2945–2954 (2003)
13. Kégl, B.: Intrinsic Dimension estimation using packing numbers. In: Neural Information Processing Systems, pp. 681–688. MIT Press, Canada (2002)
14. Costa, J.A., Hero, A.O.: Geodesic entropy graphs for dimension and entropy estimation in manifold learning. IEEE Transaction on Signal Processing 52, 231–252 (2004)
15. Fan, M., Qiao, H., Zhang, B.: Intrinsic dimension estimation of manifolds by incising balls. Patter Recoginition 42, 780–787 (2009)
16. Eriksson, B., Crovella, M.: Estimation of Inrinsic Dimension via Clustering. BU/CS Technical Report (2011)
17. Fukunaga, K., Olsen, D.: An algorithm for finding intrinsic dimensionality of data. IEEE Transaction on Computers C-20 (1971)
18. Pettis, K., Bailey, T., Jain, A., Dubes, R.: An intrinsic dimensionality estimator from near-neighbor information. IEEE Transaction on Pattern Analysis and Machine Intelligence 1, 25–36 (1979)
19. Costa, J.A., Girotra, A., Hero, A.O.: Estimating local intrinsic dimension with k-nearest neighbor graphs. In: IEEE Workshop on Statistical Signal Processing, pp. 417–422 (2005)
20. Levina, E., Bickel, P.: Maximum likelihood estimation of intrinsic dimension. In: Neural Information Processing Systems. MIT Press, Vancouver (2004)
21. Carter, K.M., Raich, R., Hero, A.O.: On local intrinsic dimension estimation and its applications. IEEE Transaction on signal processing 58, 650–663 (2010)
22. MacKay, D.J.C., Ghahramani, Z.: Comments on 'Maximum likelihood estimation of intrinsic dimension' by E.Levina and P.Bickel (2005), http://www.inference.phy.cam.ac.uk/mackay/dimension
23. Wang, Y., Jiang, Y., Wu, Y., Zhou, Z.-H.: Multi-Manifold Clustering. In: Zhang, B.-T., Orgun, M.A. (eds.) PRICAI 2010. LNCS, vol. 6230, pp. 280–291. Springer, Heidelberg (2010)
24. Fischer, B., Zöller, T., Buhmann, J.M.: Path Based Pairwise Data Clustering with Application to Texture Segmentation. In: Figueiredo, M., Zerubia, J., Jain, A.K. (eds.) EMMCVPR 2001. LNCS, vol. 2134, pp. 235–250. Springer, Heidelberg (2001)
25. Ng, A.Y., Jordan, M.L., Weiss, Y.: On spectral clustering: Analysis and algorithm. In: Neural Information Processing Systems, pp. 849–856. MIT Press, Canada (2002)
26. Shi, J.B., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 22, 888–905 (2004)
27. Zelnik-Manor, L., Perona, P.: Self-Tuning Spectral Clustering. In: Neural Information Processing Systems, pp. 1601–1608. MIT Press, Canada (2005)
28. Petland, P.A.: Fractal-based description of natural scenes. IEEE Transaction on Pattern Analysis and Machine Intelligence 6, 661–674 (1984)
29. Chaudhuri, B.B., Sarkar, N.: Texture segmentation using fractal dimension. IEEE Transaction on Pattern Analysis and Machine Intelligence 17, 72–77 (1995)

Learning to Make Social Recommendations: A Model-Based Approach

Xiongcai Cai, Michael Bain, Alfred Krzywicki, Wayne Wobcke, Yang Sok Kim, Paul Compton, and Ashesh Mahidadia

School of Computer Science and Engineering,
The University of New South Wales, Sydney, NSW 2052, Australia
{xcai,mike,alfredk,wobcke,yskim,compton,ashesh}@cse.unsw.edu.au

Abstract. Social recommendation, predicting people who match other people for friendship or as potential partners in life or work, has recently become an important task in many social networking sites. Traditional content-based and collaborative filtering methods are not sufficient for people-to-people recommendation because a good match depends on the preferences of *both* sides. We proposed a framework for social recommendation and develop a representation for classification of interactions in online dating applications that combines content from user profiles plus interaction behaviours. We show that a standard algorithm can be used to learn a model to predict successful interactions. We also use a method to search for the best model by minimising a cost based on predicted precision and recall. To use the model in real world applications to make recommendations, we generate candidate pairs using the selected models and ranked them using a novel probabilistic ranking function to score the chance of success. Our model-based social recommender system is evaluated on historical data from a large commercial social networking site and shows improvements in success rates over both interactions with no recommendations and those with recommendations generated by standard collaborative filtering.

Keywords: Machine Learning, Data Mining, Information Retrieval, Recommender Systems, Social Recommendation, Social Media.

1 Introduction

Traditional *recommender systems* attempt to discover user preferences over items by modelling the relation between users and items. The aim is to recommend items that match the *taste* (likes or dislikes) of users in order to assist the active user, i.e., the user who will receive recommendations, to select items from a potentially overwhelming set of choices. By applying recommendation techniques it is possible to increase the likelihood of the successful purchase of products or services by the active user, since services or products are personalised and presented to the active user using information obtained from the purchasing behaviour of like-minded users, e.g., as in the well-known Amazon.com recommender system [1]. More recently, the importance of *social* recommender systems

J. Tang et al. (Eds.): ADMA 2011, Part II, LNAI 7121, pp. 124–137, 2011.

to enable personalisation of activities such as search in social media has been recognized [2], where one goal is to recommend to the active user other people with whom they may want to interact.

Approaches to recommender systems can be categorised as content-based or collaborative filtering methods[1]. Collaborative filtering (CF) [4,5,6,7,1,8,9,10,11] methods recommend items based on aggregated user preferences of those items, which does not depend on the availability of item descriptions. In content-based methods [12] on the other hand, the user will be recommended items similar to those the user preferred in the past. This is usually based on models created from item descriptions using machine learning techniques. Both methods have strengths and weaknesses and can be seen as complementary; in particular content-based systems may be preferred where there is *profile* information in the form of item descriptions that can be used, e.g., to provide explanation of why recommendations were made, and when new items are added that have not been purchased by other users (the "cold-start" problem), while with collaborative filtering, user behaviours are used to generate user preferences. In this paper we aim to integrate behaviour information into content-based recommendation by simple yet powerful learning and ranking approaches rather than complicated models such as matrix factorisation and latent semantic analysis [9] to avoid expensive computation, which is vital for the large real world commercial dataset used in our experiments.

In social recommendation, for example in social networking sites, instead of mapping users to *items*, recommendation techniques are based on user to *user* interactions. In this paper, we use a model-based method that integrates user profile and behaviour information using machine learning and probabilistic ranking.

The method models the relationship between the features of a pair of candidate users as a *classifier*, where features are based on selected user profile and behaviour information, and each interaction is labelled with a target value that indicates whether there is a successful interaction between the candidates in that pair. Although in principle any classifier learning method could be applied, in this paper we used a standard commercial decision tree learning method from which a set of classification rules is automatically extracted. A learned model is selected by maximising cost-weighted performance on a training set of interactions then converted into rule sets containing characteristics of both the active user and recommended candidates to allow candidate list generation in a comprehensible knowledge representation. To make recommendations, rules are retrieved according to the characteristics of the active users and the corresponding recommended candidates are generated from the rules. Before recommended candidates are presented to the active user, they are ranked based on the probability of having a successful interaction with the active user, which again enhances the content-based method by behaviour information. An overview of the proposed recommender system is shown in Figure 1.

[1] Hybrid systems using both methods also exist [3].

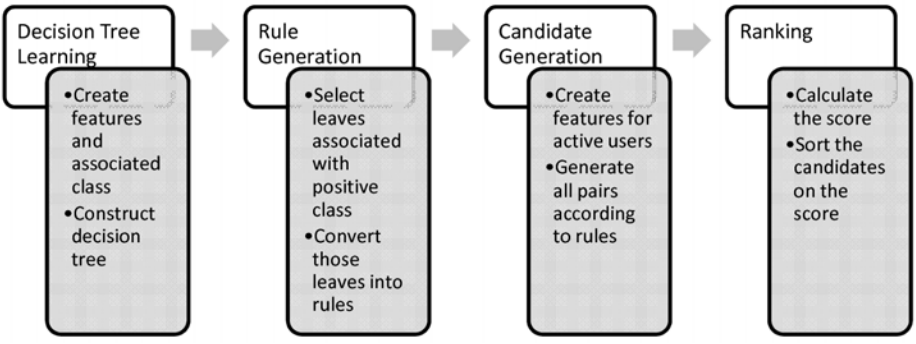


Fig. 1. Overview of the proposed social recommendation framework

The main contribution of this paper is a new model-based social recommender system, summarised as follows:

1. A general model-based people to people recommendation framework integrating user behaviour and profile content;
2. A new formula for ranking candidates using probabilistic modelling based on user behaviour information;
3. Integration of feature construction, model learning and selection, candidate generation and ranking, and evaluation of the recommendation of potential friends or partners on data from a real world online dating site.

The paper is organised as follows. Section 2 presents the problem definition of people to people recommendation in social networking sites. Section 3 develops a decision tree-based machine learning system for recommendation. Section 4 describes the method to generate recommendation candidates. Section 5 presents a method for ranking the candidates based on probabilistic modelling. Experimental evaluation is in Section 6 and we conclude in Section 7.

2 Problem Definition

Using machine learning for constructing a social recommender system requires a training set Tr to train a recommender, which produces a list of user-candidate pairs that defines a ranked list of candidates for a subset of users, which is then tested on a test set Ts. That is, the recommender produces a list of pairs of users (u_a, u_r) where u_a is the active user, the user who will receive the recommendations, and u_r is a user recommended to u_a. In this paper, the problem of learning a social recommender is defined as follows:

1. construction and selection of features based on user profiles and behaviours; and
2. learning and selection of models for generating candidate pairs (u_a, u_r); and
3. ranking and selecting a subset of generated candidates u_r for each active user u_a.

We learn models of user interactions in a social network site and use this modelling for a recommender system. Typically users start their interactions by selecting and viewing matching users based on their profiles. When interesting matches are found, users proceed to express their interest in an interaction, for example, in an online dating site, by sending a short predefined message. If a response is received and is encouraging, the original user may decide to initiate further contact. The selection of the match made by the user for this initial interaction is crucial for the success of the future interaction of the pair.

In this research, we focus on modelling such interactive messaging behaviours to predict the outcome of initiating an interaction. Specifically, a user (the *sender*) is allowed to initiate a contact to another user (the *recipient* or *receiver*). Once a contact is received, the recipient can either ignore the received contact without any reply, or send back a response drawn from a predefined set of alternatives. Typically, such a response clearly either encourages or discourages further interaction with the sender. For the purposes of this research, a response is defined as either *positive* or *negative* according to the predefined content of the response.

In terms of the sender initiating contact that might lead to a friendship or partnership, the outcome of sending a contact is deemed either *successful* or *unsuccessful*. More specifically, a contact with a positive response is defined to be a successful interaction, while a contact with a negative response is an unsuccessful interaction.

3 Model Creation

First it is necessary to determine the type of knowledge to be acquired and its use in the model. Given two classes of interaction outcome, a model must be learned from the database to predict each interaction (the target function). The problem is that this is typically a very large search space, and the best strategy to search this space for an optimal solution is not known. Given this setting, the obvious choice for the type of model to be constructed is a function that defines, for each pair of users, a value indicating whether or not an interaction is likely to be successful.

3.1 Representation of the Target Function

There are a variety of choices of representation for the target function, ranging from generative models that represent a probability distribution under certain assumptions, to discriminative models that focus on the conditional relation between features (attributes) and the class label or value to be predicted. In this work, it is natural to represent the task of classifying interactions as either successful or unsuccessful as a discriminative model, although a generative model could also have been used.

The choice of representation of the target function and associated learning method involves several crucial trade-offs which are beyond the scope of this

paper. Initial experimentation led to the selection of a representation based on *decision trees*, or *rules* derived from them, which map observations about both users involved in an interaction to conclusions about the outcome of the interaction. Note that the choice of decision tree learning is not important to our approach, since other classifier learning methods could have been used. However, given the features used in the rules, this choice allowed a balance of expressive ability, computational complexity of learning and the requirements for data.

3.2 Feature Selection and Construction

We used three sets of features. The first set is based on available user profile records in the data from the online dating site. Feature construction generated two further sets of features.

User Profile Features. *Basic* attributes usually describe the personal profile of an individual user, and these are highly domain-specific. These typically include age, gender, occupation, lifestyle choices, and so on. Additionally *derived* attributes can be used, such as differences in the values of basic attributes between pairs of users. Together these make up the "User Profile".

User Behaviour Features. This set covers behaviour patterns in terms of aggregated interaction summaries over time to form a "User Behaviour" feature set. This feature set models the user *activity*, i.e., how frequently a user participates in the system looking for friends or partners, and user *popularity*, i.e. how frequently a user is contacted by other users.

User Interest Features. This set contains keywords from free-text descriptions of "User Interests". This and the former feature set are important to augment the learned model to potentially improve recommendations. The former models the user's behaviour in the system over time, especially their level of reciprocated activity, and the latter contains details of user interests. Constructed feature sets are discussed further in Section 6.1.

3.3 Model Selection

Several different strategies were used to study models with the aim of improving recommendation in this research. A range of *costs* were used to increase the weight of predicting the class of a successful interaction as unsuccessful (a false negative) to assess the impact of trading off precision and recall in the model. We also examined the effect of using different *feature sets*, and the use of separate models for different *genders*. Details on model selection will be discussed further in Section 6.3.

4 Candidate Generation

In real world recommendation applications, efficiently generating candidate lists is important since the aim of a recommendation system is to present the active

users with an actual list of relevant users or items. Existing research on learning models for recommendation often evaluates the constructed model directly on the historical data, but pays little attention to candidate list generation. In this paper, we address this problem by using the learned rule-based model to select candidate users using the rules.

Specifically, to generate candidate lists for recommendation, we first convert a decision tree into decision rules, finding all conditions, i.e., tests on feature values, in each path from the root to each leaf in the decision tree and creating a rule for each path. Rules $r \in R$ have the following format:

$$\text{if } (c_1^a, c_2^a, ...c_n^a) \wedge (c_1^r, c_2^r, ...c_m^r) \wedge (c_1^{ar}, c_2^{ar}, ...c_k^{ar}) \text{ then } t \tag{1}$$

where c_i^a are conditions on the active user u_a, c_j^r are conditions on the recommended user u_r, c_i^{ar} are conditions on the pair (u_a, u_r) and t is the class. A rule defines two groups of users, i.e., a group of active users g^a defined by c_i^a and a group of recommended users g^r defined by c_i^r that also satisfy the conditions on the pairs of users c_i^{ar}, denoting sets of interactions with the corresponding outcome. Rules associated with the positive class indicate a match of g_a with g_r. We define positive rules R_+ as rules that have a positive prediction $+$.

Given the positive rules, to generate a candidate list for an active user u_a^i, we simply select all positive rules R_+^i where u_a^i satisfies the rule conditions. Then we find the candidates u_rs that satisfy the selected rules R_+^i such that the user-candidate pair (u_a, u_r) satisfies the conditions on pairs from the rule:

$$\begin{aligned} &(c_1^a, c_2^a, ...c_n^a) \subseteq f(u_a) \cap \\ &(c_1^r, c_2^r, ...c_m^r) \subseteq f(u_r) \cap \\ &(c_1^{ar}, c_2^{ar}, ...c_k^{ar}) \subseteq f(u_a, u_r) \end{aligned} \tag{2}$$

where $f(u)$ is the set of feature values of user u and $f(u_a, u_r)$ is the set of derived feature values of the pair (u_a, u_r). Evaluation of the model is then done by ranking the generated candidate lists and comparing them against a test set.

5 Ranking

Any system that presents results to a user, ordered by a utility function that the user cares about, is performing a ranking function. Due to the number of candidates in each list generated in a recommender, ranking is usually necessary to order and further select the candidates for the user. Since we aim to predict successful interactions, it is necessary to recommend a candidate to an active user such that they like each other. We can model the probability of this within such a user pair as:

$$P(S_{a,r}, R_{a,r}) = P(S_{a,r})\ P(R_{a,r}|S_{a,r}) \tag{3}$$

where $P(S_{a,r}, R_{a,r})$ is the probability of sending (S) a contact from the active user u_a to a candidate user u_r and receiving a positive response (R) from u_r to u_a,

$P(S_{a,r})$ is the probability of sending a contact from u_a to u_r and $P(R_{a,r}|S_{a,r})$ is the conditional probability of a positive response from u_r to u_a given the active user u_a sent a contact to the candidate user u_r. We can then model the probability of having a successful interaction based on the interactions in the training set as follows:

$$\begin{aligned} P(S_{a,r})\ P(R_{a,r}|S_{a,r}) &= \frac{n(u_a \rightarrow u_r)}{n(u_a \rightarrow all)} \frac{n(u_a \leftarrow u_r)}{n(u_a \rightarrow u_r)} \\ &= \frac{n(u_a \leftarrow u_r)}{n(u_a \rightarrow all)} \end{aligned} \tag{4}$$

where $n()$ is the number of contacts, $\rightarrow$ means the set of contacts in the data sent from the subset of users denoted by the left hand side of the $\rightarrow$ to those in the subset denoted by its right hand side, $\leftarrow$ means the set of replies in the data from the subset of users denoted by the right hand side of the $\leftarrow$ to those in the subset denoted by its left hand side, and *all* is all users in the system.

However, in recommendation, $P(S_{a,r})\ P(R_{a,r}|S_{a,r})$ is not directly evaluable, since the interaction between the active user u_a and the candidate user u_r is to be *predicted* and is usually not in the data. Thus, approximation of these probabilities is required. To do this, we make use of the rules generated from the decision tree.

Since each rule in the learned model from Section 4 defines a group of users that have similar characteristics in terms of attractiveness and taste, we can approximate u_a with lhs (see below). Here lhs denotes the group of users covered by the features $f(u_a)$ from the first part (*l*eft *h*and *s*ide) of a rule in the model, i.e., contact senders. Similarly, rhs is the group of users covered by the features $f(u_r)$ from the second part (*r*ight *h*and *s*ide) of a rule, i.e., receivers. Further, for any user in lhs there must be a user in rhs for which the pair satisfies the features $f(u_a, u_r)$, and vice versa.

Let $S_{a,rhs}$ denote the event of sending from the active user u_a to any member of the group of users rhs. Initially, from Equations 3 and 4, we have:

$$\begin{aligned} P(S_{a,rhs})\ P(R_{a,r}|S_{a,rhs}) &= \frac{n(u_a \rightarrow rhs)}{n(u_a \rightarrow all)} \frac{n(u_a \leftarrow u_r)}{n(u_a \rightarrow rhs)} \\ &= \frac{n(u_a \leftarrow u_r)}{n(u_a \rightarrow all)} \end{aligned} \tag{5}$$

It follows both that $P(S_{a,r})\ P(R_{a,r}|S_{a,r}) = P(S_{a,rhs})\ P(R_{a,r}|S_{a,rhs})$ since they are calculated as the same value, and that $P(S_{a,r})\ P(R_{a,r}|S_{a,r})$ can be represented using the first line of Equation 5.

Now, note that $u_a \leftarrow u_r$ does not appear in the training set. However, because u_a is in the group denoted by the features in lhs, the set $lhs \leftarrow u_r$ can be used to estimate the taste of u_r for users described by lhs, which includes u_a.

We therefore replace $n(u_a \leftarrow u_r)$ with $n(lhs \leftarrow u_r)$, and to maintain consistency we need also to replace $n(u_a \rightarrow rhs)$ with $n(lhs \rightarrow rhs)$. Equation 5 then becomes:

$$P(S_{a,r})\ P(R_{a,r}|S_{a,r}) = \frac{n(u_a \rightarrow rhs)}{n(u_a \rightarrow all)} \frac{n(lhs \leftarrow u_r)}{n(lhs \rightarrow rhs)} \tag{6}$$

which is now an operational version of our ranking function. Ranking of candidates for recommendation to the active user is then sorting them on their score as defined in Equation 6. Note that this ranking function does not depend on the use of rules to represent successful people-to-people interactions; it could be applied given alternative methods of grouping similar senders and receivers.

6 Experimental Evaluation

We evaluated the proposed approach in a realistic setting using a dataset from a large commercial online dating site. We compared our learning framework to a CF-based recommender system using essentially the Amazon approach [1]. Data was pre-processed in Oracle 10, decision trees were learned using See5[2] and converted to rules in SQL using Perl, and the remaining algorithms were implemented in Matlab. Tree learning requires about 15 minutes per tree. Candidate generation requires about 2 hours and ranking takes less than half an hour with Enterprise 64-bit Windows Server 2003, 2 processors of Intel(R) Xeron(R) CPU x7462@2.66GHz and 32GB RAM.

6.1 Experimental Setup

We selected three periods for training and testing that cover the data as shown in Table 1. We selected each test period to be 1 or 2 months later than the corresponding training period. Therefore three training periods and three test periods were created as shown in Table 1.

Table 1. Evaluation Period

Period	Training	Test
P1	Oct–Dec 2008	Feb 2009
P2	Jan–Mar 2009	Apr 2009
P3	Oct–Dec 2009	First week of Mar 2010

The datasets collected contained interactions between users. Specifically, the data contained records, each of which represents a contact by a tuple containing the identity of the contact's sender, the identity of the contact's receiver and an indicator showing whether the interaction was successful (with a positive response from the receiver to the sender) or unsuccessful (with a negative

[2] Available from `www.rulequest.com`

response). Contacts with no reply are excluded. Importantly, division into training and test sets was done using a *time-based* holdout method, i.e., training set examples were taken from a contiguous time period *before* the period supplying the corresponding test set, and they do not overlap. Datasets were also created using the three variant feature sets as described above in Section 3.2. We denote these as follows: F1 – User Profile; F2 – User Profile and Behaviour; and F3 – User Profile, Behaviour and Interests. Finally, in online dating applications it is also necessary to consider the gender of the active user, as this affects interaction behaviour. We denote this as follows: M – contact sender is male; and F – contact sender is female.

Datasets were created by varying the above three dimensions: three time periods, three types of feature set, and two genders, giving a total of $3 \times 3 \times 2 = 18$ possible training and test set configurations. Although the exact data set size is confidential, the combined data sets contain on the order of 10^5 active users and 10^6 interactions. Note that this is a large data set, and also that the interaction graph is very sparse, making prediction of interactions a difficult problem.

Evaluation was carried out in two stages. In the first stage models were built using training datasets from the first two time periods (P1 and P2). The learned models were tuned using the approaches described in Section 3.3 using the "test" sets from P1 and P2.

The second stage simulated running the model-based recommender on historical data in an attempt to obtain a realistic evaluation of *recommendation* performance, not simply classification. Having selected the best learning setup from the tuning in stage one we then trained another set of models using the training set from the third time period (P3), then generated candidates for active users in the corresponding third test set and ranked them based on their scores, as described in Sections 4 and 5. The results for these recommendations were compared both to the baseline rate of successful interactions in the third time period, and that of item-based CF recommendation, using evaluation metrics defined in the next section.

6.2 Evaluation Metrics

The evaluation metrics used in this research are defined in Table 2. It is important to reflect on the goals of learning a social recommender system when considering evaluation metrics.

First, it is not sufficient to run a set of interactions from a historical data set through the learned model to obtain predictive scores. This is unrealistic, since the model must actively generate and rank those predictions for each, not passively classify them. Second, the distribution of interactions from a real application in which the recommender will operate is necessarily different from historical data in which interactions are generated by some other method such as search. Finally, it is necessary to consider trade-offs in the relative importance of making fewer, higher quality recommendations against making more recommendations of which many may not lead to successful interactions.

Taking these considerations into account, the main objective in this research was to improve the *success rate* (Precision) by using the learned recommender over the default obtained by users themselves. The effectiveness of the learners are evaluated using standard precision-recall graphs.

Table 2. Evaluation Metrics

Name	Description
Success Rate (Precision)	the proportion of the true predicted successful interactions to all predicted successful interactions
Recall	the proportion of the true predicted successful interactions to all true successful interactions
Default Success Rate	the proportion of successful interactions to all interactions in the dataset initiated by users who receive recommendations
Success Rate Improvement (SRI)	the ratio of success rate to default success rate

6.3 Experimental Results for Model Selection

In the following we give results for stage one of the evaluation, feature and model selection, where the 12 training and test set configurations are designated by the identifier *Period-Features-Gender*, where *Period* is either P1 or P2, *Features* is either F1, F2 or F3, and *Gender* is either M or F.

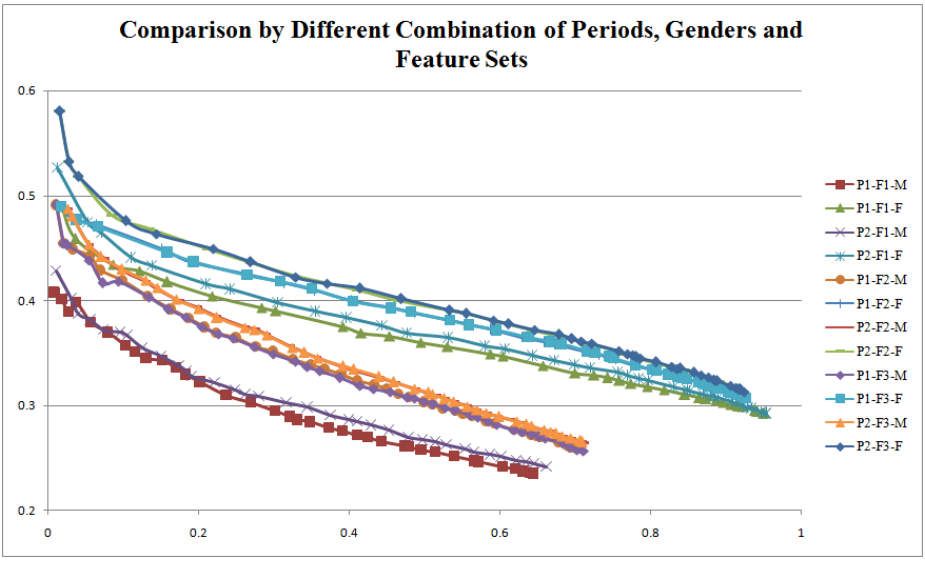


Fig. 2. Comparison by Different Costs. The x-axis is the recall ranging from 0 to 1 and y-axis the precision ranging from 0 to 1, with cost varying from 1 to 5 in steps of 0.1.

Results by Different Costs. A range of cost values from 1 to 5 were used to weight the cost of false negatives (successful contacts incorrectly predicted by the model). Specifically, 41 cost values ranging from 1 to 5 with step size 0.1 were generated, which were then used to train 41 classifiers for each dataset. Precision and recall for each data set and cost setting are plotted in Figure 2. In Figures 3 and 4, precision (respectively recall) is shown as a brown (resp. blue) solid curve, where the x-axis is the cost ranging from 1 to 5 and the y-axis is precision (resp. recall) ranging from 0 to 1. The figures demonstrate that by increasing the cost, the precision decreases while recall increases. To achieve a recall-precision ratio for the recommendation system a cost can be set on the trade-off curve between precision and recall as shown on Figure 4.

Results by Different Feature Sets. The effect of different feature sets is shown in Figure 2, where feature set F2 and feature set F3 give much better performance than feature set F1 for all combinations of periods and genders. Since F1 only contains user profile based features while F2 and F3 include both user profile based and user behaviour-based features, the results confirm that user behaviour patterns are very important for predicting interaction outcome.

Results by Different Periods. To test reliability, the trained models were tested on two different periods as shown in Table 1. Figure 2 clearly shows that very similar results were obtained for those two periods by training and testing on 6 datasets over 41 costs. This indicates that the performance of the learning methods remain almost the same over the two different periods, which demonstrates the stability of the learning method.

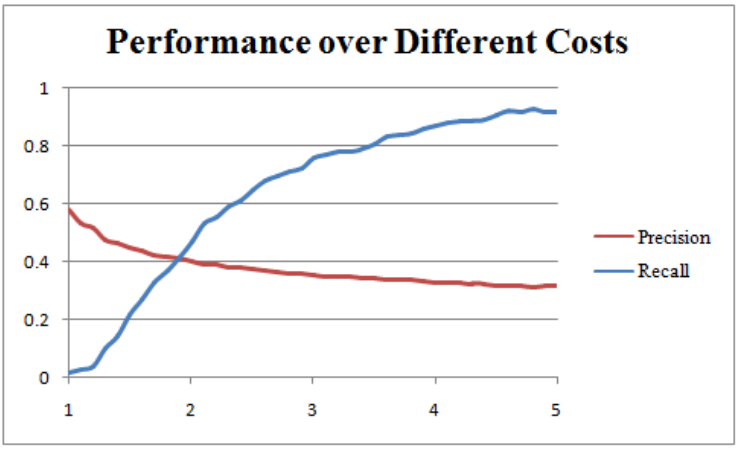


Fig. 3. Performance over different costs. Shown is the typical relationship between precision and recall as costs vary, where the x-axis is the cost ranging from 1 to 5 in steps of 0.1 and y-axis the performance (precision and recall) ranging from 0 to 1.

Results by Different Gender. As shown in Figure 2, in all 6 datasets classifiers for female users perform much better than for males with respect to precision and recall. Since the behaviour pattern of females are different from that of males in sending far fewer contacts with a higher default success rate, by separately training different models based on gender, models with better performance can be created for female senders.

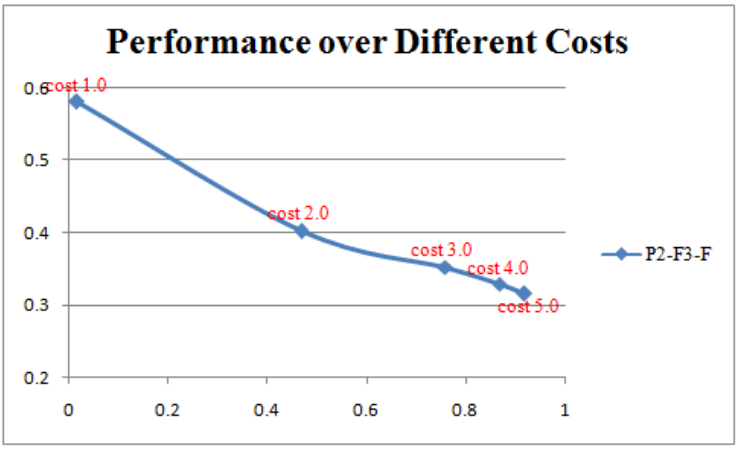


Fig. 4. Performance over different costs. The figure shows performance over different costs for Period 2, Feature Set 3 and female senders, where the x-axis is the recall ranging from 0 to 1 and y-axis the precision ranging from 0 to 1, showing the points for cost 1.0, 2,0, ..., 5.0.

6.4 Experimental Results for Recommendation

Based on the feature and model selection results, we selected the model with the following configuration: feature set F2, cost 3.5 for the male tree and cost 2 for the female tree, as giving a reasonable trade off between precision and recall (where the precision and recall lines cross in Figure 3). We then trained two decision trees on training period P3 for females and males respectively and constructed recommendations. These were then tested on test period P3.

Since recommendation is reliably and accurately implemented by standard item-based CF [1], we compared our model-based recommender with a version of this standard CF for social recommendation. The method is as follows. Recommend u_r to active user u_a when u_r has had successful interactions with a user u'_a with similar *taste* to u_a since they have in common a set of successful interactions with the same users. The results are shown in Table 3.

As shown in Table 3, our method achieved much higher precision than the default with SRI 2.14, 2.07, 1.95, 1.71 and 1.56 for candidates ranked in the Top 5, 10, 20 and 100, and all candidates, respectively, which shows that our method works very well with people to people recommendation. However, the

Table 3. Tree Recommender Results after Ranking

	Tree Recommender			Default	CF
	Recall	Success Rate	SRI	Success Rate	Success Rate
Top 5	0.64	42.7	2.14		19.8
Top 10	1.05	41.3	2.07		16.9
Top 20	1.82	38.9	1.95	20	15.3
Top 100	5.64	34.2	1.71		12.6
All	25.01	31.2	1.56		12.7

standard CF method has clearly lower precision than our method. It is actually worse than the default in terms of precision, which shows that standard CF may not be a good choice for people to people recommendation in social networking sites. Interestingly, precision results for our method on the Top 20 mean that, on average, nearly 8 out of 20 recommendations for the active user would result in successful interactions if replicated in the real world setting.

7 Conclusion

We have proposed a machine learning framework to make use of both profile information and behaviour information for people to people recommendation in social networking sites by the integration of feature construction, classifier learning, model selection, candidate generation and probabilistic modelling for ranking. We argued that it is important to evaluate machine learning for model-based social recommender systems realistically, since the model must actively generate and rank recommendations, not passively classify them. Our method is simple to train, is not restricted to a particular classifier learning algorithm, and gives improved results on datasets extracted from a large online commercial dating site. Compared with a benchmark CF recommender system, our method demonstrates higher precision and competitive recall. For future work we aim to further investigate the combination of profile-based features and behaviour-based features in hybrid models and CF-based frameworks for social recommendation.

Acknowledgements. This project is funded by the Smart Services CRC under the Australian Government's Cooperative Research Centre program. We would also like to thank our industry partners for providing the datasets.

References

1. Linden, G., Smith, B., York, J.: Amazon.com Recommendations: Item-to-Item Collaborative Filtering. IEEE Transactions on Internet Computing 7(1), 76–80 (2003)
2. Carmel, D., Zwerdling, N., Guy, I., Ofek-Koifman, S., Har'el, N., Ronen, I., Uziel, E., Yogev, S., Chernov, S.: Personalized Social Search Based on the User's Social Network. In: Proceeding of the 18th ACM Conference on Information and Knowledge Management, pp. 1227–1236. ACM, New York (2009)

3. Burke, R.: Hybrid Recommender Systems: Survey and Experiments. User Modeling and User-Adapted Interaction 12(4), 331–370 (2002)
4. Cai, X., Bain, M., Krzywicki, A., Wobcke, W., Kim, Y., Compton, P., Mahidadia, A.: Learning Collaborative Filtering and Its Application to People to People Recommendation in Social Networks. In: Proceedings of the 10th IEEE International Conference on Data Mining, pp. 743–748 (2010)
5. Cai, X., Bain, M., Krzywicki, A., Wobcke, W., Kim, Y.S., Compton, P., Mahidadia, A.: Collaborative Filtering for People to People Recommendation in Social Networks. In: Li, J. (ed.) AI 2010. LNCS, vol. 6464, pp. 476–485. Springer, Heidelberg (2010)
6. Krzywicki, A., Wobcke, W., Cai, X., Mahidadia, A., Bain, M., Compton, P., Kim, Y.S.: Interaction-Based Collaborative Filtering Methods for Recommendation in Online Dating. In: Chen, L., Triantafillou, P., Suel, T. (eds.) WISE 2010. LNCS, vol. 6488, pp. 342–356. Springer, Heidelberg (2010)
7. Breese, J.S., Heckerman, D., Kadie, C.: Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In: Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, pp. 43–52 (1998)
8. Shardanand, U., Maes, P.: Social Information Filtering: Algorithms for Automating "Word of Mouth". In: Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI 1995), pp. 210–217 (1995)
9. Hofmann, T.: Collaborative Filtering via Gaussian Probabilistic Latent Semantic Analysis. In: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 259–266 (2003)
10. Pavlov, D., Pennock, D.M.: A Maximum Entropy Approach to Collaborative Filtering in Dynamic, Sparse, High-Dimensional Domains. In: Neural Information Processing Systems, pp. 1441–1448 (2002)
11. Ungar, L., Foster, D.: Clustering Methods for Collaborative Filtering. In: Proceedings of the AAAI 1998 Workshop on Recommender Systems, pp. 112–125 (1998)
12. Pazzani, M.J., Billsus, D.: Content-Based Recommendation Systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) Adaptive Web 2007. LNCS, vol. 4321, pp. 325–341. Springer, Heidelberg (2007)

Microgroup Mining on TSina via Network Structure and User Attribute

Xiaobing Xiong[1,2], Xiang Niu[2], Gang Zhou[1], Ke Xu[2], and Yongzhong Huang[1]

[1] National Digital Switching System Engineering and Technological Research Center, China
[2] State Key Lab of Software Development Environment, Beihang University, China
{bingxiaoxiong,niuxiang,gzhougzhou,kexu999,yzhongyzhongh}@gmail.com

Abstract. In this paper, we focus on the problem of community detection on TSina: the most popular microblogging network in China. By characterizing the structure and content of microgroup (community) on TSina in detail, we reveal that different from ordinary social networks, the degree assortativity coefficients are negative on most microgroups. In addition, we find that users from the same microgroup likely exhibit some similar attributes (e.g., sharing many followers, tags and topics). Inspired by these new findings, we propose a united method for microgroup detection without losing the information of link structure and user attribute. First, the link direction is converted to the weight by giving higher value to the more surprising link, while attribute similarity between two users is measured by the Jaccard coefficient of common features like *followers*, *tags*, and *topics*. Then, above two factors are uniformly converted to the edge weight of a newly generated network. Finally, many frequently used community detection algorithms that support weighted network would be employed. Extensive experiments on real social networks show that the factors of link structure and user attribute play almost equally important roles in microgroup detection on TSina. Our newly proposed method significantly outperforms the traditional methods with average accuracy being improved by 25%, and the number of unrecognized users decreasing by about 75%.

Keywords: Microblogging, Microgroup Mining, Community Detection, United Method.

1 Introduction

In recent years, the microblogging system has emerged as a novel social media platform and provides a new means of communication, with which users can broadcast brief updates about any things happening in their daily life or work activities, such as what they are doing, watching, or thinking about. As we all know, the most famous microblogging system in the world is Twitter, while Sina microblogging (TSina) is the most famous one in China, which began in Aug. 2009 and has gained more than 100 million users until Apr. 2011.

J. Tang et al. (Eds.): ADMA 2011, Part II, LNAI 7121, pp. 138–151, 2011.

TSina introduced a new application named *Microgroup* in Nov. 2010. A microgroup is usually a group of users who have close connections or share similar interests. The number of microgroups is growing rapidly, and there are more than 350 thousand microgroups which contain more than 20 million users by Apr. 2011. The emergence of microgroup provides us a good opportunity to do research on user classification, which is often called as *community detection* in network science. Community detection within real-world networks, such as OSNs, Internet, and biological networks, is a problem of considerable practical interest and has received a great deal of attention [1] [2] [3] [4] [5] [6] [7]. Thus, we believe that community detection on microblogging systems (e.g., Twitter, TSina) is also very worthy of investigation in depth, however, little work has been done so far.

In this paper, TSina is chosen as our experiment platform for community detection on microblogging systems. As a community (i.e., microgroup) is a group of users not only with close connections, but also with similar interest on TSina, we focus on determining community memberships by using link relationship and user attribute simultaneously, and present a new united community detection method considering both factors to group TSina users with higher accuracy.

The rest of this paper is organized as follows. In Section 2 we give an overview of the related work. Section 3 describes the details of our data set. In Section 4 we characterize our sample microgroups in detail mainly from aspects of network structure and user attribute, then extract some useful findings for microgroup detection. A united method for microgroup detection on TSina is proposed in Section 5. In Section 6 we apply the newly proposed method on several real-world networks and compare their outcomes against some traditional algorithms. Finally, in Section 7, we conclude this paper briefly.

2 Related Work

Most previous community detection approaches are based on structural features (e.g., links), and a community is usually defined as a group of vertices such that there is a higher density of edges between nodes of the same group and a comparatively lower density between different groups [8] [9]. Then, an objective function named modularity degree is often used to capture the above intuition of a community [4]. As the objective is typically NP-hard to optimize, many algorithms, including spectral partitioning [10], hierarchical clustering, heuristics and approximation solutions [11], have been extensively studied. In recent years, many researches have yet been done on community detection. Gregory et al. proposed an algorithm for finding overlapping community structure in very large networks [12]. Stanoev et al. presented a novel algorithm for community detection that combines network structure with processes that support creation and/or evolution of communities [13]. Yang et al. presented a probabilistic model for community detection that aims to model both incoming links and outgoing links simultaneously and differentially on directed networks [14].

However, several previous works have found that neither link structure nor user content is sufficient to determine the community memberships, while combing

link with content usually achieves better performance [15] [16]. For example, Erosheva et al. combined LDA with LDA-Link for network analysis [17]. Yang et al. proposed a discriminative model for combining the link and content analysis for community detection from networked data, such as paper citation networks [18]. Other approaches that exploit topic models for community detection include [19] and [20]. However, these previous researches are very similar to the problem of text classification, and only act on the networks with nodes denoting text pages (e.g., blog pages, wikipedia pages, and published papers), but not on the social networks with nodes denoting users. Hence, it is a challenging but rewarding task for us to cluster homogeneous users with close connections and similar interests into communities simultaneously on microblogging systems.

3 Data Set

On TSina, microgroup was introduced in Nov. 2010 and attracted more than 20 million users until Apr. 2011. As most microgroups are formed by users with similar interests, therefore, in order to analyze the explicit and internal characteristics of microgroups, we chose 34 microgroups, covering almost all kinds of microgroups on TSina, for analysis. We crawled the link structure and content information of these microgroups in a short time interval from March 6th to March 21st, 2011. Here, user ID, name, location, gender, verified flag, published tweets, followers list, followings list, tags list, and topics list were crawled. The final total number of users we gathered is about 200,000 from 34 microgroups.

Each microgroup is a sample of community from TSina network, and can be seen as a directed graph, in which vertex means microgroup member, and edge indicates the "follow" relationship That is, a directed edge from A to B means that user A follows B, then we say that A is a follower of B, and B is a following of A. When characterizing our sample microgroups, we consider only the edges among members from the same microgroup but ignore the links between microgroups, which yields 34 separate graphs.

4 Characterizing Microgroup

Before delving into the challenging problem of microgroup detection on TSina, we run a batch of analysis on characterizing sample microgroups and expect to extract some findings that are conducive to exploit a more efficient method on microgroup detection.

We begin our analysis of microgroup with the following questions: Whether the two linked users are similar to each other on some attributes? What factors are more significant in driving users to join the same microgroup, and whether there are some similarity among the members from the same microgroup? Therefore, we first summarize the basic information of sample microgroups, then characterize them in detail from both aspects of network structure and user attribute in this section.

4.1 Basic Analysis

We summarize the basic network properties of our sample microgroups, and the average results are shown in Table 1, from which we can know that those networks are very sparse with low densities (average is 0.0077). Furthermore, about one in five members are isolated and have no link relationship with others, and they join microgroups just for sharing information, but not making friends. Hence, isolated users can not be classified only from the view of link structure, however, their attributes may help to explain why isolated users choose to join the microgroup, and they may be similar to other members on some attributes.

Table 1. Basic statistical characterizations of sample microgroup networks

	Node#	dEdge#	BiE(%)	MD	IsoN(%)	Density
Avg	1797	12587	44.82	5.9	19.85	0.0077

Node#: number of nodes; dEdge#: number of directed edges; BiE(%): ratio of bi-way edges; MD: mean degree; IsoN(%): ratio of isolated nodes; Density: network density.

The ratio of bi-edges is often used to measure the reciprocity of a social network, and many previous studies have reported high level of reciprocity on some social networks: 68% of user pairs with any link between them are connected bi-way on Flickr [21] and 84% on Yahoo!360 [22]. But there is also research showing a low level of reciprocity on Twitter: 77.9% of user pairs are one-way, and only 22.1% have reciprocal relationships between them [23]. Low reciprocity on global Twitter network may be caused by the fact that top users with large number of followers in Twitter are mostly celebrities and mass media, but most of them do not follow their followers back. On sample microgroups, the level of reciprocity is moderate: 44.8% of user pairs are bi-way, which is larger than Twitter but lower than Flickr. As similar with Twitter, most celebrities and mass media are followed by a large number of users but do not follow them back, what's more, few celebrities and media would like to join a microgroup on TSina.

4.2 Assortativity Coefficient

Similarity breeding connection is a principle that structures many kinds of network ties, including friendship, reference, support, coauthor and so on. We call the principle as homophily, an important criterion to quantify the tendency for users to be friends with others who have similar characteristics, which can be measured by assortativity coefficient. Here, in order to analyze the similarity degree among users from the same microgroup, we study the assortativity coefficients by different user attributes, including *degree*, *gender*, *location*, *VFlag*, *tweets count*, *retweets count*, and *comments count*.

Newman proposed a method to compute the degree assortativity coefficient on directed networks [24], i.e., Eq. 26, which will be employed in this paper.

When calculating the assortativity coefficients by other continuous attributes like *tweets count*, *retweets count*, and *comments count*, we measure with the standard Pearson correlation coefficient described in Eq. 21 of [24]. However, when computing assortativity coefficient by discrete or enumerative attributes like *location*, a general measure of scalar assortativity relative to a categorical variable is given by

$$r = \frac{tr(e) - \|e^2\|}{1 - \|e^2\|} \tag{1}$$

where $e = E/\|E\|$ is the normalized mixing matrix, the elements E_{ij} of E give the number of edges in the network that connect from a node of type i (e.g., users from "*Beijing*") to type j (e.g., users from "*Shanghai*").

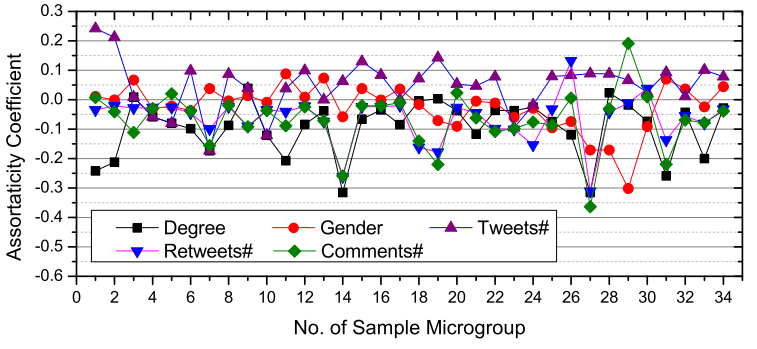


Fig. 1. Assortativity coefficients by user attributes (*Degree*, *Gender*, *Tweets#*, *Retweets#*, *Comments#*) on each sample microgroup

The results of assortativity coefficients based on some user attributes are described in Fig. 1, from which we can know that although many ordinary social networks tend to be positively assortative with respect to degree, for most of our sample microgroups, the degree assortativities (denoted by ■ in Fig. 1) are negative, or weakly positive with low values, which implies that most users with few followers tend to follow others with many followers, like celebrities. Besides, the two users with "follow" relationship are not so similar on the attribute of degree in most cases.

To some extent, the number of tweets can be used to measure a user's activity on TSina. From our observations, we find that many assortativity coefficients by *tweets#* (denoted by ▲ in Fig. 1) are positive, which is distinct from degree assortativity and indicates that many users tend to follow others with similar number of published tweets. In addition, the number of retweets and comments are important measurements for user's popularity on TSina, and assortativity coefficients by *retweets#* (denoted by ▼ in Fig. 1) imply a similar conclusion with degree assortativity. That is, most ordinary users tend to follow others with high popularity. Assortativity coefficients by *comments#* (denoted by ♦ in Fig. 1) show consistent result with *retweets#*.

Assortativity coefficients by *gender* (denoted by • in Fig. 1) are very close to 0 on most sample microgroups, which reveals that there is no obvious tendency for user to follow others with the opposite or same gender. As most members on microgroups are non-verified, and some microgroups like collegiate groups are localized in the same city, assortativity coefficients based on *VFlag* and *location* will be ignored in this paper.

By analyzing the assortativity coefficients based on different user attributes, we find that the two ends with link relationship are different from each other on many attributes like degree and popularity, which differs significantly from other ordinary social networks. For this reason, when classifying users to different microgroups, link structure is not satisfactory in determining accurately the community memberships.

4.3 Density Difference

From common definition, a community is a subset of users within which the network connections are dense, but between which they are sparser. So the density of our sample microgroups should be higher than the random sampling groups. In this paper, the density of a directed network $G(n, m)$ is defined as $d = \frac{m}{n \times (n-1)}$, and m, n are respectively the total number of directed edges and nodes in the network.

In Fig. 2, we show the densities of our sample microgroups, which are presented by the ratios compared to the random case. From Fig. 2 we know that the densities of our sample microgroups vary a lot, and many have much more compact structures than random sampling groups, like microgroup 12, 13, 25, 28, and 32. However, there are also some microgroups with density close to the random case, such as microgroup 3, 1, 21, 20, 18, and 9, besides, the density of microgroup 3 is less than the random case. The existence of these microgroups with low densities breaks the traditional definition of community, which focuses only on the compactness of the link structure. Thus, we further enhance our conclusion that it is not sufficient to identify microgroups by only considering link structure on TSina.

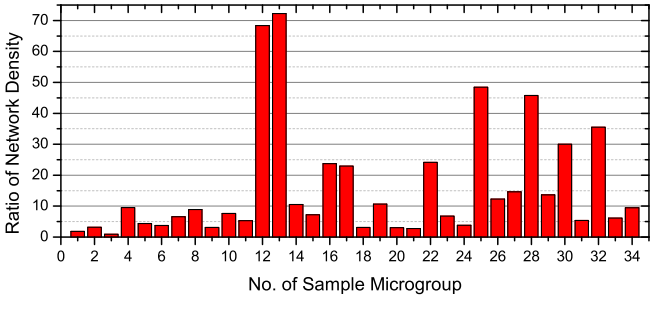


Fig. 2. Density of sample microgroups compared to random case

4.4 Attribute Similarity

In many social networks, attribute similarity is a basic principle for users to gather together in the same community. In order to dig out the distinctive characteristics of members from the same microgroup, we analyze the average similarity among users based on their common *followers*, *followings*, *tags*, and *topics*, respectively, on each sample microgroup, then compare the observation results with a series of random sampling groups. By comparison, we try to reveal what characteristics are remarkable for us to label users from the same microgroup, which can help to identify communities on TSina.

In this paper, Jaccard coefficient, a commonly used similarity metric in information retrieval [25], is used to measure the attribute similarity among users, that is, measure the probability that both x and y have common feature f. If the "feature" here is taken as *followers*, *followings*, *tags*, or *topics*, the similarity between user x and y on each feature can be calculated as follows:

$$s_f(x,y) = \frac{|\Gamma_f(x) \cap \Gamma_f(y)|}{|\Gamma_f(x) \cup \Gamma_f(y)|} \tag{2}$$

where $\Gamma_f(x)$ is the set of feature f for user x, such as the set of followers of user x, and $|\Gamma|$ is the number of elements in Γ. Then, on each microgroup, the average similarity among users by different features is:

$$avg\_s_f(G) = \frac{1}{N \times (N-1)} \sum_{x,y \in \Gamma(G)} s_f(x,y) \tag{3}$$

where $\Gamma(G)$ is the set of users on microgroup G, and N is the size of $\Gamma(G)$.

We have mentioned that a microgroup should be a group of users with similar interest measured by users' features. In order to estimate what features are more prominent, we construct a series of random sampling groups with different number of users from all crawled users. The average similarity between random sampling groups are shown in Fig. 3, from which we see that, with the size increase of random sampling groups, the average similarity measured by Jaccard coefficients are very stable on all considered features. Thus, our approach of random sampling is nearly unbiased, and the results of random cases can be seen as baseline.

The results of average similarity by different features on each sample microgroup are shown in Fig. 4, which are expressed by the ratios compared to the random sampling groups described in Fig. 3. We call the ratio as significance degree in this paper. From Fig. 4, we find that the feature of *followers* is the most significant and the average ratio is much higher (average: 22.6) than the other three features, in which *followings* is the least prominent and most ratios are close to one (average: 1.4), i.e., the average similarity by *followings* on our sample microgroups is very close to random sampling groups. Then, we conclude that users with more common followers are more likely to be similar and in the same microgroup, but this is not suitable for *followings*. The non-significance of the feature of *followings* may be induced by the truth that many users choose to

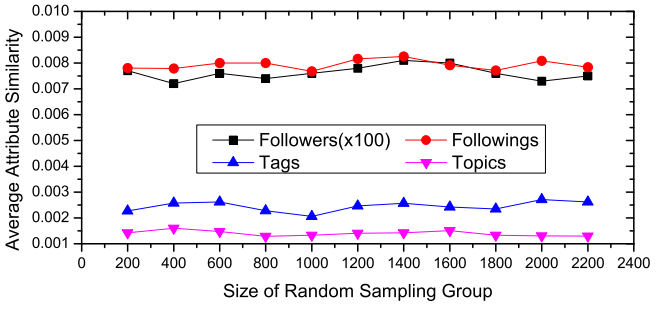


Fig. 3. Average attribute similarity measured by Jaccard coefficient on random sampling groups. The considered attributes are *Followers*, *Followings*, *Tags* and *Topics*.

follow some common celebrities simultaneously, however, the action of celebrities following expresses little about user's interest. Intuitively, *tags* and *topics* are very important indicators of a user's interest, but our results from Fig. 4 show that the two features are not so significant as expected, with the average ratios of Jaccard coefficients about 3.8 and 3.6, respectively. Thus, we know that the feature of *followers* is the most significant for microgroup detection, then followed by *tags* and *topics*, but the feature of *followings* is nearly indistinctive, and we will ignore it when identifying microgroups.

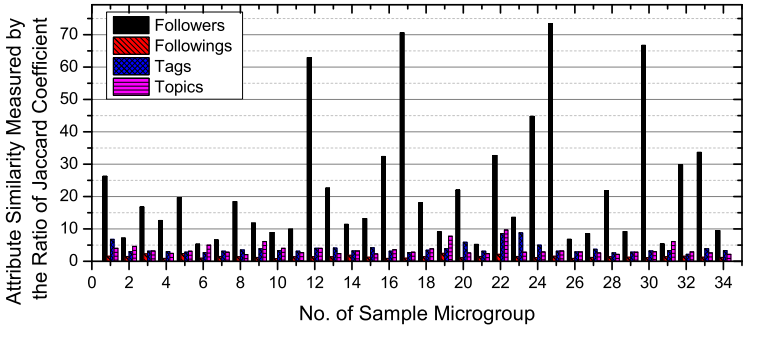


Fig. 4. Attribute similarity of sample microgroups compared to random case

5 United Microgroup Detection Method

Microblogging users construct an unweighted and directed network, and the problem of community detection on directed networks has been well studied. However, as aforementioned, neither link structure nor user attribute is satisfactory in identifying the community memberships on TSina: the link relationship is usually sparse on microblogging networks, and the two ends with link relationship are dissimilarity on many attributes. In addition, the irrelevant user

attributes may mislead the result of microgroup detection. Hence, in order to solve the problem of community detection on TSina with a higher performance, we propose a united method without losing the information of link structure and user content. Here, user content mainly includes discussed topics and attributes like followers list and tags list. Then, there are two "links" between users, one is the explicit "follow" relationship, and the other is the implicit attribute similarity between the two ends. With our new method, we uniformly convert the link structure and attribute similarity to the edge weight of a newly generated network, then many well known community detection algorithms that support undirected weighted networks would be employed.

Consider two nodes i and j on TSina network. The edge weight between the two nodes can be calculated as follows:

$$W_{ij} = \alpha L_{ij}^{'} + \beta S_{ij}^{'} \tag{4}$$

where α, β $(\alpha + \beta = 1)$ are respectively the weight values of link structure and attribute similarity in microgroup detection. $L_{ij}^{'}$ is the normalized edge weight converted from the information of direction, while $S_{ij}^{'}$ is the normalized attribute similarity between the two users.

In our united approach, instead of simply ignoring directional information, we use the method proposed by Youngdo Kim et al. to convert the information of link direction to the weight of a new undirected link [26]. The key idea is to give higher weight to the more *surprising* link, and the *surprising* degree is measured by the probability of the link. Let's consider a link directing from node i to j, and the probability of this link, when the links are assigned randomly while keeping the degree of each node, is

$$p_{ij} = \frac{k_i^{out} k_j^{in}/2m}{k_i^{out} k_j^{in}/2m + k_j^{out} k_i^{in}/2m} \tag{5}$$

where $k_i^{out} = \sum_j A_{ij}$ and $k_j^{in} = \sum_i A_{ij}$ are respectively the outgoing and incoming degree of node i and j, where $A_{ij} = 1$ if there is a link from i to j, and 0 otherwise, and m is the total number of links defined as $m = \sum_i \sum_j A_{ij}$.

As smaller p_{ij} indicates stronger relatedness for the direction from node i to j, then the weight of the link between node i and j is defined as

$$L_{ij} = A_{ij}(1 - p_{ij}) + A_{ji}(1 - p_{ji}) \tag{6}$$

$\{L_{ij}\}$ is an undirected weighted network transferred from the original directed weighted network $\{A_{ij}\}$.

Attribute similarity between two users is measured by the Jaccard coefficient of user features stated above, from which we know that the average similarity on the feature of *followers* is the most prominent, followed by *tags* and *topics*. However, similarity on the feature of *followings* is nearly the same with random case, and will be ignored when measuring the similarity between two users. Thus, we define the attribute similarity between node i and j as

$$S_{ij} = \eta_1 s_{ij}^{fol} + \eta_2 s_{ij}^{tag} + \eta_3 s_{ij}^{top} \tag{7}$$

where s_{ij}^{fol}, s_{ij}^{tag}, and s_{ij}^{top} are respectively the normalized similarity by the feature of *followers*, *tags*, and *topics* between node i and j. $\eta_i (i = 1, 2, 3)$ are their weight values which indicate the significance in measuring similarity. Here, η_i will be determined by the significance degrees of the features mentioned in above subsection, then we empirically assign $\eta_1 : \eta_2 : \eta_3 = 22.6 : 3.8 : 3.6$.

By applying our new approach introduced above, the information of link structure $\{L_{ij}\}$ and attribute similarity $\{S_{ij}\}$ can be unitedly converted to the edge weight of a new network $\{W_{ij}\}$ which is undirected and weighted. Then, many well developed community detection methods for undirected weighted networks can be applied to microgroup detection on TSina, without losing considerations of link structure and user attribute.

In this paper, three well known algorithms of CNM [8], Informap [27], and OSLOM [28] are employed to identify communities on re-modeled networks. CNM is a fast greedy modularity optimization algorithm proposed by Clauset, Newman, and Moore, and its key idea is: starting from a set of isolated nodes, the links of the original network are iteratively added to produce the largest possible increase of the modularity degree at each step. Infomap is a new information theoretic approach proposed by Rosvall and Bergstrom that reveals community structure in weighted networks, and the key is to decompose a network into modules by optimally compressing a description of information flows on the network. *Order Statistics Local Optimization Method*(OSLOM) is the first method capable of detecting communities in networks accounting for edge directions, edge weights, overlapping communities. The method is based on the local optimization of a fitness function expressing the statistical significance of communities with respect to random fluctuations, which is estimated with tools of Extreme and Order Statistics.

6 Experiments and Results

In this section, we validate the effectiveness and efficiency of our newly proposed microgroup detection method by applying it to several real-world networks from TSina. Since almost all of the frequently-used test networks (e.g. Zachary [29], Football [3], Dolphins [30]) in community detection have no information about user attribute, we collect four collegiate social networks from TSina as our test cases, and each of them is composed by some microgroups. The detecting results by our method will be compared with the real community structures of these test networks. Table 2 shows the basic information of our four test networks. For example, TU is a social network of TSina users from TSinghua University, one of the most famous universities in China, which is composed by eight microgroups, namely, Fine Arts Institute, Architecture Institute, EMBA Club, Chinese Institute, Law Institute, TSinahua Library, Electronic Engineering Institute, and Industrial Engineering Institute.

By applying our method, we convert the link relationship and attribute similarity between two users to the edge weight of a newly generated network with Eq. 4. As the weight values of α and β respectively indicate the importance of

Table 2. Basic information of test networks

Network	College	Nodes	Edges	Microgroups
TU	TSinghua University	2102	10792	8
HUST	Huazhong University of Science and Technology	792	4355	9
BNU	Beijing Normal University	670	7923	10
RUC	Renmin University of China	569	3727	5

the link and attribute in microgroup detection, we will gradually vary the two values (α, β) to seek a better assignment in our experiments. Then, the community detection algorithms support for undirected weighted network can be employed on the newly generated network.

In order to estimate the performance of community detection methods, the best way is to compare their outcomes with real community partition on artificial or real-world networks, and this can be done using *similarity measures* as mentioned in [31]. Newman used *fraction of correctly identified nodes* to measure the performance of community detection algorithm in [3]. However, it does not work well in some cases, then some other measurements have been proposed. The measurement of *Normalized Mutual Information*(*NMI*) borrowed from information theory has been proved to be reliable [2], and will be adopted to estimate the performance of our untied method. *NMI* takes the maximum value of 1 if the detected partition is completely consistent with the real case, whereas it has an expected value of 0 if the two partitions are totally independent.

In Fig. 5, we show the experimental results of our method on four test networks. In order to reveal the performance of our method under different parameters, the weight value of link structure (α in Eq. 4, $\beta = 1 - \alpha$) is tuned from 0 to 1 with 0.1 as the step size, and $\alpha = 0$ is the case of conventional clustering method considering only user attribute, $\alpha = 1$ means the case of traditional community detection method based only on link structure. On each test social network, the tendencies of tree curves respectively for CNM, OSLOM, and Infomap are very consistent. The performance of three methods are sensitive to α, and increase with the growth of α, then achieve the highest level when α is about 0.5. After that, the performances decrease with the growth of α. In addition, by comparing the three curves, we can know that the methods of CNM and Infomap always achieve better results than OSLOM. Thus, we can conclude that, no matter what algorithms you choose, link structure and user attribute play almost equally important roles for microgroup detection, and our united method considering both aspects allows us to get a better performance than the traditional algorithms based on either link relationship or attribute similarity.

Finally, Fig. 6 shows the accuracy improvement of our united method compared with the method based only on link structure, taking CNM algorithm as an example. On four test social networks, the average accuracy improvements indicated by left pillars in Fig. 6 is about 25%, and the biggest is 37% on TU network. Furthermore, as most microgroups on TSina are very sparse and many isolated users cannot be clustered to any community using the methods based only on link structure. Fortunately, using our method, more users can be

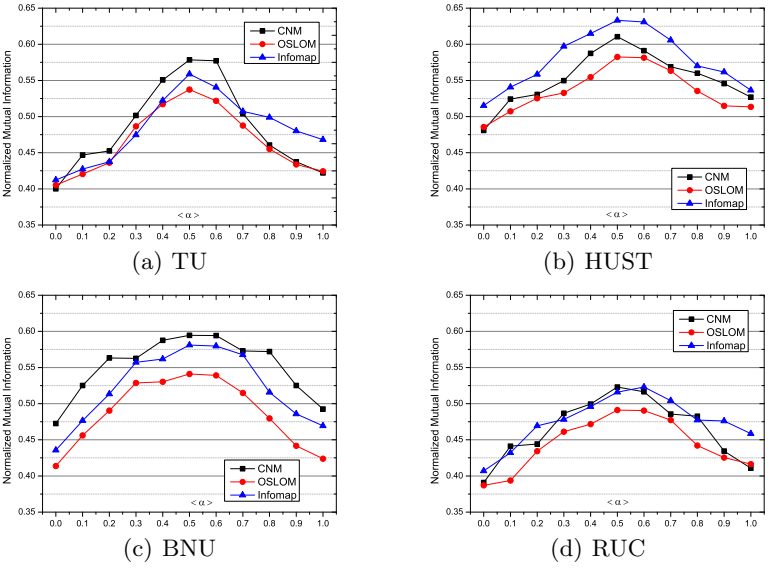


Fig. 5. Results of our method on four real-world networks

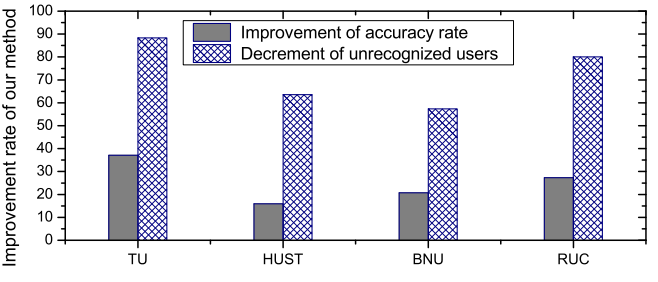


Fig. 6. Performance improvements of our method on four real-world networks

processed. The right pillars in Fig. 6 show the decrement rates of unrecognized users on test networks, and the mean value is about 72%.

7 Conclusion

In this paper, we have characterized the structure and content of many crawled sample microgroups in detail, and proposed a united method to combine link structure and user attribute for microgroup detection on TSina. Using the new method, link structure and attribute similarity between two users are converted to the edge weight of a newly generated network. Through extensive experiments on four real-world social networks, we have observed that our method achieves significant improvement over the traditional algorithms considering only link structure or user attribute.

For future work, inspired by the fact that user's interest can be extracted from published tweets with some topic models like PLSA and LDA, we plan to consider more factors to measure interest similarity between users, and try to improve the performance of our microgroup detection method. Moreover, we will validate the performance of our method on more data sets.

References

1. Fortunato, S.: Community detection in graphs. Physics Reports 486(3-5), 75–174 (2010)
2. Danon, L., Duch, J., Arenas, A., Daz-guilera, A.: Comparing community structure identification. Journal of Statistical Mechanics: Theory and Experiment 9008, 09008 (2005)
3. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. PNAS 99(12), 7821–7826 (2002)
4. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Phys. Rev. E 69(2), 26113 (2004)
5. Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., Parisi, D.: Defining and identifying communities in networks. Proceedings of the National Academy of Sciences 101(9), 2658 (2004)
6. Palla, G., Derenyi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. Nature 435, 814 (2005)
7. Arenas, A., Díaz-Guilera, A., Pérez-Vicente, C.J.: Synchronization reveals topological scales in complex networks. Phys. Rev. Lett. 96(11), 114102 (2006)
8. Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. Phys. Rev. E 70(6), 66111 (2004)
9. Flake, G., Lawrence, S., Giles, C., Coetzee, F.: Self-organization and identification of Web communities. Computer 35(3), 66–70 (2002)
10. Pothen, A., Simon, H.D., Liou, K.P.: Partitioning sparse matrices with eigenvectors of graphs. SIAM J. Matrix Anal. Appl. 11(3), 430–452 (1990)
11. Kernighan, B.W., Lin, S.: An Efficient Heuristic Procedure for Partitioning Graphs. The Bell system technical journal 49(1), 291–307 (1970)
12. Gregory, S.: Finding overlapping communities in networks by label propagation. New Journal of Physics 12(10), 103018+ (2010)
13. Stanoev, A., Smilkov, D., Kocarev, L.: Identifying communities by influence dynamics in social networks (April 2011)
14. Yang, T., Chi, Y., Zhu, S., Gong, Y., Jin, R.: Directed network community detection: A popularity and productivity link model. In: SIAM International Conference on Data Mining, pp. 742–753 (2010)
15. Cohn, D., Hofmann, T.: The missing link - a probabilistic model of document content and hypertext connectivity. In: Neural Information Processing Systems, vol. 13 (2001)
16. Getoor, L., Friedman, N., Koller, D., Taskar, B.: Learning probabilistic models of link structure. Journal of Machine Learning Research 3, 679–707 (2002)
17. Stephen, E.E., Fienberg, S., Lafferty, J.: Mixed membership models of scientific publications. Proceedings of the National Academy of Sciences (2004)
18. Yang, T., Jin, R., Chi, Y., Zhu, S.: Combining link and content for community detection: a discriminative approach. In: Knowledge Discovery and Data Mining, pp. 927–936 (2009)

19. Dietz, L., Bickel, S., Scheffer, T.: Unsupervised prediction of citation influences. In: Proceedings of the 24th International Conference on Machine Learning, pp. 233–240 (2007)
20. Amit Gruber, M.R.Z., Weiss, Y.: Latent topic models for hypertext. In: Uncertainty in Artificial Intelligence, pp. 230–239 (2008)
21. Cha, M., Mislove, A., Gummadi, P.K.: A measurement-driven analysis of information propagation in the flickr social network. World Wide Web Conference Series, pp. 721–730 (2009)
22. Kumar, R., Novak, J., Tomkins, A.: Structure and evolution of online social networks. In: Knowledge Discovery and Data Mining, pp. 611–617 (2006)
23. Kwak, H., Lee, C., Park, H., Moon, S.B.: What is twitter, a social network or a news media? World Wide Web Conference Series, pp. 591–600 (2010)
24. Newman, M.E.J.: Mixing patterns in networks. Phys. Rev. E 67(2), 26126 (2003)
25. Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. J. Am. Soc. Inf. Sci. Technol. 58, 1019–1031 (2007)
26. Kim, Y., Son, S.-W., Jeong, H.: Community Identification in Directed Networks. In: Zhou, J. (ed.) Complex 2009. LNICST, vol. 5, pp. 2050–2053. Springer, Heidelberg (2009)
27. Rosvall, M., Bergstrom, C.T.: Maps of random walks on complex networks reveal community structure. PNAS 105, 1118 (2008)
28. Lancichinetti, A., Radicchi, F., Ramasco, J.J.: Statistical significance of communities in networks. Phys. Rev. E 81(4), 46110 (2010)
29. Zachary, W.: An information flow model for conflict and fission in small groups. Journal of Anthropological Research 33, 452–473 (1977)
30. Lusseau, D., Schneider, K., Boisseau, O.J., Haase, P., Slooten, E.: The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. Behavioral Ecology and Sociobiology 54(4), 396–405 (2003)
31. Traud, A.L., Kelsic, E.D., Mucha, P.J., Porter, M.A.: Comparing Community Structure to Characteristics in Online Collegiate Social Networks. In: Proceedings of the 2009 APS March Meeting (March 2009)

Mining Good Sliding Window for Positive Pathogens Prediction in Pathogenic Spectrum Analysis*

Lei Duan[1], Changjie Tang[1], Chi Gou[1], Min Jiang[2], and Jie Zuo[1]

[1] School of Computer Science, Sichuan University,
Chengdu 610065, China
[2] West China School of Public Health, Sichuan University,
Chengdu 610041, China
{leiduan,cjtang}@scu.edu.cn

Abstract. Positive pathogens prediction is the basis of pathogenic spectrum analysis, which is a meaningful work in public health. Gene Expression Programming (GEP) can develop the model without predetermined assumptions, so applying GEP to positive pathogens prediction is desirable. However, traditional time-adjacent sliding window may not be suitable for GEP evolving accurate prediction model. The main contributions of this work include: (1) applying GEP-based prediction method to diarrhea syndrome related pathogens prediction, (2) analyzing the disadvantages of traditional time-adjacent sliding window in GEP prediction, (3) proposing a heuristic method to mine good sliding window for generating training set that is used for GEP evolution, (4) proving the problem of training set selection is NP-hard, (5) giving an experimental study on both real-world and simulated data to demonstrate the effectiveness of the proposed method, and discussing some future studies.

Keywords: Data Mining, Time Series, Sliding Window, Pathogens Prediction.

1 Introduction

Infectious disease prevention and control is an important and urgent issue in daily life. For example, thousands of people lost lives by SARS and A/H1N1. Correspondingly, adopting effective measures to prevent and control infectious diseases is a meaningful and challenging problem for public health research. To implement effective measures for infectious disease prevention and control, it is necessary for scientists to make clear of the infectious agent, that is, the pathogen of the infectious disease. The pathogen is a disease producer such as a virus, bacteria, prion, or fungus that causes disease to its host. For example, SARS is caused by a coronavirus [1].

The pathogenic spectrum of an infectious disease demonstrates the constituent ratio of each pathogen, which is related to the infectious disease. Example 1 gives an

* This work was supported by the National Research Foundation for the Doctoral Program by the Chinese Ministry of Education under grant No.20100181120029, and the Young Faculty Foundation of Sichuan University under grant No. 2009SCU11030.

J. Tang et al. (Eds.): ADMA 2011, Part II, LNAI 7121, pp. 152–165, 2011.

example of calculating the pathogenic spectrum. Pathogenic spectrum analysis is a meaningful work in public health, since the variation of the pathogenic spectrum is the basis of disease break. Specifically, the change rate of the pathogenic spectrum is a significant indicator to evaluate the possibility of infectious disease break. Moreover, predicting the trend of the pathogenic spectrum alternation is helpful for the early warning of infectious disease break.

Example 1. Given an infectious disease, suppose there are four viruses, v_1, v_2, v_3 and v_4, related to it. The virus-test result shows that the numbers of cases that are positive to these four viruses are 20, 50, 60 and 70, respectively. Then the virus pathogenic spectrum of this infectious disease consists of four parts. That is, v_1: 20/(20+50+60+70) = 10%, v_2: 50/(20+50+60+70) = 25%, v_3: 60/(20+50+60+70) = 30%, and v_4: 70/(20+50+60+70) = 35%.

As shown in Example 1, the problem of pathogenic spectrum prediction can be converted into predicting the number of positive cases of each disease-related pathogen. In practice, the public-health researchers apply the virus test to patients, and record the numbers of patients whose test results are positive. This kind of test is carried out in a fixed period, such as one week, one month. As a result, we can see that the positive pathogens prediction is a time series problem.

In public health domain, some traditional time series analysis methods, such as ARMA, ARIMA [2-4], Artificial Neural Networks (ANN) [5-7], which are implemented in SAS or SPSS software, have been widely used in positive pathogens prediction. However, none of these methods works well in all situations. For example, ARIMA is suitable for developing a linear model, while the disadvantages of ANN include "black box" nature, computational burden, proneness to over fitting, and the empirical nature of model building. Traditional methods may fail to develop adequate models due to the nonlinear dynamic behavior of time series, but also due to the lack of adaptation of the methods. This makes the problem is suitable for using heuristic methods, like evolutionary computation, which can develop the model without making many assumptions. For example, Genetic Programming has been widely performed for time series forecasting [8, 9].

The diarrhea syndrome monitoring data records the numbers of positive pathogens that are related to diarrhea syndrome every month since 2009 in China mainland. In this study, we apply GEP (Gene Expression Programming, GEP), the newest development of Genetic Programming [10, 11], to positive pathogens prediction in diarrhea syndrome monitoring data analysis.

We choose GEP as the prediction method, since it has following advantages:

- GEP can learn the fittest model from the data automatically without any predefined assumption. It has a powerful numeric calculation capability to evolve accurate model.
- Previous studies on applying GEP to time series analysis get desirable results.

The basic idea of applying GEP to time series mining is a sliding window prediction method. In training stage, once the size of sliding window is determined, the training

set can be generated by the sliding window. For data in sliding windows, GEP takes them as the independent variables and evolves a model to fit the target values. Moreover, the sliding window is always time-adjacent prior to the target value. For example, given a dataset $D = \{d_i \mid 1 \le i \le n\}$, suppose the sliding window size is 3. Then, for target value d_i, the dataset in sliding window is $\{d_{i\text{-}3}, d_{i\text{-}2}, d_{i\text{-}1}\}$. However, Example 2 demonstrates that this kind of sliding window may not be suitable to predict positive pathogens in diarrhea syndrome monitoring data.

Example 2. Let dataset $D = \{d_i \mid 1 \le i \le 24\}$ be the numbers of positive cases of a diarrhea syndrome related pathogen every month since 2005 to 2006. Each data in D is list in table below.

	Jan	*Feb*	*Mar*	*Apr*	*May*	*Jun*	*Jul*	*Aug*	*Sep*	*Oct*	*Nov*	*Dec*
2005	47	29	32	38	19	21	37	11	23	38	22	33
index	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*
2006	35	19	24	32	14	17	34	8	21	36	20	31
index	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*

Suppose the sliding window size is 3. If we apply GEP to find the relationship between d_i and (d_i-1, d_i-2, d_i-3), $4 \le i \le 24$, GEP fails to find accurate relationship. However, if we apply GEP to find the relationship between d_i and (d_i-1, d_i-12, d_i-13), $14 \le i \le 24$, GEP can find that $d_i = d_{i\text{-}12} + (d_{i\text{-}1} - d_{i\text{-}13}) * 0.8$.

Though Example 2 is a simple synthetic example, it reveals the fact that traditional time-adjacent sliding window is not suitable for predicting positive pathogens, which are related with diarrhea syndrome. The reasons include:

- Firstly, in the diarrhea syndrome monitoring data analysis, the number of positive pathogens is related to environment factors, such as season, temperature. For example, it is unreasonable to predict the number of positive pathogens in autumn by the numbers in summer.
- Secondly, besides the numbers of positive pathogens in previous months, the numbers in the same of months of last year is important while predicting the positive pathogens of current month.

Additionally, Example 2 shows that sliding window is important for GEP. Since GEP takes the data in sliding window as independent variables, it cannot evolve the accurate prediction model from incorrect dataset.

To the best of our knowledge, there is no previous work on mining sliding window for GEP prediction. The main contributions of this work include: (1) applying GEP-based prediction method to diarrhea syndrome related pathogens prediction, (2) analyzing the disadvantages of traditional time-adjacent sliding window in GEP prediction, (3) proposing a heuristic method to mine good sliding window for generating training set that is used for GEP evolution, (4) proving the problem of training set selection is NP-hard, (5) giving an experimental study on both real-world

and simulated data to demonstrate the effectiveness of the proposed method, and discussing some future studies.

The rest of this paper is organized as follows. Section 2 introduces related works. Section 3 presents the main ideas used by our methods and the implementation of the algorithm. Section 4 reports an experimental study on both real-world diarrhea syndrome monitoring data and synthetic data. Section 5 discusses future works, and concluding remarks.

2 Related Works

2.1 Traditional Time Series Prediction Methods

Time series study is distinct from other data analysis problems, since time series data have a natural temporal ordering. By time series study, scientists can extract meaningful statistics and other characteristics of the data, and use the model to forecast future events based on known past events. Specifically, the model developed by time series prediction from the past data is used to predict data points before they are measured. Time series study has been widely applied in many domains, such as econometrics, meteorology, astronomy.

The model for time series data represents stochastic process. Based on the model form, time series prediction methods can be classified into three types: linear model, such as ARMA, ARIMA [12], non-linear, such as ARCH, GARCH [13, 14], and model-free, such as some wavelet transform based methods [15].

Traditional time series modeling methods have been widely applied to many infectious disease prevention and control studies [2-7]. The authors in [3] used ARIMA to predict the number of beds occupied during a SARS outbreak in a Singapore's tertiary hospital. In [4], ARIMA is used to predict the incidence of pulmonary tuberculosis. ANN can overcome the linear-modeling limitation of ARIMA, so it has been applied to many disease incidence predictions, such as cancer and hepatitis [6, 7].

2.2 GEP-Based Time Series Prediction

GEP is a new development of Genetic Algorithms (GA) and Genetic Programming (GP). The basic steps of using GEP to seek the optimal solution are the same as those of GA and GP. However, compared with GA or GP, the coding of individuals (candidate solutions) in GEP is more flexible and efficient [10, 11].

The most characteristic players in GEP are the chromosomes and the expression trees, the latter consisting of the expression tree of the genetic information encoded in the former. The chromosome is a linear, symbolic string of fixed length. One or more genes compose a chromosome by using linking function. Each gene is divided into a head and a tail. The head contains symbols that represent both functions and terminals, whereas the tail contains only terminals [10]. For each problem, the length of the head h is chosen by the user, whereas the length of the tail t is a function of h and the number of arguments of the function with more arguments n, and is evaluated

by the equation: $t = h\ (n - 1) + 1$. Consider a gene for which the set of functions F = {+, -, *, /}. In this case the maximum number of arguments of the element in F is 2, then $n = 2$.

In GEP, the length of a gene and the number of genes composed in a chromosome are fixed. Despite its fixed length, each gene has the potential to code for expression trees of different sizes and shapes, the simplest being composed of only one node and the biggest composed of as many nodes as the length of the gene [11].

Through parsing the expression tree in the hierarchy way, the algebraic expression part of GEP genes can be obtained. The structural organization of GEP genes guarantees that any genic change in the chromosome always generates a valid expression tree [10]. That is, all candidate solutions evolved by GEP are syntactically correct. The chromosome is called as the individual's genotype, while the expression tree is called as the individual's phenotype [11].

Figure 1 shows a gene is encoded as a linear string and its expression in expression tree. The valid part of gene is shown in bold in Figure 1.

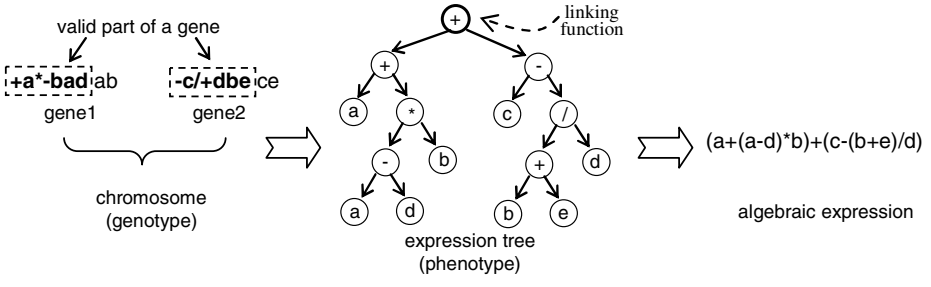


Fig. 1. The genotype, phenotype and algebraic expression of a GEP individual

The GEP algorithm begins with generating of the initial population, composed of a set of chromosomes, in a random way. Each chromosome is a candidate solution. Then the chromosomes (genotype) are expressed as expression trees (phenotype) and the fitness of each individual is evaluated by the predetermined fitness function. There are many kinds of measurements can be used as the fitness function, such as relative error and absolute error. The individuals are then selected according to fitness to reproduce with modification, generating new individuals with new traits. The individuals of the new generation are subjected to the same evolutionary process: expression of the genomes, selection by the fitness, and new individual generation. This procedure is repeated until a satisfactory solution is found, or a predetermined stop condition is reached. Then the evolution stops and the best-so-far solution, evolved by GEP, is returned [10].

GEP creates necessary genetic diversity for the selected individuals for keeping the evolutionary power in the long run. In nature, several genetic modifications, such as mutation, deletion, and insertion, are performed during the replication of the genomes. In basic GEP algorithm, the genetic operators perform in an orderly fashion, starting with replication and continuing with mutation, transposition, and recombination. The details of GEP implementation can be referred in [11].

Since GEP can evolve accurate mathematic model, it is used to build the prediction model that fits the time series data as well as possible. C. Ferreira gives a basic sliding window based method of applying GEP to time series study in [11]. This method consists of two steps.

- The first step is deciding the size of sliding window, that is, how many previous data points are used in predicting current data point. Suppose the window size is s. then the model predicts the value at a moment t, d_t, using the previous s values in the sample, denoted as d_{t-1}, d_{t-2},…, d_{t-s}. Based on the sliding window, the data set is participated into several training samples.
- Then GEP evolves a function f that predicts the values of a time series data as accurately as possible. Formally, let $f(d_{t-1}, d_{t-2},\ldots, d_{t-s}) = d'_t$, the function f that has the smallest error between d_t and d'_t is the best model, which is to be used for further prediction.

GEP has been used successfully to solve various time series problems so far. Besides the work in [11], the authors in [16] designed a GEP-based method, called as Differential by Microscope Interpolation, for sunspot series prediction. In [17], the authors applied an adaptive GEP-based method to predict the precipitation and temperatures in a region of Romania.

3 Sliding Window Mining

3.1 Sub-sliding Windows Enumeration

As stated above, the first step of applying GEP to time series prediction is determining the sliding window. In the basic GEP-based method for time series prediction, if the size of sliding window is s, for the value to be predicted at a moment t, d_t, the data in sliding window are previous s values to d_t. However, this kind of time-adjacent window may not be suitable for diarrhea syndrome monitoring data analysis as shown in Example 2.

Let the number of observed data be n. We select s data to compose the sliding window. Then, there will be C_n^s different sliding windows for selection. In general, the size of sliding window, s, is no greater than half of all observed data, $n/2$. We can get following:

$$C_n^s = \frac{n!}{(n-s)!s!} \geq \frac{(2s)!}{(2s-s)!s!} = \frac{(2s)!}{(s!)^2} \geq 2^s \tag{1}$$

From equation presented above, we can see that the search of selecting s data from all observed data is in exponential space. As a result, a polynomial time algorithm cannot enumerate all sliding windows that consist of s data.

From the diarrhea syndrome monthly monitoring data, we get two observations as follows. Firstly, the periodicity exists in the monthly positive pathogens. Intuitively, it is worthwhile to consider the pathogenic spectrum in June 2009, when predict the one in June 2010. Secondly, it is unreasonable to select much data, which are in the same

observation time period but the intervals to the predicted data are large in the sliding window. For example, compared with Nov. 2010, the data in Feb. 2010 is not helpful to improve the prediction accuracy of pathogenic spectrum in Dec. 2010.

According to the characteristics of monthly positive pathogens prediction, we design a heuristic method to enumerate candidate sliding windows based on following two principles:

- Considering data in previous time periods while predicting the current data.
- Paying more attention to the recent than to the past in prediction.

Given a dataset, D, contains all observed data. Let $d_t \in D$ be the value to be predicted at moment t, the time period of the observed data be T. Then we divide D into time-partitions, P, from d_t backward. For each $p_i \in P$, $p_i = \{d_r \mid 1 \le t - i \cdot T < r \le t\text{-}(i\text{-}1)\cdot T\}$. So, p_i with smaller index is closer to d_t. For example, suppose $T = 4$ and $t = 15$, then $p_1 = \{d_{12}, d_{13}, d_{14}, d_{15}\}$ as well as $p_2 = \{d_8, d_9, d_{10}, d_{11}\}$.

Definition 1 (sub-sliding window). Given a sliding window W, w_i is a sub-sliding window of W, iff w_i satisfies following conditions: i) $W = \bigcup_{i=1}^{k} w_i$; ii) $w_i \cap w_j = \varnothing$, $i \neq j$.

Definition 1 shows that $|W| = |w_1| + |w_2| + \ldots + |w_k|$, and $0 \le |w_i| \le |W|$. In this study, the value of k is determined by the user, and each sub-sliding sliding window is a subset of time-partition. That is, for each $w_i \in W$, $w_i \subseteq p_i$. As a result, the value of k is not greater than $|P|$. It is worthwhile to note that as w_i can be null ($\varnothing$), k is the maximal number of sub-sliding windows. The data in each sub-sliding window satisfy following constraints.

Constraint (i) The data in w_i are those in the rightmost side of p_i. But d_t is excluded from w_1, since it is the value to be predicted.

Constraint (ii) $|w_{i+1}| - |w_i| < \delta$, where δ is a predefined small positive integer.

In this work, we set δ is 1, since we prefer to pay more weight to the recent than to the past in prediction. Alternatively, the relationship between $|w_i|$ and $|w_{i+1}|$ can be a ratio. Note that, there is no limitation of how greater $|w_i|$ than $|w_{i+1}|$ is. Constraints (i) and (ii) satisfy the two principles stated above.

Example 3. Given a sliding window $W = \{w_1, w_2, w_3\}$, $|W| = 7$. Suppose the time period is 5 and d_{15} is the value to be predicted. Figure 2 illustrates the sub-sliding windows, when $|w_1| = 3$, $|w_2| = 2$, and $|w_3| = 2$.

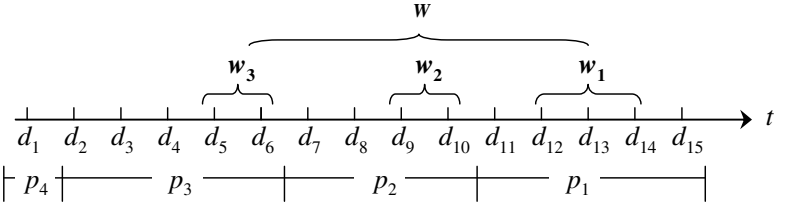


Fig. 2. An Example of a sliding window consists of 3 sub-sliding windows

Sub-sliding windows compose the sliding window for prediction. Given the size of sliding window and the maximal number of sub-sliding windows, there are different sub-sliding window combinations. Take w_1, w_2, and w_3 in Example 3 as an example, the lengths of them, denoted as ($|w_1|$, $|w_2|$ $|w_3|$), can be (7, 0, 0), (6, 1, 0), (6, 0, 1), (5, 2, 0), (5, 1, 1), (4, 3, 0), (4, 2, 1), (4, 1, 2), (3, 4, 0), (3, 3, 1), (2, 3, 2), (2, 2, 3), besides (3, 2, 2). The data in sub-sliding window are determined as soon as the size of the sub-sliding window is determined.

3.2 Finding the Best Sliding Window

Once the size of sliding window and the maximal number of sub-sliding windows are determined, an available sub-sliding window combination can be generated. Each sub-sliding window combination constructs a candidate sliding window. We find all candidate sliding windows that satisfied Constraint (i) and (ii), and make use of these candidate sliding windows to generate training sets. Afterwards, we apply GEP to training sets to evolve the best model as well as good sliding window. Algorithm 1 describes the pseudo code of finding the most accurate model for prediction.

Algorithm 1: Prediction_Model_Mine (D, T, w, k)
Input: (1) observed dataset: D; (2) the time period: T; (3) the size of sliding window: w; (4) the maximal number of sub-sliding windows: k.
Output: prediction model: gepModel.

```
begin
    1. subwinSet ← subWinGenerate(w,k)
    2. dataSubSet ← DataSplit(D,T)
    3. For each subwin in subwinSet
    4.      trainingSet ← Select(subwin, dataSubSet)
    5.      TrainSets ← TrainSets + trainingSet
    6. For each trainset in TrainSets
    7.      gepScore ← gepPrediction(trainset)
    8. gepModel ← the model with the highest gepScore
    9. return gepModel
end.
```

In Algorithm 1, Function *subWinGenerate*(*w*,*k*) in Step 1 generates all candidate sliding windows, which satisfy Constraint (i) and (ii), based on the size of sliding window (*w*) and the maximal number of sub-sliding windows (*k*). For each data to be taken as a target value in training set, Function *DataSplit*(*D*,*T*) in Step 2 divides the data before it into several time-partitions. From Step 3 to Step5, for each candidate sliding window, Function *Select*(*subwin*, *dataSubSet*) generates the training samples by selecting data from *dataSubSet* based on sliding window *subwin*. Each generated training set is evaluated by GEP in Step 7. Then the most accurate model (good sliding window) evolved by GEP will be used for prediction.

Proposition 1. Given a sliding window $W = \{w_1, w_2, w_3, \ldots, w_k\}$, $k > 1$ and $|W| = M$. Let $NC(W)$ be the number of sub-sliding window combinations satisfying the Constraints (i) and (ii). Then $NC(W) < (M+2)^{k-1}$.

Proof. We prove Proposition 1 by induction.

Basis: When $k = 2$, $W = \{w_1, w_2\}$, $|w_2| = M - |w_1|$. As $|w_1| \in \{0, 1, 2, ..., M\}$, $NC(W) = (M+1) < (M+2)$ as desired.

Inductive steps: Assume $NC(W) < (M+2)^{n-1}$, when $k = n$. That is, $W = \{w_1, w_2, \ldots, w_n\}$. For $k = n + 1$, let $W = W' \cup w_{n+1}$, where $W' = \{w_1, w_2, w_3, \ldots, w_n\}$. As $|W'| = M - |w_{n+1}|$, $NC(W') < (M - |w_{n+1}| + 2)^n < (M + 2)^n$. The number of combinations between W' and w_1 is $(M + 1)$. Thus, $NC(W) < (M + 2)^n \cdot (M + 1) < (M + 2)^{(n+1)} = (M + 2)^k$.

Thus, it holds for $k = (n + 1)$ and this completes the proof.

Proposition 1 shows that the number of sub-sliding window combinations is increased in polynomial space. Thus, the calculation-step in Algorithm 1 generates training sets for prediction in polynomial time.

Algorithm 1 describes the process of generating the training sets followed by applying GEP method to evolve the best model for prediction. In this work, we call this process as *Training set selection problem*.

Intuitively, the training set selection problem is more difficult than finding the minimal attribute reduction of decision table, which is a NP-hard problem proved by Wong S K M and Ziarko W [18]. From Reference [19], we have following lemma.

Lemma 1. The problem of subset sum is NP-complete.

Theorem 1. The problem of training set selection is NP-hard.

Proof. The basic idea of proof is proving that the subset sum problem is polynomial time Turing-reducible to training set selection problem.

Given an integer set $C = \{c_1, c_2, \ldots, c_n\}$, construct a training set D_T as follows.

$$D_T = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{bmatrix} \text{ where } a_{ij} \in \{c_1, c_2, \ldots, c_n\} \cup \{0\}, \text{ and } m \geq 2^n.$$

Suppose the mother function:

$$f(x) = b_0 + b_1 x + b_2 x^2 + \ldots + b_n x^n - d_t$$

where $x \in [1 - \varepsilon, 1 + \varepsilon]$, ε is a predefined small positive number, d_t is the target value in prediction. Without loss of generality, we assume both training data and d_t are integers, since we can expand all non-integer values to integers synchronously.

We now apply GEP to train $f(x)$ over D_T to optimize $f(x)$, so that $|f(x)| < 1$. The values of $(b_0, b_1, b_2, \ldots, b_n)$ are fetched from D_T in training process.

When $x = 1$,

$$|f(1)| = |b_0 + b_1 + b_2 + \ldots + b_n - d_t| < 1$$

As all values of b_0, b_1, b_2, …, b_n and d_t are integers, the value of $|f(1)|$ is an integer. However, the only integer that is less than 1 is 0, so $|f(1)| = 0$. Then, $b_0 + b_1 + b_2 + \ldots + b_n = d_t$. This shows that the subset sum problem is reduced to this problem.

By Proposition 1, the size of D_T, generated in our proposed method, is increased in polynomial space. The training set for prediction can be generated in polynomial time. Moreover, based on [9], GEP can evolve $f(x)$ to the optimize-target in polynomial time. Finally, the subset sum problem can be reduced to training set selection problem in polynomial time. The problem of training set selection is NP-hard.

4 Experimental Study

4.1 Real-World Positive Pathogens Prediction

To evaluate the performance of our GEP-based sliding window mining method, we implement all proposed algorithms in Java. The experiments are performed on an Intel Pentium Dual 1.80 GHz (2 Cores) PC with 2G memory running Windows XP operating system. We apply our proposed method to the real-world diarrhea syndrome monitoring data, which is provided by the department of health statistics, Sichuan University. There are four viruses, calicivirus, rotavirus, adenovirus and astrovirus, related to diarrhea syndrome, so these four viruses are pathogens of diarrhea syndrome. The monitoring data contains the numbers of cases that are positive to virus test for these four pathogens in every month of Year 2009 and Year 2010. Since the data is sensitive, we skip over the semantic details and formulate the data formally as follows. Let D be the monitoring data of any related pathogen. Suppose $D = \{d_i \mid 1 \leq i \leq 24\}$, where d_i is the number of positive pathogen cases in one month. In D, d_1 is the data of January in 2009, and d_{24} is the data of December in 2010. In our work, we just consider the effectiveness of the proposed method in the real-world positive pathogens prediction, instead of the meaning of predicted data.

As the observation time period of diarrhea syndrome monitoring data is one year, the monitoring data can be divided into two time-partitions, p_1 and p_2, at most based on Algorithm 1. Take d_{24} as an example, the first time-partition p_1 contains data in Year 2010, and the second time-partition p_2 contains data in Year 2009.

Table 1. Parameters for GEP Evolution

Parameter	value	Parameter	value
Population size	100	One-point recombination rate	0.4
Number of Generations	10000	Two-point recombination rate	0.2
Linking function	+	Gene recombination rate	0.1
Function set	{+, –, *, / }	IS transposition rate	0.1
Number of genes	3	IS elements length	1, 2, 3
Gene head size	8	RIS transposition rate	0.1
Selection operator	tournament	RIS elements length	1, 2, 3
Mutation rate	0.04	Gene transposition rate	0.1

The proposed method is applied to the monitoring data to generate the training sets for GEP prediction. The size of sliding window is set as 6. In considering the requirement of diarrhea syndrome analysis and the total number of monitoring data is small, for each pathogen, we take the last monitoring data, d_{24}, as the test value in the experiments. Let W be the sliding window. There are two sub-sliding windows, w_1 and w_2, satisfying $w_1 \subseteq p_1$ and $w_2 \subseteq p_2$. The proposed method generates different combinations of w_1 and w_2, as well as corresponding training sets. For each training set, we run GEP 10 times independently, and record the average training accuracy and prediction accuracy, which are measured in absolute error. Table 1 lists the GEP related parameters in our experiments.

Table 2 to Table 5 lists the experiment results. As the size of sliding window is set as 6, the available sub-sliding window combinations include ($|w_1|$=6, $|w_2|$=0), ($|w_1|$=5, $|w_2|$=1), ($|w_1|$=4, $|w_2|$=2) and ($|w_1|$=3, $|w_2|$=3). We take the combination ($|w_1|$=6, $|w_2|$=0) as the baseline sliding window, for it is the traditional time-adjacent sliding window. The highest average accuracies of training and test are in bold font. As the main purpose of this experiment is verifying the effectiveness of the method to discover good sliding window for GEP prediction, without loss of generality, we apply the basic GEP method on each training set. We believe that more accurate prediction results can be got by some improved GEP methods, such as the methods in [16, 17].

Table 2. The experimental results on positive calicivirus prediction when $|W| = 6$

	($\|w_1\|$=6, $\|w_2\|$=0)	($\|w_1\|$=5, $\|w_2\|$=1)	($\|w_1\|$=4, $\|w_2\|$=2)	($\|w_1\|$=3, $\|w_2\|$=3)
Training Accu.	96.21	89.72	93.08	**82.60**
Test Accu.	135.67	98.89	104.10	**45.67**

Table 3. The experimental results on positive rotavirus prediction when $|W| = 6$

	($\|w_1\|$=6, $\|w_2\|$=0)	($\|w_1\|$=5, $\|w_2\|$=1)	($\|w_1\|$=4, $\|w_2\|$=2)	($\|w_1\|$=3, $\|w_2\|$=3)
Training Accu.	**102.82**	109.39	116.68	107.22
Test Accu.	**222.89**	357.33	330.75	290.57

Table 4. The experimental results on positive adenovirus prediction when $|W| = 6$

	($\|w_1\|$=6, $\|w_2\|$=0)	($\|w_1\|$=5, $\|w_2\|$=1)	($\|w_1\|$=4, $\|w_2\|$=2)	($\|w_1\|$=3, $\|w_2\|$=3)
Training Accu.	18.34	18.77	17.18	**16.04**
Test Accu.	10.56	12.73	14.75	**5.13**

Table 5. The experimental results on positive astrovirus prediction when $|W| = 6$

	($\|w_1\|$=6, $\|w_2\|$=0)	($\|w_1\|$=5, $\|w_2\|$=1)	($\|w_1\|$=4, $\|w_2\|$=2)	($\|w_1\|$=3, $\|w_2\|$=3)
Training Accu.	14.79	13.69	14.21	**12.85**
Test Accu.	44.78	56.80	58.17	**32.11**

From Table 2 to Table 5, we can see that for predicting the positive pathogens of calicivirus, rotavirus, adenovirus and astrovirus, different sub-sliding window combinations get different training and test accuracies. The highest training accuracy and test accuracy can be got when the combination of sub-sliding windows is ($|w_1|$=3,

$|w_2|$=3), while for predicting the positive pathogens of adenovirus, the highest training accuracy and test accuracy are got when the combination of sub-sliding windows is ($|w_1|$=6, $|w_2|$=0). Thus, better sliding window for prediction, compared with traditional time-adjacent sliding window, can be discovered by our proposed method. Moreover, in the case of time-adjacent sliding window is good for prediction our method also can find it, such as predicting the positive pathogens of rotavirus.

For each training set generated by the good sliding window, we increase the number of evolution generations as 30000, and run GEP 10 times independently. Then more accurate prediction results can be got as list in Table 6.

Table 6. The prediction accuracy of GEP-based method evolving 30000 generations

	calicivirus	rotavirus	adenovirus	astrovirus
Test Accu.	38.17	217.80	3.83	18.50

As shown in above tables (from Table 2 to Table 6), we can see that it is necessary and effective to apply the proposed method to diarrhea syndrome related pathogens prediction to discover good sliding windows, which can improve the prediction accuracy .

4.2 Synthetic Data Prediction

As there are only two years real-world diarrhea syndrome monitoring data available, in order to demonstrate that the proposed sliding window mining method is effective for long-term monitoring data, we copy the calicivirus monitoring data 10 times to simulate the 20-years monitoring data. We apply the proposed method to the simulated data to generate the training sets for GEP prediction. The size of sliding window is set as 7. The maximal number of sub-sliding window (k) is set as 2, since we simulate the data by the 2-years monitoring data. The sliding window includes the data, which equal to the values of the target data, in the case of k equals to 3.

For each dataset generated by sliding windows that enumerated by the proposed method, we keep the last 12 data as the test set, and run GEP on the rest data 10 times independently. The GEP related parameters are kept the same as shown in Table 1. Table 7 lists the average training and prediction accuracies of GEP model per data under each sliding window, which is composed by different sub-sliding windows.

Table 7. The experimental results on simulated data when $|W|$ = 7

	($|w_1|$=7, $|w_2|$=0)	($|w_1|$=6, $|w_2|$=1)	($|w_1|$=5, $|w_2|$=2)	($|w_1|$=4, $|w_2|$=3)	($|w_1|$=3, $|w_2|$=4)
Trai. Accu.	19.14	20.93	18.86	17.89	19.59
Test Accu.	22.80	26.50	22.25	20.40	25.25

Table 7 shows the model with the highest accuracy, evolved by GEP, is got when the combination of sub-sliding windows is ($|w_1|$=4, $|w_2|$=3). Besides combination ($|w_1|$=4, $|w_2|$=3), the model evolved under the combination ($|w_1|$=5, $|w_2|$=2) is more accurate than the one evolved under time-adjacent sliding window, i.e. the

combination ($|w_1|$=7, $|w_2|$=0). The combinations ($|w_1|$=6, $|w_2|$=1) and ($|w_1|$=3, $|w_2|$=4) are not suitable for prediction compared with other combinations.

In addition, compared with Table 2, we can see that the performance on synthetic dataset is better than the one on real dataset. We analyze the reason lies that for synthetic dataset analysis, there are more data are taken as the training set, which improve the accuracy of evolved model. After all, from the experimental results on the synthetic data, we can see that mining good sliding window is helpful for GEP to evolve accurate models.

5 Discussions and Conclusions

Positive pathogens prediction is the basis of pathogenic spectrum analysis, which is a meaningful work in public health. Different form traditional methods that may fail to develop adequate models due to the nonlinear dynamic behavior of time series, or the lack of adaptation of the methods, GEP can develop the model without making many assumptions. As a result, applying GEP to positive pathogens prediction is desirable. However, traditional time-adjacent sliding window may not be suitable for GEP evolving accurate prediction model. Based on analyzing the characteristics of diarrhea syndrome, we propose a heuristic method to mine good sliding window for generating training set, which is used for GEP evolution. Furthermore, we prove the problem of training set selection is NP-hard. The experimental study on real-world positive pathogens prediction shows that our proposed method is necessary and effective for diarrhea syndrome related pathogens prediction.

There are many works worth to be deeply analyzed in the future. For example, how to add the environment factors in good sliding window mining, how to describe the relationships among previous data, and how to evaluate the candidate sliding windows in a fast way. Moreover, we will consider applying the proposed method to other applications in public health, and other domains, such as economics and finance.

Acknowledgments. The authors thank to the faculty of the department of health statistics, Sichuan University, for providing the research data and their helpful comments to this work.

References

1. United Nations World Health Organization, http://www.who.int/mediacentre/releases/2003/pr31/en/
2. Reis, B.Y., Mandl, K.D.: Time Series Modeling for Syndromic Surveillance. BMC Med. Inform. Decis. Mak. 3(1), 2 (2003)
3. Earnest, A., Chen, M.I., Ng, D., Sin, L.Y.: Using Autoregressive Integrated Moving Average (ARIMA) Models to Predict and Monitor the Number of Beds Occupied During a SARS Outbreak in a Tertiary Hospital in Singapore. BMC Health Services Research 5, 5–36 (2005)

4. Meng, Lei, Wang, Yuming: Application of ARIMA Model on Prediction of Pulmonary Tuberculosis Incidence. Chinese Journal of Health Statistics 27(5), 507–509 (2010)
5. Zhang, G.P.: Time Series Forecasting Using a Hybrid ARIMA and Neural Network Model. Neurocomputing 50, 159–175 (2003)
6. Khan, J., Wei, J.S., Ringnér, M., Saal, L.H., et al.: Classification and Diagnostic Prediction of Cancers Using Gene Expression Profiling and Artificial Neural Networks. Nature Medicine 7, 673–679 (2001)
7. Guan, P., Huang, D.-S., Zhou, B.-S.: Forecasting Model for the Incidence of Hepatitis A based on Artificial Neural Network. World Journal of Gastroenterology 10(24), 3579–3582 (2004)
8. De Falco, Della Cioppa, A., Tarantino, E.: A Genetic Programming System for Time Series Prediction and Its Application to El Niño Forecast. Advances in Soft Computing 32, 151–162 (2005)
9. Barbulescu, A., Bautu, E.: ARIMA Models versus Gene Expression Programming in Precipitation Modeling. In: Proc. of the 10th WSEAS Int'l Conf. on Evolutionary Computing, pp. 112–117 (2009)
10. Ferreira, C.: Gene Expression Programming: A New Adaptive Algorithm for Solving Problems. Complex Systems 13(2), 87–129 (2001)
11. Ferreira, C.: Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence. Angra do Heroismo, Portugal (2002)
12. Brockwell, P., Davies, R.: Introduction to Time Series. Springer, New York (2002)
13. Bollerslev, T.: Generalized Autoregressive Conditional Heteroskedasticity. Journal of Econometrics 31, 307–327 (1986)
14. Hacker, R.S., Hatemi, J.A.: A Test for Multivariate ARCH Effects. Applied Economics Letters 12(7), 411–417 (2005)
15. Chui, C.K.: An Introduction to Wavelets. Academic Press, San Diego (1992)
16. Zuo, J., Tang, C., Li, C., Yuan, C.-A., Chen, A.-l.: Time Series Prediction Based on Gene Expression Programming. In: Li, Q., Wang, G., Feng, L. (eds.) WAIM 2004. LNCS, vol. 3129, pp. 55–64. Springer, Heidelberg (2004)
17. Barbulescu, A., Bautu, E.: Time Series Modeling Using an Adaptive Gene Expression Programming Algorithm. International Journal of Mathematical Models and Methods in Applied Sciences 3(2), 85–93 (2009)
18. Wong, S.K.M., Ziarko, W.: On Optimal Decision Rules in Decision Tables. Bulletin of Polish Academy of Sciences 33(11-12), 693–696 (1985)
19. Sipser, M.: Introduction to the Theory of Computation, 2nd edn., Thomson Learning, Stanford (2005)

Mining Patterns from Longitudinal Studies*

Aída Jiménez, Fernando Berzal, and Juan-Carlos Cubero

Department of Computer Science and Artificial Intelligence
CITIC, University of Granada, Granada 18071 Spain
aidajm@gmail.com, {fberzal,jc.cubero}@decsai.ugr.es

Abstract. Longitudinal studies are observational studies that involve repeated observations of the same variables over long periods of time. In this paper, we propose the use of tree pattern mining techniques to discover potentially interesting patterns within longitudinal data sets. Following the approach described in [15], we propose four different representation schemes for longitudinal studies and we analyze the kinds of patterns that can be identified using each one of the proposed representation schemes. Our analysis provides some practical guidelines that might be useful in practice for exploring longitudinal datasets.

1 Introduction

A longitudinal study is a correlational research study where the same individuals are repeatedly observed through time, so that it is possible to identify changes in individuals over time [8] [7]. Longitudinal studies provide more accurate change observations than cross-sectional studies, which do not track the same individuals.

Researchers might be interested in two kinds of changes that can be observed in longitudinal studies. On the one hand, they might inquire about within-individual change over time. They study how individual outcome variables rise or fall over time. The goal of within-individual analysis is to describe the *shape* of each individual trajectory growth. On the other hand, we might also ask about inter-individual differences in changes. They evaluate whether different individuals present similar or different patterns of within-individual behavior and they look for predictors of the detected similarities and differences. The goal of inter-individual analysis is therefore to detect heterogeneity in change across individuals and to determine the relationships between the predictors and the shape of each individual trajectory growth [20].

Henceforth, let $i_1 \dots i_n$ be the individuals in the longitudinal study, where n is the number of individuals. For each individual, let $t_1 \dots t_T$ be the observations on the individual, where T is the number of observations performed during the longitudinal study. For each of those observations, let $v_1 \dots v_V$ be the variables that are tracked during the study, where V is the number of variables under

* The work described in this paper has been partially supported by the TIN2009-08296 research project from the Spanish Ministry of Science and Innovation.

J. Tang et al. (Eds.): ADMA 2011, Part II, LNAI 7121, pp. 166–179, 2011.

study. Finally, let $s_1 \dots s_S$ be the static features of each individual (i.e., those that are invariant over time, such as their *sex*), where S is the number of static features collected for the individuals participating in the longitudinal study.

Obviously, longitudinal datasets include the measured values for the different variables in each observation, as well as the static features of the individuals taking part in the longitudinal study. Those datasets can be organized in tabular form following two different approaches:

- A **person-level dataset** in which each individual is represented by a single record. This individual record includes all the measurements performed on that particular individual during the longitudinal study. In other words, each individual record contains the S static features of the individual and the $T \times V$ individual measurements corresponding to the T observations on the V variables tracked by the longitudinal study.
- A **person-period dataset** in which each individual is represented by several records, a different one for each observation. In this case, each tuple represents the V measurement performed on an individual in a given observation and there will be T different records for each individual under study. It should be noted that, for convenience, the static features of each individual are typically included in all the records representing the different observations on that individual.

A detailed discussion on the use of those alternative tabular representation formats can be found in [20]. Tables 1 and 2 illustrate them by showing the same (simplistic) longitudinal dataset using the person-level and person-period formats.

Longitudinal studies are often employed in biomedical [3], psychological [19], and sociological [18] research, where individuals are usually human beings, albeit they can also be animals. In Computer Science, they have been used to study the adoption of new technologies [10] and the evolution of user behavior [6].

Table 1. Person-level dataset representation of longitudinal studies

id	S_{sex}	$V_{0,age}$	$V_{0,height}$	$V_{0,weight}$	$V_{1,age}$	$V_{1,height}$	$V_{1,weight}$
1	m	15	(160,165]cm	(55,60]kg	16	(165,170]cm	(60,65]kg
2	f	13	(150,155]cm	(40,45]kg	14	(155,160]cm	(45,50]kg

Table 2. Person-period dataset representation of longitudinal studies

id	t	S_{sex}	V_{age}	V_{height}	V_{weight}
1	0	m	15	(160,165]cm	(55,60]kg
1	1	m	16	(165,170]cm	(60,65]kg
2	0	f	13	(150,155]cm	(40,45]kg
2	1	f	14	(155,160]cm	(45,50]kg

In this paper, we propose four tree-based representation schemes for longitudinal databases. We show how frequent tree pattern techniques can be employed in longitudinal studies to discover patterns that could not be obtained when using the traditional person-level or period-level representation formats. We also study the kind of patterns that can be identified using each representation scheme. This way, users will be able to choose the most suitable representation scheme depending on the kind of information they might be interested in.

Our paper is organized as follows. In Section 2, we review some basic concepts related to trees and we survey the state of the art in longitudinal data analysis. Section 3 presents four tree-based representation schemes for longitudinal studies. In Section 4, we analyze the kinds of patterns that can be identified in longitudinal studies using the proposed representation schemes. Finally, we end our paper with some conclusions in Section 5.

2 Background

Since our approach is based on trees, we now proceed to review some terminology related to trees, we survey the tree pattern mining techniques that make our approach feasible, and we provide some pointers to relevant work on longitudinal data analysis.

2.1 Trees

A **labeled tree** is a connected acyclic graph that consists of a vertex set V, an edge set $E \subseteq V \times V$, an alphabet Σ for vertex and edge labels, and a labeling function $L : V \cup E \rightarrow \Sigma \cup \varepsilon$, where ε stands for the empty label. The size of a tree is defined as the number of nodes it contains.

A tree is **rooted** if its edges are directed and a special node v_0, the root, can be identified. The root is the node from which it is possible to reach all the other vertices in the tree. In contrast, a tree is **free** if its edges have no direction, that is, when the tree is undirected. Therefore, a free tree has no predefined root.

Optionally, a tree can also have a binary ordering relationship $\leq$ defined over its nodes (i.e. '$\leq$' $\subseteq V \times V$). Rooted trees can be classified as **ordered trees**, when there is such a predefined order within each set of sibling nodes,

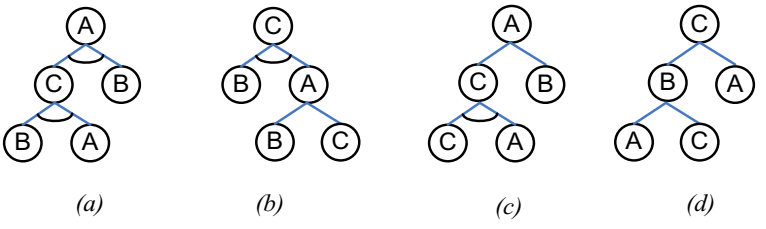


Fig. 1. Different kinds of rooted trees (from left to right): *(a)* completely-ordered tree; *(b)* and *(c)* partially-ordered trees; *(d)*unordered tree

or **unordered trees**, when there is no such a predefined order among sibling nodes. **Partially-ordered trees** contain both ordered and unordered sets of sibling nodes. They can be useful when the order within some sets of siblings is important but it is not necessary to establish an order relationship within all the sets of sibling nodes [14].

Figure 1 shows illustrates the different kinds of rooted trees. In this figure, ordered sibling nodes are joined by an arc, while unordered sets of sibling nodes do not share such an arc.

2.2 Tree Pattern Mining

Several frequent tree pattern mining algorithms have been proposed in the literature [13]. Most existing algorithms follow the Apriori iterative pattern mining strategy [2], where each iteration is broken up into two distinct phases:

- *Candidate Generation*: A candidate is a potentially frequent subtree. In Apriori-like algorithms, candidates are generated from the frequent patterns discovered in the previous iteration. Most algorithms generate candidates of size $k+1$ by merging two patterns of size k having $k-1$ elements in common. The most common strategies to generate such candidates are the following:
  - **Rightmost expansion** generates subtrees of size $k+1$ from frequent subtrees of size k by adding nodes only to the rightmost branch of the tree. This technique is used in algorithms like FREQT [1], uFreqt [17], and UNOT [4].
  - The **equivalence class-based extension** technique generates a candidate $(k+1)$-subtree by joining two frequent k-subtrees with $(k-1)$ nodes in common and that share a $(k-1)$-prefix in their string codification. This extension mechanism is used, for instance, in Zaki's TreeMiner [25] and SLEUTH [24] algorithms, as well as in our POTMiner [14] algorithm.
  - The **right-and-left tree join** method, which was proposed with the AMIOT algorithm [12], considers both the rightmost and the leftmost leaves of a tree in the generation of candidates.
  - Finally, the **extension and join** technique defines two extension mechanisms and a join operation to generate candidates. This method is used by HybridTreeMiner [5].
- *Support Counting*: Given the set of potentially frequent candidates, this phase consists of determining their actual support and keeping only those candidates whose support is above the predefined minimum support threshold (i.e., those candidates that are actually frequent).

Some of the proposed algorithms have also been derived from the FP-Growth algorithm [9]. Within this category, the PathJoin algorithm [23] uses compact structures called FP-Trees to encode input data, while CHOPPER and XSpanner [22] use a sequence-based codification for trees to identify frequent subtrees using frequent sequences.

2.3 Longitudinal Data Analysis

Longitudinal studies have traditionally resorted to statistical techniques, ranging from regression models [11] and parametric curves [8] to likelihood-based methods[7] and marginal models [16]. Statistical clustering techniques have also been used to cluster longitudinal data [8], e.g. for grouping individuals according to their behavior [21]. Scalable data mining techniques, however, have not been widely used to analyze data from longitudinal studies.

Researchers face many challenges when dealing with longitudinal datasets in practice, such as the treatment of missing data, the dynamic relationships between predictors and outcomes over time, and the crossover design (when two or more treatments are given to the same subject in different orders). Frequent pattern mining techniques are robust with respect to missing data and they can also be useful for dealing with the dynamics of longitudinal studies, which is behind changing relationships overtime and crossover design issues.

3 Tree-Based Representation Schemes for Longitudinal Data

In this section, we propose four alternative tree-based representation schemes for longitudinal data. These representation schemes let us obtain a tree set from a longitudinal study database.

The main idea behind the four representation schemes is that each individual in the longitudinal study will be represented as a partially-ordered tree.

In all the proposed representation schemes, the label at the root of the trees will be the name of the study and static features will be represented as children of this root node. Measurements from the different observations performed along the longitudinal study, however, will be represented in different ways depending on our particular choice of representation scheme, as we now proceed to detail.

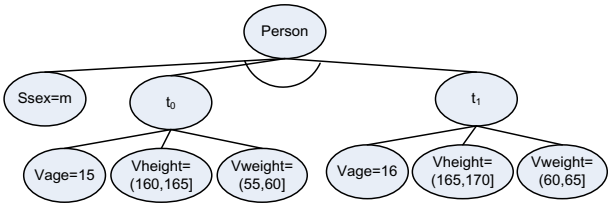
3.1 Timestamp-Based Representation

The timestamp-based representation scheme represents the observations as they are collected during the course of the longitudinal study. All the measurements taken during a given observation period are collated as a subtree within the tree representing a given individual history. This representation is, therefore, equivalent to the person-level dataset format traditionally employed for longitudinal datasets.

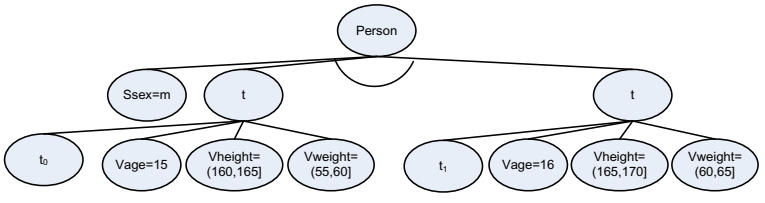
In the timestamp-based representation scheme, the root node of each individual tree has a child node for each observation in the study, for which we will use the notation t_i, where i is the observation number.

Each t_i node has as many child nodes as different variables are tracked in the longitudinal study. For each V_j variable, we will use the notation $V_j = v_{ij}$, where v_{ij} is the value of the variable V_j at the i-th observation.

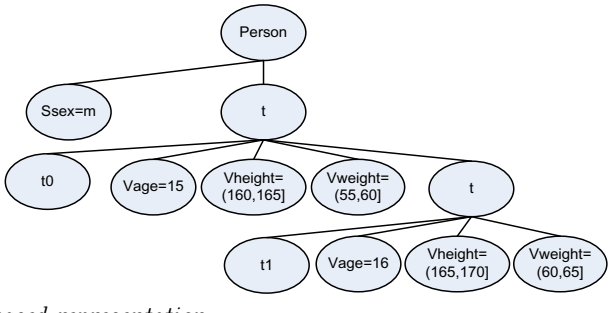
Figure 2 a) shows the timestamp-based representation of the first individual from the longitudinal study we introduced in Tables 1 and 2.



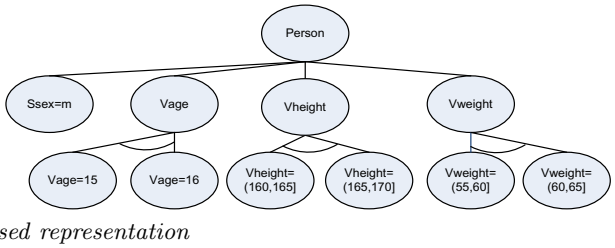
a) Timestamp-based representation



b) Shallow time-based representation



c) Deep time-based representation



d) Variable-based representation

Fig. 2. Different tree-based representations of the first individual from the longitudinal study shown in person-level format in Table 1 and in period-level format in Table 2

3.2 Shallow Time-Based Representation

The shallow time-based representation scheme represents all the observations at the same tree level and it employs intermediate nodes to group all the measurements performed at each observation. These intermediate nodes are the parents of the nodes representing each particular measurement and they also have a special node representing the observation number as their child.

Intermediate nodes are labeled with t, and each t node has a t_i child node, where i is the observation number. The sibling nodes of this t_i node are the nodes $V_j = v_{ij}$, where v_{ij} represents the value of the variable V_j at the i-th observation.

Figure 2 b) shows the shallow time-based representation of the first individual from the longitudinal study in Tables 1 and 2.

3.3 Deep Time-Based Representation

The deep time-based representation represents each observation at a different tree level. This way, sequences of consecutive observations are represented as paths within the trees. As in the shallow time-based representation scheme, intermediate nodes are introduced to collate all the measurements performed at each observation.

In the deep time-based representation scheme, the intermediate t node representing the first observation is a child of the root in the tree. The second intermediate t node is a child of the first t one, and so on. Therefore, the t node at depth i has, as children, the nodes $V_j = v_{ij}$ representing particular measurements at the i-th observation, the node t_i representing the observation number, and the t node of the $(i + 1)$-th observation.

Figure 2 c) shows the deep time-based representation of the first individual from the longitudinal study in Tables 1 and 2.

3.4 Variable-Based Representation

The variable-based representation scheme groups data according to the observed variables instead of the observation period.

In the variable-based representation scheme, the root node of each tree representing an individual has a child for each variable that has been tracked during the longitudinal study, for which we will use the notation V_j.

Each V_j node has a child node for representing the measurement of the variable performed at each observation, for which we will use the notation $V_j = v_{ij}$, where v_{ij} represents the value of the variable V_j at the i-th observation.

Figure 2 d) shows the variable-based representation of the first individual from the longitudinal study in Tables 1 and 2.

4 Identifying Frequent Patterns in Longitudinal Studies

The use of tree-based representation schemes for longitudinal databases lets us apply tree mining techniques to identify frequent patterns in longitudinal

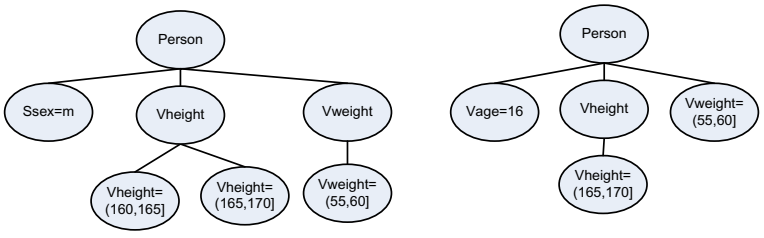


Fig. 3. Induced *(left)* and embedded *(right)* subtrees from the tree in Figure 2 d)

databases. Since we have used partially-ordered trees in our tree-based representation schemes, we propose the use of POTMiner [14] to identify patterns on the trees representing the individuals participating in a longitudinal study.

When working on trees, different kinds of subtrees can be defined depending on the way we define the matching function between the pattern and the tree it derives from:

- A **bottom-up subtree** T' of T, with root v, can be obtained by taking one vertex v from T with all its descendants and their corresponding edges.
- An **induced subtree** T' can be obtained from a tree T by repeatedly removing leaf nodes from a bottom-up subtree of T.
- An **embedded subtree** T' can be obtained from a tree T by repeatedly removing nodes, provided that ancestor relationships among the vertices of T are not broken.

Existing algorithms typically focus on identifying induced or embedded subtrees, as the ones shown in Figure 3, which can be obtained from the tree shown in Figure 2 d). If the induced subtree on the left is frequent in our longitudinal database, that means that male teenagers whose weight is in the (55,60]kg interval and change their height from (160,165]cm to (165,170]cm during the course of the longitudinal study are frequent in our database. When the embedded subtree on the right is frequent, 16-year-old teenagers who are (165,170]cm tall and weigh (55,60]kg sometime during the duration of the study are frequent in the population under study, albeit it is not necessarily true that all those attribute values happen at the same time (i.e. those particular values might have been measured at different observation periods).

4.1 Induced Patterns in Longitudinal Studies

Different induced patterns can be identified in longitudinal databases depending on the representation scheme we use. These patterns can provide us information about the following aspects of a longitudinal study:

- *Specific variable changes.* We can detect how a variable has changed over time and when that change occurred. This change patterns can be obtained using the timestamp-based representation scheme and correspond to the typical

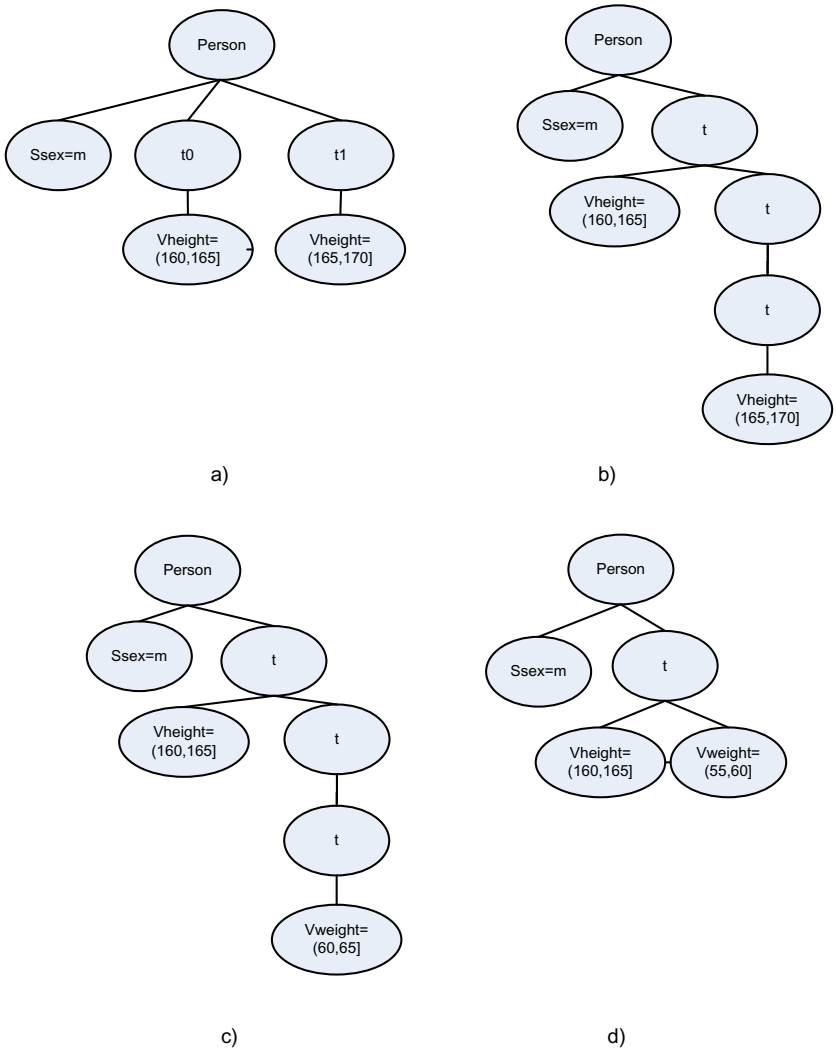


Fig. 4. Some induced patterns from the longitudinal database in Table 1

kind of information we can obtain when mining data directly from longitudinal studies in their traditional person-level and person-period formats. The pattern in Figure 4 a) shows the evolution of the height variable in male participants between the first and the second observation period.

– *Variable evolution after X observation periods.* We can detect how a variable has changed over time and how much time was needed for the change to occur. This evolution pattern can be obtained as an induced pattern using the deep time-based representation. The pattern in Figure 4 b) shows how the height variable evolved after two observation periods. In contrast to Figure 4 a), where the change was discovered in consecutive observations, the change

needed two observation periods in Figure 4 b). It should also be noted that it is not necessary for the evolution of all the individuals to happen at the same time, as it would be the case if we employed the traditional person-level and person-period representation formats for the longitudinal study. In Figure 4 b), the observed change might have occurred between the first and the third observation periods for some individuals, between the second and the fourth for others.

- *Response to stimuli after X observation periods.* When individuals are exposed to a given stimulus at different times, it is interesting to observe the consequence of the presence of the stimulus after a given number of observation periods. The response patterns of individuals to the presence of a given stimulus X observations later can be analyzed with the help of induced patterns if we resort to the deep time-based representation scheme. The pattern in Figure 4 c) shows that it is frequent for male teenagers to be between 165 and 170cm tall in a given observation period and weigh between 65 and 70kg two observation periods later.
- *Variable co-occurrence.* Induced patterns can also detect the co-occurrence of values of different variables in the longitudinal database. As before, these co-occurrences do not have to appear in the same observation period for all the individuals participating in the longitudinal study. Either the shallow or the deep time-based representation schemes can be used to discover such co-occurrence patterns. For instance, the pattern in Figure 4 d) shows that male teenagers who are between 160cm and 165cm tall and weigh between 55kg and 60kg during the same observation period are frequent in the longitudinal study.

4.2 Embedded Patterns in Longitudinal Studies

As happened with induced patterns, we can also employ embedded patterns to study different aspects of a longitudinal database. The information we can extract from embedded patterns can be useful for addressing the following issues:

- *Detecting frequent variable values*: Using any of the different representation schemes we proposed in the previous section, we can detect frequent values of the variables in the longitudinal study even when they might not be frequent during any particular observation period. These frequent patterns could also be discovered if we used the person-period database representation (provided that we were careful with the support counts for static features), but they cannot be discovered using the person-level dataset format. The pattern in Figure 5 a) represents a frequent set of features for individuals in our teenager longitudinal database (male, between 160cm and 165cm tall, and also between 60kg and 65kg).
- *Presence of uncommon variables*: Embedded patterns can detect the presence of uncommon variables, i.e. those that usually have null values because they do not apply to the majority of the individuals under study. Such patterns can help us relate their appearance to the evolution of other variables,

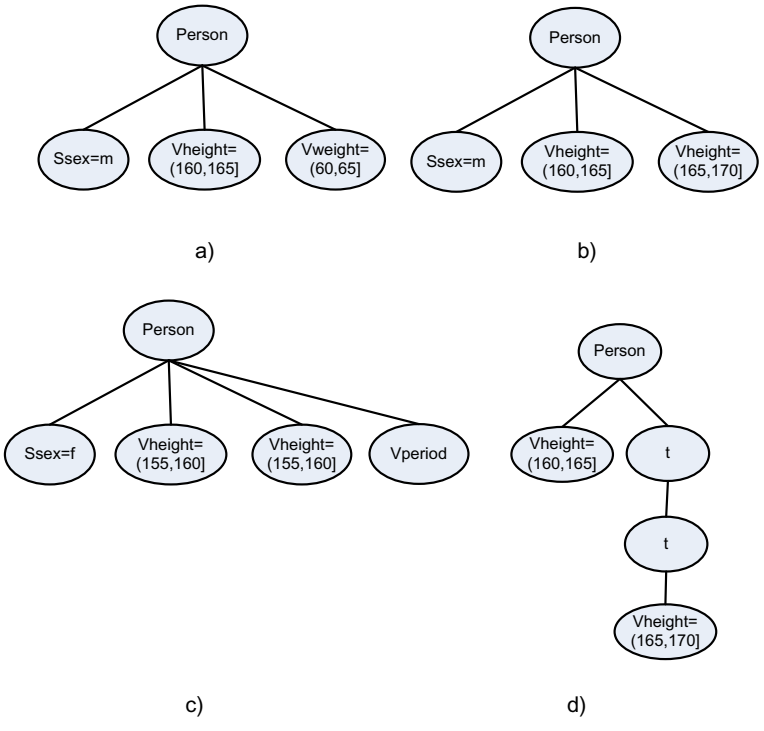


Fig. 5. Some embedded patterns from the longitudinal database in Table 1

but only when using the variable-based representation scheme. An example is shown in Figure 5 c), where it is observed that the presence of the period in female teenagers co-occurs with the absence of height growth.

– *Variable evolution during an indeterminate number of observation periods*: Embedded patterns can detect the evolution of a variable without taking into account how much time was needed for the evolution to occur, since intermediate nodes do not have to be present for the embedded pattern to appear. These embedded patterns can be discovered using any of the four tree-based representation schemes proposed in this paper, but neither the traditional person-level nor the person-period dataset format are suitable for analyzing such situations. The pattern in Figure 5 b) shows an increase in the height variable for male teenagers, even though the particular number of observation periods required by each individual to evolve might vary.
– *Variable evolution after at least X observation periods*: Embedded patterns can also show how a variable needs at least X observation periods to change. These patterns can only be identified when using the deep time-based representation scheme. The required number of observation periods is determined by the number of t intermediate nodes that appear in the embedded pattern. The pattern in Figure 5 d) illustrates that height changes from 160-165cm to 165-170cm are frequent after, at least, two observation periods.

Table 3. Summary of the kinds of patterns that can be identified using the four different tree-based representation schemes proposed in this paper for representing longitudinal databases

		Timestamp	Shallow time	Deep time	Variable
Induced	Specific changes	•			
	Evolution (fixed duration)			•	
	Response to stimulus			•	
	Variable co-occurrence		•	•	
Embedded	Frequent values	•	•	•	•
	Uncommon variables				•
	Evolution (uncertain duration)	•	•	•	•
	Evolution (minimum duration)			•	

4.3 Choosing the Right Representation Scheme

In this final subsection, we summarize the kinds of patterns that can be obtained by using the four different tree-based representation schemes we have proposed for longitudinal databases in Section 3.

Table 3 shows how the deep time-based representation scheme is the most versatile scheme since it is able to address six of the scenarios we have studied in the previous paragraphs. The deep time-based representation scheme lets us extract patterns that provide us information about variable evolution after a given, undeterminate, or bounded period of time. It lets us analyze the individual responses to a given stimulus when not all individuals receive the stimulus at the same time. It also addresses the variable co-occurrence problem (when such co-occurrences appear at different times for different individuals).

When we want to observe individual evolution at specific observation periods, we resort to the timestamp-based representation scheme or just employ the traditional person-level and person-period tabular representation formats.

The shallow time-based representation scheme is useful for finding variable co-occurrences when they do not happen at the same observation period for all individuals, albeit the deep time-based representation scheme can also be used in this situation.

Finally, the variable-based representation scheme is our preferred choice for detecting the presence of uncommon variables and their influence on other variables.

Depending on the kind of information we want to obtain from a given longitudinal database, we would extract induced or embedded pattern using one (or several) of the proposed representation schemes.

It should be noted that the advantage of using embedded patterns, when feasible, is that they need a lower number of nodes to collect all the information we might be interested in (i.e., they are typically smaller than the corresponding induced patterns), hence they often need less computational effort to be identified.

5 Conclusions

In this paper, we have proposed a new approach to mine longitudinal databases using tree pattern mining techniques. We have introduced four different

tree-based representation schemes and we have compared them. The deep time-based representation scheme is the most versatile, although we should resort to the timestamp or the variable-based representation schemes for specific usage scenarios. The choice between induced and embedded patterns, which is orthogonal to the representation scheme selection, will often be determined by the specific situation we try to address and it usually involves computational trade-offs.

References

1. Abe, K., Kawasoe, S., Asai, T., Arimura, H., Arikawa, S.: Efficient substructure discovery from large semi-structured data. In: Proceedings of the 2nd SIAM International Conference on Data Mining (2002)
2. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, pp. 487–499 (1994)
3. Alati, R., O'Callaghan, M., Najman, J.M., Williams, G.M., Bor, W., Lawlor, D.A.: Asthma and internalizing behavior problems in adolescence: A longitudinal study. Psychosomatic Medicine 67(3), 462–470 (2005)
4. Asai, T., Arimura, H., Uno, T., Nakano, S.-i.: Discovering Frequent Substructures in Large Unordered Trees. In: Grieser, G., Tanaka, Y., Yamamoto, A. (eds.) DS 2003. LNCS (LNAI), vol. 2843, pp. 47–61. Springer, Heidelberg (2003)
5. Chi, Y., Yang, Y., Muntz, R.R.: HybridTreeMiner: An efficient algorithm for mining frequent rooted trees and free trees using canonical form. In: Proceedings of the 16th International Conference on Scientific and Statistical Database Management. pp. 11–20 (2004)
6. Cothey, V.: A longitudinal study of world wide web users' information-searching behavior. Journal of the American Society for Information Science and Technology 53(2), 67–78 (2002)
7. Diggle, P.J., Liang, K.Y., Zeger, S.L.: Analysis of longitudinal data. Oxford Statistical Science Series, vol. 13. Clarendon Press (1994)
8. Fitzmaurice, G.M., Laird, N.M., Ware, J.H.: Applied Longitudinal Analysis. Wiley Series in Probability and Statistics (2004)
9. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1–12 (2000)
10. Harada, S., Wobbrock, J.O., Malkin, J., Bilmes, J.A., Landay, J.A.: Longitudinal study of people learning to use continuous voice-based cursor control. In: Proceedings of the 27th International Conference on Human Factors in Computing Systems, pp. 347–356 (2009)
11. Hedeker, D., Gibbons, R.D.: Longitudinal Data Analysis. Wiley-Interscience (2006)
12. Hido, S., Kawano, H.: AMIOT: Induced ordered tree mining in tree-structured databases. In: Proceedings of the 5th IEEE International Conference on Data Mining, pp. 170–177 (2005)
13. Jiménez, A., Berzal, F., Cubero, J.C.: Frequent tree pattern mining: A survey. Intelligent Data Analysis 14, 603–622 (2010)
14. Jiménez, A., Berzal, F., Cubero, J.C.: POTMiner: Mining ordered, unordered, and partially-ordered trees. Knowlegde and Information Systems 23(2), 199–224 (2010)

15. Jiménez, A., Berzal, F., Cubero, J.C.: Using trees to mine multirelational databases. In: Data Mining and Knowledge Discovery pp. 1–39 (2011), http://dx.doi.org/10.1007/s10618-011-0218-x
16. Lee, K., Mercante, D.: Longitudinal nominal data analysis using marginalized models. Computational Statistics & Data Analysis 54(1), 208–218 (2010)
17. Nijssen, S., Kok, J.N.: Efficient discovery of frequent unordered trees. In: First International Workshop on Mining Graphs, Trees and Sequences (MGTS 2003), in conjunction with ECML/PKDD 2003, pp. 55–64 (2003)
18. Raudenbush, S.W., Shing Chan, W.: Growth curve analysis in accelerated longitudinal designs. Journal of Research in Crime and Delinquency 29(4), 387–411 (1992)
19. Roberts, B.W., DelVecchio, W.F.: The rank-order consistency of personality traits from childhood to old age: A quantitative review of longitudinal studies. Psychological Bulletin 126(1), 3–25 (2000)
20. Singer, J.D., Willett, J.B.: Applied Longitudinal Data Analysis: Modeling Change and Event Occurrence. Oxford University Press (2003)
21. de Vries, H., van't Riet, J., Spigt, M., Metsemakers, J., van den Akker, M., Vermunt, J.K., Kremers, S.: Clusters of lifestyle behaviors: Results from the Dutch SMILE study. Preventive Medicine 46(3), 203–208 (2008)
22. Wang, C., Hong, M., Pei, J., Zhou, H., Wang, W., Shi, B.-L.: Efficient Pattern-Growth Methods for Frequent Tree Pattern Mining. In: Dai, H., Srikant, R., Zhang, C. (eds.) PAKDD 2004. LNCS (LNAI), vol. 3056, pp. 441–451. Springer, Heidelberg (2004)
23. Xiao, Y., Yao, J.F., Li, Z., Dunham, M.H.: Efficient data mining for maximal frequent subtrees. In: Proceedings of the 3rd IEEE International Conference on Data Mining, pp. 379–386 (2003)
24. Zaki, M.J.: Efficiently mining frequent embedded unordered trees. Fundamenta Informaticae 66(1-2), 33–52 (2005)
25. Zaki, M.J.: Efficiently mining frequent trees in a forest: Algorithms and applications. IEEE Transactions on Knowledge and Data Engineering 17(8), 1021–1035 (2005)

Mining Top-K Sequential Rules

Philippe Fournier-Viger and Vincent S. Tseng

Department of Computer Science and Information Engineering
National Cheng Kung University
philippe.fv@gmail.com, tsengsm@mail.ncku.edu.tw

Abstract. Mining sequential rules requires specifying parameters that are often difficult to set (the minimal confidence and minimal support). Depending on the choice of these parameters, current algorithms can become very slow and generate an extremely large amount of results or generate too few results, omitting valuable information. This is a serious problem because in practice users have limited resources for analyzing the results and thus are often only interested in discovering a certain amount of results, and fine-tuning the parameters can be very time-consuming. In this paper, we address this problem by proposing TopSeqRules, an efficient algorithm for mining the top-k sequential rules from sequence databases, where *k* is the number of sequential rules to be found and is set by the user. Experimental results on real-life datasets show that the algorithm has excellent performance and scalability.

1 Introduction

Nowadays, huge amounts of sequential information are stored in databases (e.g. stock market data, biological data and customer data). Discovering patterns in such databases is important in many domains, as it provides a better understanding of the data. For example, in international trade, one could be interested in discovering temporal relations between the appreciations of currencies to make trade decisions. Various methods have been proposed for mining patterns in sequential databases such as mining repetitive patterns, trends and sequential patterns (see [1] for a survey). Among them, mining sequential patterns is probably the most popular set of techniques (e.g. [2, 3, 4]). It consists of finding subsequences appearing frequently in a database. However, knowing that a sequence appear frequently in a database is not sufficient for making prediction [5]. An alternative that addresses the problem of prediction is *sequential rule mining* [5-12]. A sequential rule indicates that if some item(s) occurred, some other item (s) are likely to occur with a given confidence or probability afterward. Sequential rule mining has many applications (e.g. stock market [7], weather observation [9], drought management [10] and e-learning [5, 6]).

Sequential rule mining algorithms have been developed for discovering rules in a single sequence (e.g. [9, 12]) or in multiple sequences (e.g. [5, 6, 7, 10, 11]). To mine sequential rules, users typically have to set two parameters: (1) a minimum support threshold and (2) a minimal confidence threshold. But one important question that has not been addressed in previous research is: "How can we choose appropriate values for these parameters if we don't have any background knowledge about the data-

J. Tang et al. (Eds.): ADMA 2011, Part II, LNAI 7121, pp. 180–194, 2011.

base?" It is an important question because if these parameters are set too high, few patterns are found and algorithms have to be rerun to find more patterns, and if parameters are set too low, algorithms become incredibly slow and generate an extremely large amount of results. In practice, to find appropriate values for these parameters, people generally successively try different values by guessing and executing the algorithms over and over until being satisfied by the results, which can be very time-consuming. However, in data mining, users are often only interested in discovering the "top" patterns in a database because they have limited resources for analyzing patterns that are found [13-16]. In this paper, we address this issue for the task of mining sequential rules in sequence databases. We propose *TopSeqRules*, an algorithm for mining only the top-k sequential rules, where *k* is a parameter set by the user. This allows the user to specify for example, that he wants to discover the top 500 rules. Although several top-k pattern mining algorithms have been designed for mining patterns like frequent itemsets (e.g. [13, 14, 16]) and sequential patterns (e.g. [15]), we are the first to address the problem for sequential rules. The rest of this paper is organized as follows. Section 2 reports related work and defines the problem of top-k sequential rule mining. Section 3 describes TopSeqRules, optimizations and extensions. Then, section 4 presents the evaluation. Finally, Section 5 presents the conclusion.

2 Problem Definition and Related Work

There exist several definitions of what is sequential rule mining [5-12] (see [5] for a literature review). In this paper, we use the definition of [6] for discovering sequential rules common to multiple sequences because it has two reported real applications [6] and because not many works have addressed the case of multiple sequences, despite that it has many potential applications. According to this definition, a *sequence database* (defined as in sequential pattern mining [2]) is a set of sequences $S=\{s_1, s_2 \ldots s_s\}$ and a set of items $I=\{i_1, i_2, \ldots i_t\}$ occurring in these sequences. A *sequence* is defined as an ordered list of *itemsets* (sets of items) $s_x=I_1, I_2, \ldots I_n$ such that $I_1, I_2, \ldots I_n \subseteq I$, and where each sequence is assigned a unique *sid* (sequence id). As an example, figure 1.a depicts a sequence database containing four sequences with sids *seq1*, *seq2*, *seq3* and *seq4*. In this example, each single letter represents an item. Items between curly brackets represent an itemset. For instance, the sequence *seq1* means that items *a* and *b* occurred at the same time, and were followed successively by *c*, *f*, *g* and *e*. A *sequential rule* $X \Rightarrow Y$ is defined as a relationship between two itemsets X, $Y \subseteq I$ such that $X \cap Y = \emptyset$ and $X, Y \neq \emptyset$. Note that X and Y are unordered. The interpretation of a rule $X \Rightarrow Y$ is that if the items of X occur in a sequence, the items in Y will occur afterward in the same sequence. Formally, a rule $X \Rightarrow Y$ is said to *occur* in a sequence $s_x=I_1, I_2 \ldots I_n$ if there exists an integer *u* such that $1 \leq u < n$, $X \subseteq \bigcup_{i=1}^{u} I_i$ and $Y \subseteq \bigcup_{i=u+1}^{n} I_i$. For example, the rule $\{a, b, c\} \Rightarrow \{e, f, g\}$ occurs in the sequence $\{a, b\}$, $\{c\}$, $\{f\}$, $\{g\}$, $\{e\}$, whereas the rule $\{a, b, f\} \Rightarrow \{c\}$ does not because item *c* does not occur after *f*. A rule $X \Rightarrow Y$ is said to be of size $v*w$ if $|X| = v$ and $|Y| = w$. For example, the rules $\{a, b, c\} \Rightarrow \{e, f, g\}$ and $\{a\} \Rightarrow \{e, f\}$ are of size 3*3 and 1*2 respectively.

Furthermore, a rule of size $f*g$ is said to be *larger than* another rule of size $h*i$ if $f > h$ and $g \geq i$, or if $f \geq h$ and $g > i$. For a given sequence database and a rule X⇒Y, the notation *sids*(X⇒Y) represents the *sids set* (the set of sequence ids) of the sequences where the rule occurs. For instance, *sids*({*a*}⇒{*b*}) = {*seq2*, *seq3*}. For an itemset X and a sequence database, the notation *sids*(X) denotes the sids set corresponding to sequences where all the items of X appears. For example, *sids*({*a*, *b*, *c*}) = {*seq1*, *seq2*}. For the sake of brevity, in the rest of this paper, curly brackets will be omitted when using the "*sids"* notation with itemsets containing a single item. Two interestingness measures are defined for sequential rules, which are similar to those used in association rule mining [17]. The *support* of a rule X⇒Y is defined as sup(X⇒Y) = |*sids*(X⇒Y)| / |S|. The *confidence* is defined as conf(X⇒Y) = |*sids*(X⇒Y)| / |*sids*(X)|. The *problem of mining sequential rules common to multiple sequences* is to find all valid rules in a sequence database [6]. A *valid rule* is a rule such that its support and confidence are respectively no less than user-defined thresholds *minsup* and *minconf*. For example, figure 1.b illustrates some valid rules found in the database shown in figure 1.a for *minsup* = 0.5 and *minconf* = 0.5. Moreover, a rule having a support higher or equal to *minsup* is said to be a *frequent rule*. Thus, by definition, valid rules are a subset of frequent rules.

a)

ID	Sequences
seq1	{*a*, *b*},{*c*},{*f*},{*g*},{*e*}
seq2	{*a*, *d*},{*c*},{*b*},{*a*, *b*, *e*, *f*}
seq3	{*a*},{*b*},{*f*},{*e*}
seq4	{*b*},{*f*, *g*}

b) →

ID	Rule	Support	Conf.
r1	{*a*, *b*, *c*}⇒{*e*}	0.5	1.0
r2	{*a*}⇒{*c*, *e*, *f*}	0.5	0.66
r3	{*a*, *b*}⇒{*e*, *f*}	0.5	1.0
r4	{*b*}⇒{*e*, *f*}	0.75	0.75
r5	{*a*}⇒{*e*, *f*}	0.75	1.0
r6	{*c*}⇒{*f*}	0.5	1.0
r7	{*a*}⇒{*b*}	0.5	0.66
…	…	…	…

Fig. 1. A sequence database (left) and some sequential rule found (right)

To define an algorithm for discovering the "top-k" sequential rules, we first need to define what a "top-k sequential rule" is. In frequent pattern mining, top-k pattern mining algorithms have been defined principally for mining frequent itemsets [13, 14, 16] and sequential patterns [15]. For discovering these types of patterns only the minimum support is generally used. Consequently, the problem of mining the top-k patterns for these types of patterns is defined as discovering the *k* patterns having the highest support [13- 16]. For sequential rule mining, however, the problem of mining the top-k sequential rules could be stated in different ways because two interestingness measures are used (the support and the confidence) instead of one. We thus see two possible definitions: (I) to discover the *k* rules having the highest support such that their confidence is higher than *minconf,* (II) to discover the *k* rules having the highest confidence such that their support is higher than *minsup*.

For this paper, we choose definition I over definition II because in practice the parameter *minconf* is much easier to set than *minsup* because *minconf* represents the minimum confidence that a user want in rules, while choosing an appropriate value for *minsup* depends solely on the characteristics of the database and it is impossible to

know a priori what is an appropriate value for *minsup*. However, later on, in section 3.4 we will explain how TopSeqRules could be adapted for definitions I. Based on definition II, we define the problem of mining *the top-k sequential rules* as follows:

Definition 1 (top-k sequential rule mining): To discover in a sequence database a set L containing k rules such that for each rule $r_m \in$ L, $conf(r_m) \geq minconf$, and there does not exist a rule $r_n \notin$ L such that $sup(r_n) > sup(r_m)$ and $conf(r_n) \geq minconf$.

To mine top-k patterns, all top-k pattern mining algorithms (e.g. [13-16]) follow a same general process, although they have several differences. The *general process for mining* top-k *patterns* from a database is the following. Initially, a top-k algorithm sets the minimum interestingness criterion (e.g. *minsup*) to the lowest possible value to ensure that all the top-k patterns will be found. Then, the algorithm starts searching for patterns by using a search strategy. As soon as a pattern is found, it is added to a list of patterns L ordered by the interestingness of patterns. The list is used to maintain the top-k patterns found until now. Once k patterns are found, the value for the minimum interestingness criterion is raised to the interestingness value of the least interesting pattern in L. Raising the minimum interestingness value is used to prune the search space when searching for more patterns. Thereafter, each time a pattern is found that meets the minimum interestingness criterion, the pattern is inserted in L, the pattern(s) in L not respecting the minimum interestingness criterion anymore are removed from L, and the minimum interestingness criterion is raised to the value of the least interesting pattern in L. The algorithm continues searching for more patterns until no pattern are found by the search strategy.

What distinguish top-k pattern mining algorithms are their data structures, input, output and search strategies to discover patterns. As any other data mining algorithms, a top-k algorithm needs to use appropriate data structures and search strategies to be efficient in both memory and execution time. But besides that, the efficiency of a top-k algorithm depends largely on how fast it can raise the minimum interestingness criterion (e.g. *minsup*) to prune the search space. To raise the support quickly, it is desirable that a top-k pattern mining algorithm uses strategies to find the most interesting patterns as early as possible. Hence, to design an efficient top-k sequential rule mining algorithm, several questions have to be addressed such as "Which search strategy and data structures should be used?", and "What optimizations can be applied?"

3 The TopSeqRules Algorithm

To answer this challenge, we propose TopSeqRules, a top-k algorithm based on the search strategy of RuleGrowth for generating valid rules [5]. RuleGrowth is the current best algorithm for mining sequential rules according to the definition of section 2. Its search strategy consists of first finding rules containing only two items and then to find larger rules by recursively growing the rules by scanning the sequences containing them to find single items that could expand their left or right parts. These two processes for expanding rules are named *left expansion* and *right expansion*. TopSeqRules integrates these processes with the *general process for mining* top-k *patterns*

described in section 2. Furthermore, to mine the top-k rules efficiently, it also add optimizations and the strategy of always trying to generate the most promising rules first, to try to prune the search space quickly by raising *minsup*.

Before presenting TopSeqRules, we introduce preliminary definitions and properties related to left/right expansions. A *left expansion* is the process of adding an item i to the left side of a rule X⇒Y to obtain a larger rule X∪{i}⇒Y. Similarly, a *right expansion* is the process of adding an item i to the right side of a rule X⇒Y to obtain a rule X⇒Y∪{i}. Left/right expansions have four important properties.

Property 1 (left expansion, effect on support): If an item i is added to the left side of a rule *r:*X⇒Y, the support of the resulting rule *r'*:X∪{i}⇒Y can only be lower or equal to sup(*r*). **Proof:** The support of *r* and *r'* are respectively |*sids*(X⇒Y)| / |S| and |*sids*(X∪{i}⇒Y)| / |S|. Since |*sids*(X⇒Y)| ≥ |*sids*(X∪{i}⇒Y)|, sup(*r*) ≥ sup(*r'*).

Property 2 (right expansion, effect on support): If an item i is added to the right side of a rule *r:*X⇒Y, the support of the resulting rule *r'*:X⇒Y∪{i} can only be lower or equal to sup(*r*). **Proof:** The support of *r* and *r'* are respectively |*sids*(X⇒Y)| / |S| and |*sids*(X⇒Y∪{i})| / |S|. Since |*sids*(X⇒Y)| ≥ |*sids*(X⇒Y∪{i})|, sup(*r*) ≥ sup(*r'*).

Properties 1 and 2 imply that the support is monotonic with respect to left/right expansions. In other words, performing any combinations of left/right expansions of a rule can only result in rules having a support that is lower or equal to the original rule. Therefore, all the frequent can be found by recursively performing expansions on frequent rules of size 1*1. Moreover, property 1 and 2 guarantee that expanding a rule having a support less than *minsup* will not result in a frequent rule. The confidence is not monotonic with respect to expansions, as next properties demonstrate.

Property 3 (left expansion, effect on confidence): If an item i is added to the left side of a rule *r*:X⇒Y, the confidence of the resulting rule *r'*: X∪{i}⇒Y can be lower, higher or equal to the confidence of *r*. **Proof:** The confidence of *r* and *r'* are respectively |*sids*(X⇒Y)| / |*sids*(X)| and |*sids*(X∪{i}⇒Y)| / |*sids*(X∪{i})|. Because |*sids*(X⇒Y)| ≥ |*sids*(X∪{i}⇒Y)| and |*sids*(X)| ≥ |*sids*(X∪{i})|, conf(*r*) can be lower, higher or equal to conf(*r'*).

Property 4 (right expansion, effect on confidence): If an item i is added to the right side of a rule *r:*X⇒Y, the confidence of the resulting rule *r'*: X⇒Y∪{i} is lower or equal to the confidence of *r*. **Proof:** The confidence of *r* and *r'* are respectively |*sids*(X⇒Y)| / |*sids*(X)| and |*sids*(X⇒Y∪{i})| / |*sids*(X)|. Since |*sids*(X⇒Y)| ≥ |*sids*(X⇒Y∪{i})|, conf(*r*) ≥ conf(*r'*).

TopSeqRules relies on sids sets to calculate the support and confidence of rules obtained by left or right expansions. Sids sets have two important properties.

Property 5 (sids set of a rule and its itemsets): For any sequential rule X⇒Y, *sids*(X⇒Y) ⊆ *sids*(X) ∩ *sids*(Y). **Proof:** A rule can only occur in a sequence if all items from its left and right parts appear in it.

Property 6 (sids set of a rule obtained by left/right expansion): For any sequential rule *r'* obtained by a left or right expansion of a rule *r*, the relationship *sids*(*r'*) ⊆

sids(*r*) holds. **Proof**. If the rule *r* does not occur in a sequence, the rule *r'* also cannot. Therefore, the sids set of *r'* must be a subset of the sids set of *r*.

3.1 The Algorithm

TopSeqRules takes as input a sequence database *S*, a number *k* of rules that the user wants to discover, and the *minconf* threshold. The algorithm uses three main internal variables. The first one is *minsup,* wich is initially set to 0 and is raised dynamically as soon as *k* rules are found, as it will be explained. The second variable is a set named L to keep the top-k rules found until now that have a support and confidence higher or equals to *minsup* and *minconf*. The third variable is a set named R to store the rules that should be expanded to have a chance of finding more valid rules.

The Main Procedure. The main procedure of TopSeqRules is shown in figure 2. The algorithm first scans the database once to calculate *sids*(*c*) for each item *c*. Then, the algorithm generates all valid rules of size 1*1. This is done by taking each pair of items *i, j*, where *i* and *j* each have at least *minsup*×|S| sids (if this condition is not met, no rule having at least the minimum support can be created with *i* and *j*). The algorithm then scans sequences in *sids*(*i*) ∩ *sids*(*j*) to calculate *sids*($i \Rightarrow j$) and *sids*($j \Rightarrow i$), the sids of sequences where the rule $\{i\} \Rightarrow \{j\}$ and $\{j\} \Rightarrow \{i\}$ occur, respectively (because of property 5). After this, the support of the rule $\{i\} \Rightarrow \{j\}$ is obtained by dividing |*sids*($i \Rightarrow j$)| by |S|. For each rule $\{i\} \Rightarrow \{j\}$ or $\{j\} \Rightarrow \{i\}$ that is valid, the procedure SAVE is called with the rule and L as parameters so that the rule is recorded in the set L of the current top-k rules found. Also, each rule $\{i\} \Rightarrow \{j\}$ or $\{j\} \Rightarrow \{i\}$ that is frequent is added to the set R, to be later considered for expansion.

After that, a loop is performed to recursively select the rule *r* with the highest support in R such that *sup*(*r*) ≥ *minsup* and expand it. The idea behind this loop is to always expand the rule from R having the highest support first because it is more likely to generate rules having a high support and thus to allow to raise *minsup* more quickly for pruning the search space. The loop terminates when there is no more rule in R having a support higher or equal to *minsup*. For expanding a rule, a flag *expandLR* indicates if the rule should be left and right expanded by calling the procedure EXPAND-L and EXPAND-R or just left expanded by calling EXPAND-L. For all rules of size 1*1, this flag is set to true. The utility of this flag for larger rules will be explained later.

The Save Procedure. The role of SAVE (figure 3) is to raise *minsup* and update the list L when a new valid rule *r* is found. The first step of SAVE is to add the rule *r* to L. Then, if L contains more than *k* rules and the support is higher than *minsup*, rules from L that have exactly the support equal to *minsup* can be removed until only *k* rules are kept. Finally, *minsup* is raised to the support of the rule in L having the lowest support. By this simple scheme, the top-k rules found are maintained in L.

Now that we have described how rules of size 1*1 are generated and the mechanism for maintaining the top-k rules in L, we explain how rules of size 1*1 are expanded to find larger rules. Without loss of generality, we can ignore the top-k aspect for the explanation and consider the problem of generating all valid rules. To recursively expand rules and find all valid rules starting from rules of size 1*1, a few problems had to be solved.

Problem 1: How can we guarantee that all valid rules are found by recursively performing left/right expansions starting from rules of size 1*1? The answer is found in properties 1 and 2, which states that the support of a rule is monotonic with respect to left/right expansions. This implies that all rules can be discovered by recursively performing expansions starting from frequent rules of size 1*1. Moreover, these properties imply that infrequent rules should not be expanded because they will not lead to valid rules. However, no similar pruning can be done for the confidence because it is not monotonic with respect to left expansion (property 3).

Problem 2: How we can guarantee that no rules are found twice by recursively making left/right expansions? To guarantee this, two sub-problems had to be solved. First, if we grow rules by performing expansions recursively, some rules can be found by different combinations of left/right expansions. For example, consider the rule $\{a, b\} \Rightarrow \{c, d\}$. By performing, a left and then a right expansion of $\{a\} \Rightarrow \{c\}$, one can obtain the rule $\{a, b\} \Rightarrow \{c, d\}$. But this rule can also be obtained by performing a right and then a left expansion of $\{a\} \Rightarrow \{c\}$. A simple solution to avoid this problem is to forbid performing a right expansion after a left expansion but to allow performing a left expansion after a right expansion. An alternative solution is to not allow a left expansion after a right expansion.

TOPSEQRULES (S, *k, minconf*)
1. R := Ø. L := Ø. *minsup* := 0.
2. **Scan** the database S once. Record the sids set of each item *c* in a variable *sids*(c).
3. **FOR** each pairs of items *i, j* such that |*sids(i)*|≥ *minsup* and ∩ |*sids(j)*|≥ *minsup:*
4. *sids(i⇒j)* := Ø. *sids(j⇒i)* := Ø.
5. **FOR** each sid *s* ∈ (*sids(i)* ∩ *sids(j)*:
6. **IF** *i* occurs before *j* in *s* **THEN** *sids(i⇒j)* := *sids(i⇒j)* ∪ {*s*}.
7. **IF** *j* occurs before *i* in *s* **THEN** *sids(j⇒i)* := *sids(j⇒ i)* ∪ {*s*}.
8. **END FOR**
9. **IF** |*sids(i⇒j)*| / |S| ≥ *minsup* **THEN**
10. *conf*({i}⇒{j}) := |*sids(i⇒j)*| / | *sids*(*i*).
11. **IF** *conf*({i}⇒{j}) ≥ *minconf* **THEN SAVE**({i}⇒{j}, L, *k, minsup*).
12. **Set flag** expandLR **of** {i}⇒{j}**to true**.
13. R := R∪{{i}⇒{j}}.
14. **END IF**
… [lines 9 to 14 are repeated here with *i* and *j* swapped] …
15. **END FOR**
16. **WHILE** ∃r ∈ R **AND** *sup*(r) ≥ *minsup* **DO**
17. **Select** the rule *rule* having the highest support in R
18. **IF** *rule*.expandLR = true **THEN**
19. **EXPAND-L**(*rule*, L, R, *k*, *minsup, minconf*).
20. **EXPAND-R**(*rule*, L, R, *k*, *minsup, minconf*).
21. **ELSE EXPAND-R**(*rule*, L, R, *k*, *minsup, minconf*).
22. **REMOVE** *rule* from R. **REMOVE** from R all rules r ∈ R | *sup*(r) <*minsup*.
23. **END WHILE**

Fig. 2. The TopSeqRules algorithm

The second sub-problem is that rules can be found several times by performing left/right expansions with different items. For example, consider the rule $\{b, c\} \Rightarrow \{d\}$.

```
SAVE(r, R, k, minsup)
1.  L := L∪{r}.
2.  IF |L| ≥ k THEN
3.      IF sup(r) > minsup THEN
4.          WHILE |L| > k AND ∃s ∈ L | sup(s) = minsup
5.              REMOVE s from L.
6.      END IF
7.      Set minsup to the lowest support of rules in L.
8.  END IF
```

Fig. 3. The SAVE procedure

A left expansion of $\{b\}\Rightarrow\{d\}$ with item c can result in the rule $\{b, c\}\Rightarrow\{d\}$. But that latter rule can also be found by performing a left expansion of $\{c\}\Rightarrow\{d\}$ with b. To solve this problem, a solution is to only add an item to an itemset of a rule if the item is greater than each item in the itemset according to the lexicographic ordering. In the previous example, this would mean that item c would be added to the left itemset of $\{b\}\Rightarrow\{d\}$. But b would not be added to the left itemset of $\{c\}\Rightarrow\{d\}$ because b is not greater than c. By using this strategy and the previous one, no rules are found twice. We now explain how EXPAND-L and EXPAND-R have been implemented based on these strategies.

The EXPAND-R Procedure. The procedure EXPAND-R (cf. figure 4) takes as parameters a rule I⇒J to be expanded, L, R, k, *minsup* and *minconf*. To expand I⇒J, EXPAND-R has to identify items that can expand the rule I⇒J to produce a valid rule. By exploiting the fact that any valid rule is a frequent rule, this problem is decomposed into two sub-problems, which are (1) determining items that can expand a rule I⇒J to produce a frequent rule and (2) assessing if a frequent rule obtained by an expansion is valid. The first sub-problem is solved as follows. To identify items that can expand a rule I⇒J and produce a frequent rule, the algorithm scans each sequence *sid* from *sids*(I∩J). During this scan, for each item $c\notin$ I appearing in sequence *sid* after I, the algorithm adds *sid* to a variable *sids*(I⇒J∪{c}) if c is lexically larger than all items in J (this latter condition is to ensure that no duplicated rules will be generated, as explained). When the scan is completed, for each item c such that $|sids(I\Rightarrow J\cup\{c\})| / |S| \geq minsup$, the rule I⇒J∪{$c$} is deemed frequent and is added to the set R so that it will be later considered for expansion. Note that the flag *expandLR* of each such rule is set to *false* so that each generated rule will only be considered for right expansions (to make sure that no rules are found twice by different combinations of left/right expansions, as explained). Finally, the confidence of each frequent rule I⇒J∪{c} is calculated to see if the rule is valid, by dividing $|sids(I\Rightarrow J\cup\{c\})|$ by $|sids(I)|$, the value *sids*(I) having already been calculated for I⇒J. If the confidence of I⇒J∪{c} is no less than *minconf,* then the rule is valid and the procedure SAVE is called to add the rule to L, the list of the current top-k rules.

The EXPAND-L Procedure. The procedure EXPAND-L (cf. figure 5) takes as parameters a rule I⇒J to be expanded, L, R, k, *minsup* and *minconf*. This procedure is similar to EXPAND-R. The only extra step that is performed compared to EXPAND-R is that for each rule I∪{c}⇒J obtained by the expansion of I⇒J with an item c, the value *sids*(I∪{c}) necessary for calculating the confidence is obtained by intersecting *sids*(I) with *sids*(c). The value sids(c) is known from the initial database scan.

3.2 Implementing TopSeqRules Efficiently

We implemented TopSeqRules in Java. We used two optimizations, which greatly improve TopSeqRules' execution time in our experiments (cf. section 4).

Optimization 1: Implementing L and R with Efficient Data Structures. TopSeqRules performs three operations on L, which are insertion, deletion and finding the rule having the lowest support. The three same operations are performed on R plus finding the rule having the highest support (to select the most promising rules from R). Because these operations are performed constantly by TopSeqRules, it is important to implement L and R with data structures that support performing these operations efficiently. To address this issue, we implement L with a *Fibonacci heap* sorted by the support of the rules. It has an amortized time cost of $O(1)$ for insertion and obtaining the minimum, and $O(log(n))$ for deletion [18]. For R, we used a *red-black tree* because we also need the operation of finding the maximum. A red-black tree guarantees a $O(log(n))$ worst-case time cost for the four operations [18] .

Optimization 2: Merging Database Scans for the Left/Right Expansions of a Rule. The second optimization reduces the number of database scans. Recall that EXPAND-L and EXPAND-R are both applied to each rule I⇒J having the flag *expandLR* set to true. Performing EXPAND-L and EXPAND-R each requires to scan each sequence from *sids*(I⇒J) once. A simple optimization is to combine the database

EXPAND-R(I⇒J, L, R, *k*, *minsup, minconf*)
1. **FOR** each *sid* ∈ *sids*(I⇒J), scan the sequence *sid*. For each item *c* appearing in sequence *sid* that is lexically larger than all items in J and appear after I, record *sid* in a variable *sids*(I⇒J∪{*c*}).
2. **FOR** each item *c* such that |*sids*(I⇒J∪{*c*})| ≥ minsup×|S| :
3. **Set flag** expandLR **of** I⇒J∪{c} **to false**.
4. R := R∪{I⇒J∪{c}}.
5. **IF** | *sids* (I⇒J∪{*c*})| / | *sids* (I)| ≥ *minconf* **THEN** SAVE(I⇒J∪{*c*}, L, *k, minsup*).
6. **END FOR**
7. **END FOR**

Fig. 4. The EXPAND-R procedure

EXPAND-L(I⇒J, L, R, *k*, *minsup, minconf*)
1. **FOR** each *sid* ∈ *sids*(I⇒J), scan the sequence *sid*. For each item c∉J appearing in sequence *sid* that is lexically larger than all items in I and appear before J, record *sid* in a variable *sids*(I∪{c}⇒J)
2. **FOR** each item *c* such that |*sids* (I∪{*c*}⇒J)| / |S| ≥ *minsup* :
3. **Set flag** expandLR **of** I⇒J∪{c} **to true**.
4. *sids*(I∪{*c*}) := Ø.
5. **FOR** each *sid* ∈ *sids*(I) such that *sid* ∈ *sids*(*c*), *sids*(I∪{*c*}):= *sids*(I∪{*c*}) ∪{*sid*}.
6. SAVE(I∪{*c*}⇒J, L, *k, minsup*).
7. **IF** |*sids*(I∪{*c*}⇒J)| / |*sids*(I∪{*c*})| ≥ *minconf* **THEN** R := R∪{I∪{*c*}⇒J}.
8. **END FOR**

Fig. 5. The EXPAND-L procedure

scans of EXPAND-L and EXPAND-R so that they use the same database scan for identifying left and right expansions, when the flag *expandLR* is set to true.

3.3 Extensions

The TopSeqRules algorithm can be extended in several ways. We list two.

Extension 1: Using a different definition of what is a top-k sequential rule. In section 2, we presented two possible definitions of what is a top-k sequential rule, and selected definition I for presenting the algorithm. However, TopSeqRules could easily be modified so that the support is fixed instead of the confidence for finding the top-k rules. This would result in a mining algorithm for definition II. However, the resulting algorithm would be inefficient unless the support is set high, because the confidence could not be raised dynamically to prune the search space because the confidence is not monotonic with respect to left/right expansions (cf. section 3).

Extension2: Using different interestingness measures. This paper considered the confidence and support because they are the standard measures for sequential rules. But other measures could be used. For example, more than twenty interestingness measures have been proposed for association rule mining [17]. Many of those could be adapted for sequential rule mining and integrated in the TopSeqRules algorithm because the calculation is done similarly to the calculation of the confidence and support. For example, the *lift* [17] could be adapted for sequential rules as *lift*(I⇒J) = *sup*(I⇒J)/ (sup(I) × sup(J)) for a sequential rule I⇒J. Compared to the confidence, using the *lift* would just require to calculate *sup*(J) for each rule in addition to *sup*(I). However, to be able to prune the search space, it is necessary that the "top-k" condition is defined on a monotonic interestingness measure like the support. For example, with just a few modifications, one could mine the *k* rules having a support higher or equal to *minsup* such that their *lift* is no less than a *minlift* threshold.

4 Evaluation

We evaluated TopSeqRules on a notebook with a 2.53 Ghz processor, Windows XP and 1 GB of free RAM. Experiments were carried on three real-life datasets representing three types of data. Table 1 summarizes the characteristics of the datasets. *BMSWebview1* was downloaded from http://fimi.ua.ac.be/data/. *Sign* was downloaded from http://cs-people.bu.edu/panagpap/Research/asl_mining.htm. S*nake* was obtained from the authors of [4].

Table 1. Datasets characteristics

Datasets	|S|	|I|	Avg. item count / sequence	Type of data
BMSWebView1	59601	497	2.5 (σ = 4.85)	click-stream from web store
Sign	730	310	93.39 (σ = 4.59)	language utterances
Snake	163	20	60.61 5 (σ = 0.89)	protein sequence

4.1 Influence of *k*

The first experiment was done to evaluate the influence of *k* on the execution time and the memory consumption. We ran TopSeqRules with *minconf = 0.3* on each dataset while varying *k* from 500 to 5000. Results are shown in figure 6. Our first observation is that the execution time and the maximum memory consumption is excellent for these real-life datasets (in the worst case, the algorithm took a little less than 1 minute to terminate and used about 1 gigabyte of memory). Furthermore, it can be seen that the algorithm performance and memory usage grows linearly with *k*. The only exception is for *k*=3000 to *k*=4500 for the *BMS-WebView1* dataset where the memory usage remains the same and the execution time increases more quickly. We found that this is not caused by the algorithm design. But it is caused by the Java garbage collection mechanism overhead when the memory usage is close to the 1GB limit set for the experiment.

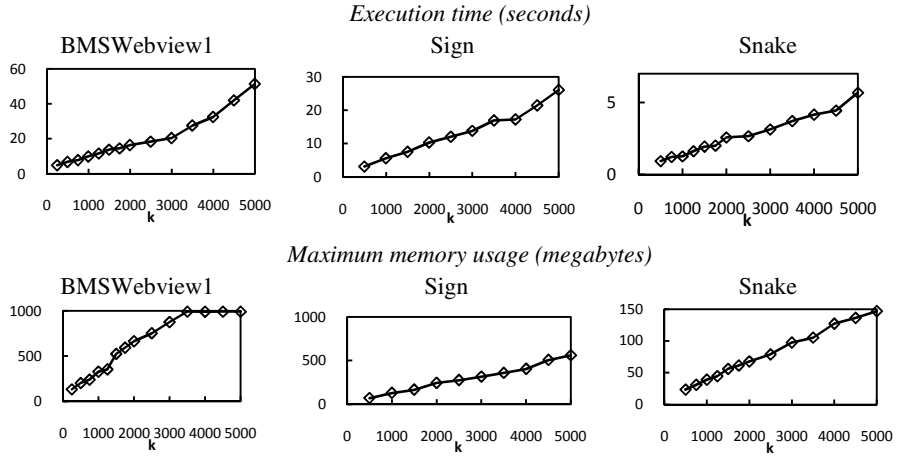


Fig. 6. Results of varying *k*

4.2 Influence of *minconf*

In a second experiment, we tested the influence of *minconf* on the execution time and memory consumption. We ran TopSeqRules on the same datasets with *k* = 1000 while varying *minconf* to observe its influence on the execution time and the memory usage. Results are shown in figure 7. It can be seen that as the confidence increases, the execution time and memory usage increase in an exponential manner. The reason is that setting the confidence higher means that the algorithm has to generate more rules to be able to raise the minimum support threshold when searching for the top-k sequential rules. Nevertheless, the algorithm ran successfully with high *minconf* thresholds in our experiment within the memory limit (up to 0.7, 0.85 and 0.99, respectively for *BMSWebview1, Sign* and *Snake*).

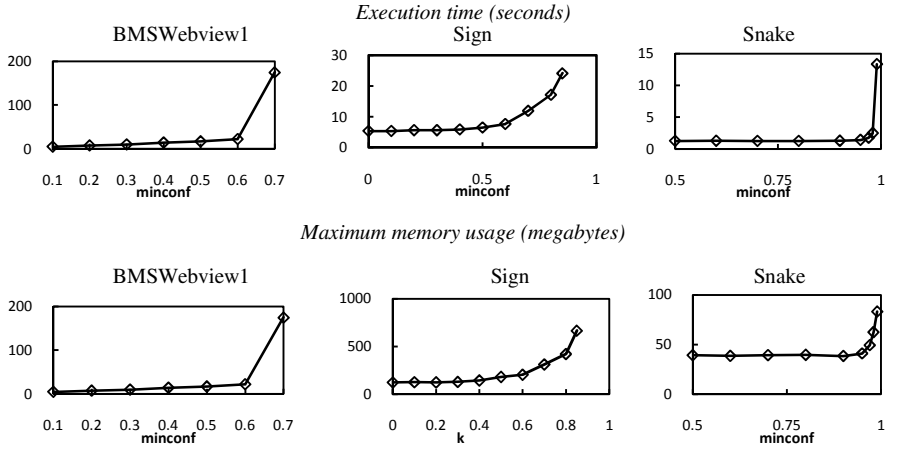


Fig. 7. Results of varying *minconf*

4.3 Influence of |S|

In a third experiment, we tested the scalability with respect to the number of sequences, by applying TopSeqRules with *minconf*=0.3 and *k*=1000 on 50% to 100 % of the sequences of each dataset. Figure 8 shows the results. It can be seen that the execution time and memory usage increase slowly when the number of sequences increases. This is because the performance of TopSeqRules depends more on the number of rules generated and stored in R than the number of sequences, and the number of rules generated remains more or less the same when the database size increase. This shows that TopSeqRules has an excellent scalability.

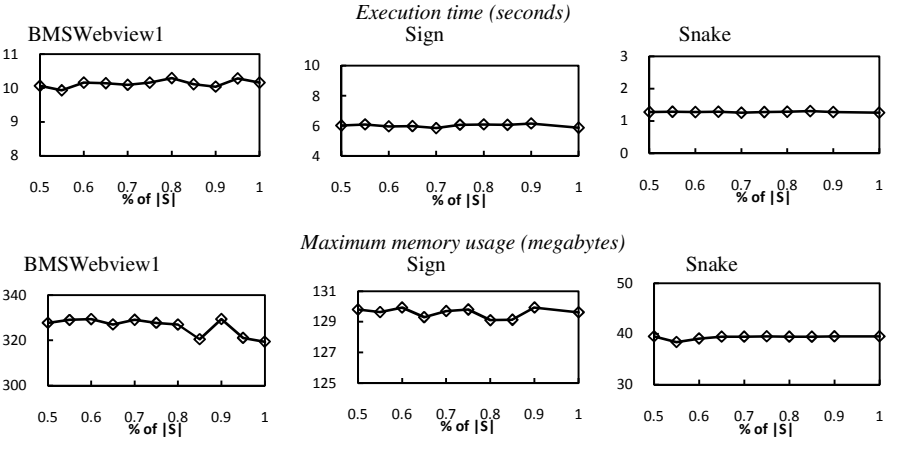


Fig. 8. Results of varying the database size

4.4 Performance Comparison

In a fourth experiment, we compared the performance of TopSeqRules with RuleGrowth [6] (also implemented in Java), the state of the art algorithm for the problem of mining sequential rules presented in section 2. To compare their performance, we first considered the scenario where the user would choose the optimal parameters for RuleGrowth to produce the same amount of result produced by TopSeqRules. For this scenario, we ran TopSeqRules on the three datasets with the parameters used in section 4.1. We then ran RuleGrowth with *minsup* equals to the lowest support for the rules found by TopSeqRules, for each *k* and each dataset. Results are shown in figure 10. It can be observed that the execution time of TopSeqRules is generally close to RuleGrowth's execution time except for *BMSWebView1* where the gap is larger. But the main difference between the performance of TopSeqRules and RuleGrowth is in the memory usage. TopSeqRules uses more memory because it keeps the set R of rules to be expanded into memory. For this reason, as *k* is set to larger value, the memory requirement of TopSeqRules increases. These results are excellent considering that the parameters of RuleGrowth were chosen optimally, which is rarely the case in real-life if the user has no a priori knowledge of the database. If the parameters of RuleGrowth are not chosen optimally, it can run much slower than TopSeqRules, or generate too few or too many results. For example, consider the case where the user wants to discover the top 1000 rules from a database and do not want to find more than 2000 rules. To find this amount of rules, the user needs to choose *minsup* from a very narrow range of values (shown in Table 2 for each dataset). For example, for *BMSWebView1*, the range of *minsup* values that will satisfy the user is 0.0011 to 0.0009. This means that a user having no a priori knowledge of the database has only a 0.02 % chance of selecting a *minsup* value that will make him satisfied. If the users choose a higher *minsup*, not enough rules will be found, and if *minsup* is set lower, too many rules will be found and the algorithm may become slow. This clearly shows the benefits of using TopSeqRules.

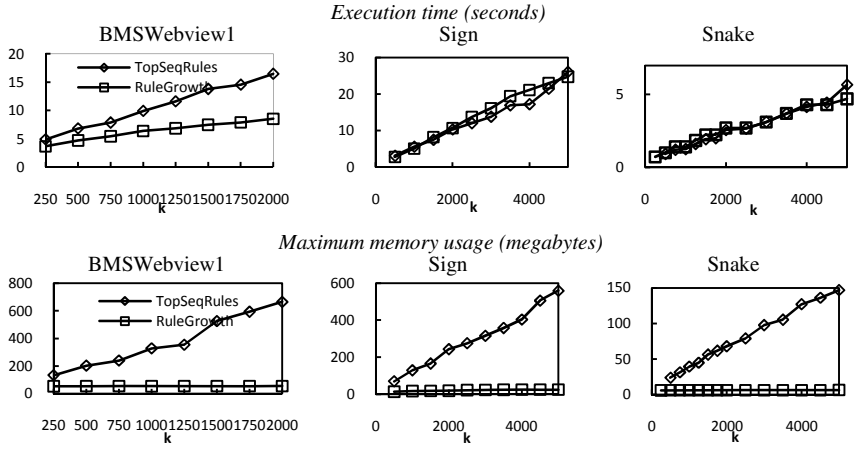


Fig. 9. Performance comparison for optimal parameters selection

Table 2. Interval of *minsup* values to find the top 1000 to 2000 rules for each dataset

Datasets	*minsup* for k=1000	*minsup* for k=2000	interval size
BMSWebView1	0.0011	0.0009	0.0002
Sign	0.420	0.384	0.036
Snake	0.960	0.944	0.016

4.5 Influence of Optimizations and of Expanding the Most Promising Rules First

Lastly, we evaluated the benefit of using the optimizations and of expanding the most promising rules first. Due to space limitations, we do not show the results as charts. We observed that *optimization 1* (data structures) and *optimization 2* (merging database scans) each reduce the execution time by about 20% to 40 % on all datasets. For the strategy of expanding the most promising rules first with the set R, we found that if this strategy is deactivated, the algorithm cannot terminate within 1 hour on all datasets because the algorithm cannot prune the search space efficiently. This is because in the worst case a top-k algorithm has to explore the whole search space before finding the top-k rules. If there are d items, the number of rules to consider is in the worst case $\sum_{k=1}^{d-1}\left[\binom{d}{k}\times\sum_{j=1}^{d-k}\binom{d-k}{j}\right]=3^d-2^d+1$, which is exponential. For this reason, it is necessary to use the set R to expand the most promising rules first to try to raise *minsup* as fast as possible and prune the search space.

5 Conclusion

Mining sequential rules requires specifying parameters (e.g. the minimal confidence and the minimal support) that are difficult to set. To address this problem, we propose an efficient algorithm named TopSeqRules that let the user specify *k,* the amount of sequential rules to be output. Experimental results with real-life datasets show that the algorithm has excellent scalability, that its execution time linearly increases with the parameter *k* and that the algorithm had no problem running in reasonable time and memory limits for *k* values of up to 5000 for all datasets. Results also show that when parameters are chosen optimally, RuleGrowth can be slightly faster than TopSeqRules. However, if *minsup* is set higher than a narrow range of values, RuleGrowth generates too few results and if it is set lower, it generates too many results and can become much slower. This clearly shows the benefit of using TopSeqRules when the user has no a priori knowledge about a sequence database.

References

1. Laxman, S., Sastry, P.: A survey of temporal data mining. Sadhana 3, 173–198 (2006)
2. Agrawal, R., Srikant, R.: Mining Sequential Patterns. In: Proc. ICDE 1995, pp. 3–14 (1995)

3. Zaki, M.J.: SPADE: An Efficient Algorithm for Mining Frequent Sequences. Machine Learning 42(1/2), 31–60 (2001)
4. Jonassen, I., Collins, J.F., Higgins, D.G.: Finding flexible patterns in unaligned protein sequences. Protein Science 4(8), 1587–1595 (1995)
5. Fournier-Viger, P., Nkambou, R., Tseng, V.S.: RuleGrowth: Mining Sequential Rules Common to Several Sequences by Pattern-Growth. In: Proc. SAC 2011, pp. 954–959 (2011)
6. Fournier-Viger, P., Faghihi, U., Nkambou, R., Mephu Nguifo, E.: CMRules: Mining Sequential Rules Common to Several Sequences. Knowledge-based Systems 25(1), 63–76 (2012)
7. Das, G., Lin, K.-I., Mannila, H., Renganathan, G., Smyth, P.: Rule Discovery from Time Series. In: Proc. ACM SIGKDD 1998, pp. 16–22 (1998)
8. Deogun, J.S., Jiang, L.: Prediction Mining – An Approach to Mining Association Rules for Prediction. In: Ślęzak, D., Yao, J., Peters, J.F., Ziarko, W.P., Hu, X. (eds.) RSFDGrC 2005, Part II. LNCS (LNAI), vol. 3642, pp. 98–108. Springer, Heidelberg (2005)
9. Hamilton, H.J., Karimi, K.: The TIMERS II Algorithm for the Discovery of Causality. In: Ho, T.-B., Cheung, D., Liu, H. (eds.) PAKDD 2005. LNCS (LNAI), vol. 3518, pp. 744–750. Springer, Heidelberg (2005)
10. Harms, S.K., Deogun, J.S., Tadesse, T.: Discovering Sequential Association Rules with Constraints and Time Lags in Multiple Sequences. In: Hacid, M.-S., Raś, Z.W., Zighed, D.A., Kodratoff, Y. (eds.) ISMIS 2002. LNCS (LNAI), vol. 2366, pp. 432–441. Springer, Heidelberg (2002)
11. Lo, D., Khoo, S.-C., Wong, L.: Non-redundant sequential rules – Theory and algorithm. Information Systems 34(4-5), 438–453 (2009)
12. Mannila, H., Toivonen, H., Verkano, A.I.: Discovery of frequent episodes in event sequences. Data Mining and Knowledge Discovery 1(1), 259–289 (1997)
13. Wang, J., Han, J., Lu, Y., Tzvetkov, P.: TFP: An Efficient Algorithm for Mining Top-k Frequent Closed Itemsets. IEEE TKDE 17(5), 652–664 (2005)
14. Pietracaprina, A., Vandin, F.: Efficient Incremental Mining of top-K Frequent Closed Itemsets. In: Corruble, V., Takeda, M., Suzuki, E. (eds.) DS 2007. LNCS (LNAI), vol. 4755, pp. 275–280. Springer, Heidelberg (2007)
15. Tzvetkov, P., Yan, X., Han, J.: TSP: Mining top-k closed sequential patterns. Knowledge and Information Systems 7(4), 438–457 (2005)
16. Chuang, K.-T., Huang, J.-L., Chen, M.-S.: Mining top-k frequent patterns in the presence of the memory constraint. VLDB 17(5), 1321–1344 (2008)
17. Tan, P.-N., Kumar, V., Srivastava, J.: Selecting the right objective measure for association analysis. Information Systems 29(4), 293–313 (2004)
18. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd edn. MIT Press (2009)

Mining Uncertain Data Streams Using Clustering Feature Decision Trees

Wenhua Xu[1], Zheng Qin[2], Hao Hu[2], and Nan Zhao[2]

[1] Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
xwh07@mails.tsinghua.edu.cn
[2] School of Software, Tsinghua University, Beijing 100084, China
{qinzh,haoh09,zhaonan}@mails.tsinghua.edu.cn

Abstract. During the last decade, classification from data streams is based on deterministic learning algorithms which learn from precise and complete data. However, a multitude of practical applications only supply approximate measurements. Usually, the estimated errors of the measurements are available. The development of highly efficient algorithms dealing with uncertain examples has emerged as an new direction. In this paper, we build a CFDTu model from data streams having uncertain attribute values. CFDTu applies an uncertain clustering algorithm that scans the data stream only once to obtain the sufficient statistical summaries. The statistics are stored in the Clustering Feature vectors, and are used for incremental decision tree induction. The vectors also serve as classifiers at the leaves to further refine the classification and reinforce any-time property. Experiments show that CFDTu outperforms a purely deterministic method in terms of accuracy and is highly scalable on uncertain data streams.

1 Introduction

Most available approaches in classifying data streams focused on handling precise and deterministic data examples [4,7,10,16,24]. However, data streams are often inaccurate due to many reasons. The uncertainty in data streams should be processed carefully. Otherwise, the induced model could be unreliable and less practical. Therefore, the development of highly efficient algorithms dealing with uncertain examples becomes a hot topic in stream data mining literature.

The uncertainty is basically referred to as the level of imprecise which can be quantified in some way [2]. It may exist in attribute values as well as in class labels of the examples. In this paper, we will focus on the examples whose attribute values are uncertain while the class labels are deterministic.

The uncertainty of attribute values is commonly specified in the form of probability density functions (PDF). The PDF can be related to an individual attribute of all examples, or related to an individual attribute of each example. However, the entire PDF are hardly available in real applications, and are usually inserted only as a modeling assumption. Therefore, a more practical and

J. Tang et al. (Eds.): ADMA 2011, Part II, LNAI 7121, pp. 195–208, 2011.

flexible assumption is adopted which assumes that the standard errors of the attribute values are known. By exploiting such information, the quality of the mining results are hopefully improved.

Chapelle et al. proposed that if two examples are in the same cluster, they are likely to be of the same class [9]. The assumption should be reasonable on the basis of the sheer existence of classes. If there is a densely populated continuum of objects, it may seem unlikely that they were ever distinguished into different classes. Moreover, the assumption does not imply that each class forms a single compact cluster. It addresses that usually, one does not observe objects of two distinct classes in the same cluster.

Motivated by the above assumption, we propose CFDTu algorithm learning from uncertain data streams. Very Fast Decision Tree (VFDT) is one of the most successful and prominent algorithms specifically designed for classifying deterministic data streams [10]. CFDTu applies an efficient uncertain clustering algorithm on VFDT that scans the example only once to obtain the sufficient statistical summaries. The statistics are stored in the form of micro-clusters, and are used for incremental decision tree induction. The micro-clusters also serve as classifiers at the leaves to further refine the classification results.

We have three contributions. First, we propose an efficient and practical classification algorithm for uncertain data streams represented by the standard error model. To the best of our knowledge, there are no other classification methods that apply standard error model. Second, we provide functional tree leaves to VFDT which can improve its accuracy and reinforce any-time property. Experiments on both synthetic and real-world datasets show that CFDTu outperforms a purely deterministic method in terms of accuracy and is highly scalable on uncertain data streams.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 provides an overview of VFDT and describes how to build CFDTu. Section 4 demonstrates experimental results on synthetic and real-world datasets. We present conclusions in section 5.

2 Related Works

Our work is related to stream data classification algorithms and uncertain data classification techniques, so we briefly discuss both of them.

The problem of stream data classification is that: given an infinite amount of continuous data examples, how to model them in order to capture their trends and patterns, and make time-critical predictions [24]. A stream data classification algorithm must meet several requirements different from the traditional setting [5,21]. First, process one example at a time, and inspect it at most once. Second, use a limited amount of memory. Third, work in a limited amount of time. Fourth, be ready to classify at any time. This is the so-called anytime property, which indicates that the induction model is ready to be applied at any point between training examples.

There are two main strategies: single model classification and ensemble classification. Single model classification techniques incrementally update the models

with new data during the training. VFDT is an incremental decision tree algorithm and represents current state-of-the-art for classifying high-speed data streams. A number of extensions of VFDT have been proposed, such as Concept-adapting Very Fast Decision Tree (CVFDT) [13], VFDTc [11], Hoeffding Option Tree [18], Adaptive Hoeffding Option Tree [5], Hoeffding Perceptron Tree [6], etc. Single model techniques are usually difficult to cope with concept drift. To overcome the weakness, some ensemble techniques have been proposed [5,7,16,24]. These ensemble approaches can be more efficiently built than updating a single model and they usually achieve better accuracy.

Some previous work focused on building classification models on uncertain data examples. All of them assume that the PDF of attribute values are known. Bi et al. presented a general statistical framework to build a SVM classification model on input data corrupted with uncertainty [3]. Qin et al. proposed a rule-based model eRule for classifying uncertain data [19]. Tsang et al. developed an uncertain decision tree (UDT) for uncertain data [23]. Qin et al. developed another type of decision tree DTU for uncertain data classification [20]. Ge et al. proposed a neural network method for classifying uncertain data [12].

Since the uncertainty is prevalent in data streams, the research of classification is dedicated to uncertain data streams nowadays. Unfortunately, only a few algorithms are available. Pan et al. proposed two types of ensemble classification algorithms, static classifier ensemble (SCE) and dynamic classifier ensemble (DCE), for mining uncertain data streams [17]. Liang et al. proposed a CVFDT based decision tree named UCVFDT for uncertain data streams [15]. UCVFDT has the ability to handle examples with uncertain attribute values by adopting the model described in [20] to represent uncertain nominal attribute values.

There is also research on clustering uncertain data streams with concept drift. Aggarwal and Yu proposed a framework for clustering uncertain data streams [1]. They designed uncertain micro-clusters to track the statistics of the stream, and leveraged them for the clustering process.

3 VFDT and CFDTu

In this section, we present a decision tree induction algorithm CFDTu, an acronym for Clustering Feature Decision Tree for uncertain data streams, which is an extension of VFDT. We will first introduce some definitions and an uncertain data model.

The classification problem for data streams is generally defined as follows. A set of infinite training examples $S = \{(\boldsymbol{x}_t, y_t)|t = 1, \ldots, N, \ldots\}$ of the form $(\boldsymbol{x}, y)$ is given, where $\boldsymbol{x}_t = (x_{t1}, \ldots, x_{tD})$ is a value vector of D-dimensional attributes, each of which may be numerical or nominal. The attributes vector is represented as $\boldsymbol{X} = (X_1, \ldots, X_D)$. y_t is a nominal class label, that is $y_t \in \{l_1, \ldots, l_K\}$. The stream data classification is to build a model $y = f(\boldsymbol{x})$ from the training examples that will predict the class labels of future examples with high accuracy.

3.1 Uncertain Data Models

When a numerical attribute value x_{td} is uncertain, it is referred to as an uncertain numerical attribute (UNA). In this paper, the value of UNA is treated as a continuous random variable and denoted by the standard error model introduced in [1], which assumes that the attribute values and the standard errors of the values are available.

The uncertainty of the standard error model is characterized by the attribute value x_{td} and the associated estimated error $e_{td}(x)$. The exact function $e_{td}(x)$ is usually unknown, yet the standard deviation of the error can be measured. Therefore, it is adopted to represent the uncertainty information, and is denoted by $\psi_{td}(x) = SD[e_{td}(x)]$. We also assume that the mean of the error is 0, that is $E[e_{td}(x)] = 0$. The corresponding D-dimensional error vector of the example $\boldsymbol{x}_t$ is denoted by $\psi_t(\boldsymbol{x}) = (\psi_{t1}(x), \ldots, \psi_{tD}(x))$. Therefore, the tth training example is denoted by the triple $(\boldsymbol{x}_t, \psi_t(\boldsymbol{x}), y_t)$, and the corresponding classification model is denoted by $y = f(\boldsymbol{x}, \psi(\boldsymbol{x}))$.

3.2 VFDT

VFDT is a classification model that constructs a decision tree from data streams incrementally. The only information needed in memory is the tree itself, which stores sufficient statistics at its leaves in order to grow. VFDT is constructed by making recursive splits of leaves and subsequently obtaining internal decision nodes, such that a tree structure is formed. The splits are decided by heuristic evaluation functions to choose the best cut-point of an attribute. The main innovation of VFDT is the use of Hoeffding bound to decide how many examples are necessary to be collected for heuristic evaluation at each leaf. Assuming a real-valued random variable r whose range is R_a, suppose there are N independent measurements of this variable whose mean value is $\overline{r}$. The Hoeffding bound indicates that, with probability $1 - \delta$, the true mean of the variable is at least $\overline{r} - \epsilon$, where

$$\epsilon = \sqrt{\frac{R_a^2 \ln(1/\delta)}{2N}}. \tag{1}$$

Let $H(\cdot)$ be a heuristic evaluation function, assume H is maximized. Let X_a be the attribute with highest observed $\overline{H}$ after observing N examples, and X_b be the second highest attribute. Let $\Delta\overline{H} = \overline{H}(X_a) - \overline{H}(X_b)$ be the difference between the two best observed heuristic values. Then, given a desired δ, the Hoeffding bound guarantees that X_a is the best attribute with probability $1 - \delta$ if N examples have been observed at this leaf and $\Delta\overline{H} > \epsilon$. At this point, the leaf can be split using X_a, and is converted into an internal decision node.

Each leaf only stores the sufficient statistics about attribute values and class distributions, instead of the whole example. When an example traverses to a leaf, the statistics are updated, and the heuristic function is evaluated to check if the leaf can be split. Assuming the heuristic function is the information gain, the sufficient statistics are the counts n_{idk}, representing the number of examples of class l_k that reach the leaf, where the attribute X_d takes the value i. VFDT uses

majority voting strategy in classification. To classify an unlabeled test example, it is filtered down to an appropriate leaf, where it is classified with the majority class of the training examples at the leaf.

3.3 CF Vector

CFDTu extends VFDT by using micro-clustering and clustering feature techniques. It has the ability of learning from uncertain data streams represented by the standard error model. Fig. 1 shows the structure of CFDTu, where each leaf contains a number of micro-clusters. Micro-clusters are used for storing the statistics and computing the heuristic evaluation function, as well as performing classification on test examples.

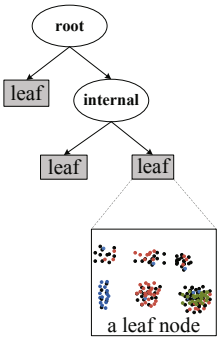


Fig. 1. Structure of CFDTu

The micro-clustering technique was originally exploited by Zhang et al. [26], and then was adopted in [1,14,25]. The concept of micro-cluster, also known as the Clustering Feature vector, denotes a statistically summarized representation of a group of examples which belong to the same cluster.

An uncertain Clustering Feature (CF) vector is a tuple that summarizes the sufficient information of a cluster. The characteristic of CF vector makes the clustering incremental without expensive computations. Given a cluster containing N uncertain examples $\{(\boldsymbol{x}_t, \psi_t(\boldsymbol{x}), y_t)|t = 1, \ldots, N\}$, its uncertain CF vector is a tuple $CF = (\boldsymbol{LS}, \boldsymbol{SS}, \boldsymbol{SSE}, \boldsymbol{NC}, N)$, where $\boldsymbol{LS} = \sum_{t=1}^{N} \boldsymbol{x}_t$ is the linear sum of the examples, and $\boldsymbol{SS} = \sum_{t=1}^{N} \boldsymbol{x}_t^2$ is the square sum of the examples, and $\boldsymbol{SSE} = \sum_{t=1}^{N} \psi_t^2(\boldsymbol{x})$ is the square sum of the standard errors of the examples, and the class distribution vector $\boldsymbol{NC} = (n_1, \ldots, n_K)$ is the number of examples from different class $l_1, \ldots, l_K$ respectively.

The additivity of the CF vector indicates that if $CF_1 = (\boldsymbol{LS}_1, \boldsymbol{SS}_1, \boldsymbol{SSE}_1, \boldsymbol{NC}_1, N_1)$ and $CF_2 = (\boldsymbol{LS}_2, \boldsymbol{SS}_2, \boldsymbol{SSE}_2, \boldsymbol{NC}_2, N_2)$ are the CF vectors of two disjoint clusters, the CF vector of the cluster that is formed by merging the two disjoint clusters is

$$CF_1{+}CF_2{=}(\boldsymbol{LS}_1{+}\boldsymbol{LS}_2, \boldsymbol{SS}_1{+}\boldsymbol{SS}_2, \boldsymbol{SSE}_1{+}\boldsymbol{SSE}_2, \boldsymbol{NC}_1{+}\boldsymbol{NC}_2, N_1{+}N_2). \quad (2)$$

3.4 CFDTu Overview

CFDTu inspects each example only once to obtain sufficient information. Learning from an uncertain example $(\boldsymbol{x}_t, \psi_t(\boldsymbol{x}), y_t)$ is achieved by adding it into a micro-cluster of a leaf. The process is as follows:

1. Identifying an appropriate leaf. The example traverses the CFDTu from the root to a leaf, testing the appropriate attribute at each internal decision node and following the branch corresponding to the attribute value of the example.
2. Seeking a micro-cluster. Choose the nearest micro-cluster at the leaf which is determined by the expected distance of the example to the micro-clusters. How to compute the expected distance will be introduced in section 3.5.
3. Modifying the leaf: If the micro-cluster can incorporate the example without violating the threshold R_{max} condition, add the example into it. If not, construct a new one for the example. Then discard the example to release memory. In this way, information driven from an example is learned.
4. Splitting the leaf: If a specified number of examples are collected, compute the heuristic evaluation function from the micro-clusters, and use the Hoeffding bound to decide when a particular attribute has won against all of the others. If so, the leaf is converted into internal decision node, and two new leaves are created. The process will be introduced in section 3.6.

CFDTu has two parameters at each leaf, the maximum radius of micro-clusters R_{max} and the maximum number of micro-clusters W. The radius of each micro-cluster has to be less than R_{max} and each leaf contains at most W micro-clusters $\{CF_1, \ldots, CF_W\}$.

The value of R_{max} will greatly affect the number of micro-clusters at a leaf. The initial value R^0_{max} can be set conservatively, which may produce more than W micro-clusters. R^0_{max} is determined at the very beginning of the model training and is achieved using the k-means clustering algorithm to create W clusters. Once these clusters have been established, the minimal value of the radii of all clusters can be assigned to R^0_{max}. When a new blank leaf is constructed, R_{max} should be adjusted to produce an appropriate number of micro-clusters. We use $R^{i+1}_{max} = R^i_{max}(W_i/W)^{1/D}$ where W_i is the number of micro-clusters at the parent node.

3.5 Uncertain Clustering

When a leaf is split and converted into an internal decision node, two blank children leaves are created. In each leaf, an uncertain clustering algorithm works in an incremental way to create W micro-clusters. The algorithm starts off with a number of null clusters and initially creates new singleton clusters, to which new examples are added subsequently. For any incoming uncertain examples, the nearest cluster is determined by using the distance of the example to the micro-clusters.

The distance metric function is the expected distance of the uncertain example $\boldsymbol{x}_t$ to the centroid $\boldsymbol{c}_i$ of an uncertain micro-cluster CF_i, which is denoted by

$ED(\boldsymbol{x}_t, \boldsymbol{c}_i)$. The centroid $\boldsymbol{c}_i$ is treated as a random variable vector and computed as the mean values and the mean error terms of the uncertain examples in the cluster, and $\hat{\boldsymbol{c}}_i$ is the instantiation of the centroid. Therefore, we have

$$\boldsymbol{c}_i = \frac{1}{N_i}(\boldsymbol{LS}_i + \boldsymbol{E}_i) \tag{3}$$

$$\hat{\boldsymbol{c}}_i = \boldsymbol{LS}_i/N_i, \tag{4}$$

where E_i is the sum of the errors of the examples, that is

$$\boldsymbol{E}_i = \sum_{\boldsymbol{x}_t \in CF_i} e_t(\boldsymbol{x}). \tag{5}$$

According to [1], given $E[e_t(\boldsymbol{x})] = 0$, we have

$$E(\|\boldsymbol{c}_i\|^2) = \frac{1}{N_i^2}\sum_{d=1}^{D}(LS_{id}^2 + SSE_{id}) \tag{6}$$

where $\|c_i\|^2$ is the square of Euclidean norm. $ED(\boldsymbol{x}_t, \boldsymbol{c}_i)$ is formulated as

$$ED(\boldsymbol{x}_t, \boldsymbol{c}_i) = E(\|\boldsymbol{x}_t - \boldsymbol{c}_i\|^2) = E(\|\boldsymbol{x}_t\|^2) - E(\|\boldsymbol{c}_i\|^2) + 2E(\boldsymbol{x}_t)\cdot E(\boldsymbol{c}_i), \tag{7}$$

where

$$E(\boldsymbol{x}_t)\cdot E(\boldsymbol{c}_i) = \frac{1}{N_i}\sum_{d=1}^{D}(x_{td}\cdot LS_{id}) \tag{8}$$

$$E(\|\boldsymbol{x}_t\|^2) = \|\boldsymbol{x}_t\|^2 + \|\psi_t(\boldsymbol{x})\|^2. \tag{9}$$

Therefore, by substitute the Eq. (6), (8) and (9) to Eq. (7), we have

$$ED(\boldsymbol{x}_t,\boldsymbol{c}_i)=\|\boldsymbol{x}_t\|^2+\|\psi_t(\boldsymbol{x})\|^2+\frac{1}{N_i^2}\sum_{d=1}^{D}(LS_{id}^2+SSE_{id})-\frac{2}{N_i}\sum_{d=1}^{D}(x_{td}\cdot LS_{id}). \tag{10}$$

The expected distances of an uncertain example $\boldsymbol{x}_t$ to all micro-clusters are computed, the nearest distance is chosen. If the nearest distance is less than the threshold R_{max}, $\boldsymbol{x}_t$ can be added to the micro-cluster. Otherwise, a new micro-cluster containing the singleton example needs to be created.

Adding an example $(\boldsymbol{x}_t, \psi_t(\boldsymbol{x}), y_t)$ to a micro-cluster is updating the CF vector of the cluster. That is

$$CF+(\boldsymbol{x}_t, \psi_t(\boldsymbol{x}), y_t) = (\boldsymbol{LS}+\boldsymbol{x}_t, \boldsymbol{SS}+\boldsymbol{x}_t^2, \boldsymbol{SSE}+\psi_t^2(\boldsymbol{x}), \boldsymbol{NC}+\boldsymbol{y}_t, N+1), \tag{11}$$

where $\boldsymbol{NC} + y_t$ means $n_k + 1$ if $y_t = l_k$.

Micro-clusters $CF_1, \ldots, CF_W$ are the sufficient statistics needed to compute heuristic evaluation function in section 3.6. After the example is added into a micro-cluster, examples in the micro-cluster can be substituted by the centroid.

3.6 Heuristic Evaluation Function

In CFDTu a leaf that contains a heuristic evaluation function for continuous attributes creates two descendant leaves. For an attribute X_d, the heuristic evaluation function is a condition of the form $X_d < cutp_d$ to determine if the attribute value x_{td} of an example $\boldsymbol{x}_t$ is less than $cutp_d$ or not. Then the cut-point $cutp_d$ divides the training dataset into two subsets, which correspond to the values TRUE (subset A) and FALSE (subset B) for the function. What needs to be determined is how to split subsets of $\boldsymbol{X}$ to produce the best tree-structured classifier. CFDTu uses recursive binary partition strategy.

At each stage in the recursive split, CFDTu uses an exhaustive method. All attributes $\boldsymbol{X}$ of the examples and all possible cut-point for each attribute are evaluated. To compute the class distribution of the examples for each possible cut-point, we compute the information of the two partitions using the following heuristic function [11]:

$$info(X_d^u) = P(X_d < cutp_{du}) \cdot Less(X_d^u) + P(X_d > cutp_{du}) \cdot Great(X_d^u), \quad (12)$$

where $cutp_{du}$ is the uth possible cut-point for attribute X_d, $info(X_d^u)$ is the information for X_d and $cutp_{du}$, and $Less(X_d^u)$ is the information of $X_d < cutp_{du}$, $Great(X_d^u)$ is the information of $X_d > cutp_{du}$. They are computed by:

$$Less(X_d^u) = -\sum_{k=1}^{K} P(y = l_k | X_d < cutp_{du}) \cdot \log[P(y = l_k | X_d < cutp_{du}], \quad (13)$$

$$Great(X_d^u) = -\sum_{k=1}^{K} P(y = l_k | X_d > cutp_{du}) \cdot \log[P(y = l_k | X_d > cutp_{du}]. \quad (14)$$

The fundamental variables for computing all these necessary statistics are the counts n_{idk}, which can be derived directly from the micro-clusters. Suppose a leaf contains W micro-clusters $\{CF_1, \ldots, CF_W\}$, and $\{\hat{\boldsymbol{c}}_1, \ldots, \hat{\boldsymbol{c}}_W\}$ are the current instantiations of the centroids. Since examples in a micro-cluster can be substituted by the centroid of the cluster, so the number of examples that have the value $\hat{\boldsymbol{c}}_i$ is N_i, the number of examples of class l_k that have the value $\hat{\boldsymbol{c}}_i$ is NC_{ik}. Moreover, as we have $\hat{\boldsymbol{c}}_i = (\hat{c}_{i1}, \ldots, \hat{c}_{id}, \ldots, \hat{c}_{iD})$ where $\hat{c}_{id}$ is the attribute-value of $\hat{\boldsymbol{c}}_i$ corresponding to attribute X_d, then the number of examples that have the value $\hat{c}_{id}$ is NC_{ik}, that is $n_{idk} = NC_{ik}$.

Letting $\hat{c}_{d(1)} < \ldots < \hat{c}_{d(W)}$ be the ordered distinct values of $\hat{c}_{d1} < \ldots < \hat{c}_{dW}$ corresponding to attribute X_d, then the uth possible cut-point can be

$$cutp_{du} = (\hat{c}_{d(u)} + \hat{c}_{d(u+1)})/2 \qquad (u = 1, \ldots, W - 1). \quad (15)$$

Therefore, even if there are a large number of examples, there are a finite number of possible cut-points to be considered.

Let $n_{d(1)}, \ldots, n_{d(W)}$ be the number of examples corresponding to $\hat{c}_{d(1)}, \ldots, \hat{c}_{d(W)}$, and N be the total number of examples at the leaf. Therefore,

$$P(X_d < cutp_{du}) = \sum_{w=1}^{u} n_{d(w)}/N, \quad (16)$$

$$P(y = l_k | X_d < cutp_{du}) = \sum_{\substack{w=1 \\ y=l_k}}^{u} n_{d(w)} / \sum_{\substack{w=1 \\ y=l_k}}^{W} n_{d(w)}. \quad (17)$$

$H(X_d^u)$ for attribute X_d at $cutp_{du}$ is computed by:

$$H(X_d^u) = Info(S) - Info(X_d^u). \quad (18)$$

Therefore, we can use the same method as in the induction of VFDT to choose the appropriate attribute and the corresponding cut-point.

3.7 Functional Tree Leaves

One of the innovations of our algorithm is that the current micro-clusters at the leaves also serve as classifiers. The majority class strategy of VFDT uses only the information about class distributions and does not consider the attribute-values. There is more information available during prediction at the leaves than using majority class classification. For example, Naive Bayes classification model can be adopted without extra statistics. This enhancement is referred to as the functional tree leaves [11].

For CFDTu, to predict the class label of a test example $(\boldsymbol{x}, \psi(\boldsymbol{x}))$, it traverses the tree from the root to a leaf. After the example falls into a leaf, the nearest neighbor algorithm is applied to find the nearest micro-cluster CF_i from all micro-clusters at the leaf and the corresponding majority class can be determined from $\boldsymbol{NC}_i$. This class label is the predicted value for $(\boldsymbol{x}, \psi(\boldsymbol{x}))$.

4 Experimental Results

We applied CFDTu on both synthetic datasets and real-world datasets to evaluate its performance in four aspects. These were accuracy in uncertain classification, memory cost, running time, and scalability. As currently there are no other methods can handle the standard error uncertainty, we compared CFDTu against the deterministic baseline method Hoeffding Tree Naive Bayes Adaptive (HT-NBA), and show how the uncertain data model and clustering feature techniques can be more effective in the case of uncertain data streams. HT-NBA works by performing a Naive Bayes prediction at the leaves of the VFDT. When performing a prediction on a test example, the leaf will only return a Naive Bayes prediction if it has better accuracies than the majority class, otherwise it resorts to a majority class prediction.

4.1 Datasets and Experiment Setup

There is a shortage of publicly available large real-world datasets that are suitable for the evaluation of data stream methods. Thus we used several synthetic datasets that can be found in the literature. They were Hyperplane, Random

Radial Basis Function (Random RBF), Random Tree, and Waveform. The detailed description can be found in [5]. Artificial concept drift was not generated, as both of the two methods are stationary stream data classifiers.

Two real-world datasets were Forest Covertype and KDD-99. We used all the 10 numeric attributes, 581,012 examples of Covertype dataset, and 34 numeric attributes, 1,000,000 examples of KDD-99 dataset in our experiments. We normalized all attribute values into values between $[0, 1]$. The basic properties of each dataset are listed in Table 1.

Table 1. Properties of the datasets

Name	Nominal	Numeric	Classes
Hyperplane	0	10	2
Random RBF	0	10	5
Random Tree	0	10	5
Wave21	0	21	3
Wave40	0	40	3
Covertype	44	10	7
KDD-99	0	34	23

We also introduced synthetic uncertainty information into all attributes of the datasets. The generative approach followed from [1]. An error is added to a deterministic attribute value x_{td} to generate uncertainty. Letting σ_d^0 be the standard deviation of the entire dataset along the dimension d, σ_d is defined as a uniform random variable drawn from the interval $[0, \eta \cdot \sigma_d^0]$, where η is the error level. Then, for the dimension d, error which is drawn from the Gaussian distribution with zero mean and standard deviation σ_d is added. $\eta 0$ denotes a deterministic dataset, and $\eta 0.5$ denotes that an error level is 0.5.

MOA (Massive Online Analysis) [4] is a software environment for stream data mining, including classification, clustering, and performance evaluations. The uncertain stream data classification algorithm CFDTu and HT-NBA were implemented in MOA. The experiments were performed on a 2.6 GHz Intel Dual-Core PC with 2 GB RAM, running Windows XP.

4.2 Classification Accuracy, Runtime, and Memory Usage

Experiments were conducted to evaluate CFDTu and HT-NBA using both synthetic datasets and real-world dataset with different error levels. The evaluation strategy was *interleaved test-then-train*: each example is used for testing the model before using it to train. Five synthetic datasets comprising up to 1,000,000 examples and 2 real-world datasets were used in the evaluation. Parameter settings are: $\tau = 0.05, n_{min} = 1000, \delta = 5 \times 10^{-6}$ for both CFDTu and HT-NBA, $W = 20$ for CFDTu in most cases, and $W = 30$ for dataset KDD-99.

Table 2 and Table 3 report the experiment results. Three dimensions of accuracy, memory usage and running time are compared. Accuracy is measured as the

final percentage of examples correctly classified over the test/train interleaved evaluation. Time is measured in seconds, and memory in MB.

Table 2. Experimental results using HT-NBA

	$\eta = 0$			$\eta = 0.1$			$\eta = 0.5$			$\eta = 1$		
Dataset	Acc.	Mem.	Time	Acc.	Mem.	Time	Acc.	Mem.	Time	Acc.	Mem.	Time
Hyperplane	89.93	0.90	15.60	89.33	0.92	17.55	81.01	0.93	18.97	72.17	0.96	17.32
Random RBF	94.52	0.41	16.49	94.44	0.41	18.67	89.01	0.45	20.48	83.83	0.42	22.15
Random Tree	94.28	0.36	14.51	89.07	0.35	17.05	77.19	0.39	19.55	67.47	0.35	19.55
Wave21	83.90	1.14	33.68	83.01	1.10	36.79	77.46	1.07	37.33	76.22	1.05	37.52
Wave40	83.68	2.02	65.93	83.57	2.01	70.34	77.16	1.90	70.95	75.16	2.04	74.12
Covertype	80.74	0.35	65.93	79.20	0.31	14.52	72.00	0.28	15.71	69.12	0.27	16.29
KDD-99	99.46	0.42	62.93	98.95	0.36	70.53	94.02	0.44	76.92	93.60	0.67	91.51
Average	89.50	0.80	31.81	88.22	0.78	35.06	81.12	0.78	37.13	76.80	0.82	39.78

Table 3. Experimental results using CFDTu

	$\eta = 0$			$\eta = 0.1$			$\eta = 0.5$			$\eta = 1$		
Dataset	Acc.	Mem.	Time	Acc.	Mem.	Time	Acc.	Mem.	Time	Acc.	Mem.	Time
Hyperplane	85.16	3.07	28.44	84.05	3.06	26.16	81.53	3.40	28.84	76.67	3.89	44.68
Random RBF	90.46	0.62	46.83	89.97	0.64	52.29	87.06	0.66	38.33	85.51	0.72	48.34
Random Tree	90.80	1.05	23.57	88.45	1.16	38.31	82.33	1.01	39.64	77.01	1.08	23.82
Wave21	81.65	3.55	74.23	80.98	3.65	61.07	78.60	3.68	65.60	77.37	3.73	57.31
Wave40	79.75	6.78	110.09	79.25	0.93	114.55	77.91	6.44	102.09	76.54	7.63	125.64
Covertype	78.05	0.81	62.58	77.16	1.13	36.41	75.24	0.89	40.83	72.48	0.98	43.52
KDD-99	98.06	1.09	170.57	97.55	2.54	174.19	96.71	1.34	195.58	96.06	1.26	223.74
Average	86.28	2.42	73.76	85.34	2.54	71.85	82.77	2.49	72.99	80.23	2.76	81.00

The results show that HT-NBA can obtain better accuracies when error level is low. However, its accuracy decreases significantly when error level increases. The uncertainty greatly affects the computation of the heuristic evaluation function, making HT-NBA choose improper split attributes and cut-points.

CFDTu has better accuracy than HT-NBA when error level increases. There are three reasons. First, by exploiting the uncertainty information, an example can be assigned to an appropriate cluster, making the boundary of examples from different classes more accurate. Second, by using micro-clustering approach, CFDTu can group examples not only by their class labels or by the values of a single attribute, but also by their overall similarity. Therefore, the count n_{idk} in CFDTu is more accurate and representative than that in HT-NBA. Then the heuristic evaluation function derived from n_{idk} can choose more appropriate attributes and cut-points. Third, the micro-cluster classifiers at the leaves can exploit the available information of examples.

The tables also show that HT-NBA is more efficient and space saving than CFDTu on all of the datasets.This is because that CFDTu needs to devote more time dealing with the uncertainty information. CFDTu needs 2-3 times more spaces than that of HT-NBA. Considering its ability for uncertain data classification, the efficiency and the space complexity is quite modest.

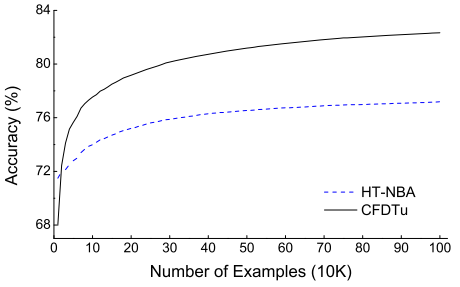


Fig. 2. Accuracy on Random Tree dataset

The learning curves for the Random Tree dataset where the error level is 0.5 are plotted in Fig. 2. The X-axis represents the number of examples that have been learned and tested, and the Y-axis represents the evaluation accuracy.

It can be learned from Fig. 2 that the accuracies of both methods are improved with more examples being learned. The accuracy of HT-NBA is higher than that of CFDTu at beginning, but it improves slowly. As the learning progresses, CFDTu finally surpasses HT-NBA.

4.3 Runtime and Scalability

Fig. 3 reports the scalability of CFDTu on high-dimensional and multi-class data. The size of the tree including internal decision nodes and leaves and runtime on Random RBF dataset comprising 1,000,000 examples are measured. The tree size for a different number of attributes (D) with a different number of classes (K) are plotted. We observe that higher values of K lead to a smaller size of tree. This could occur because when K is larger, more examples are needed to determine the optimal split attribute. Since more examples are used at one leaf split, fewer splits can be made with a fixed number of examples, leading to fewer internal nodes and leaves. Moreover, the size of the tree drops with the increasing number of attributes D for a particular value of K. It has the same reason. When the number of attributes increases, more examples are needed per leaf and fewer nodes can be generated.

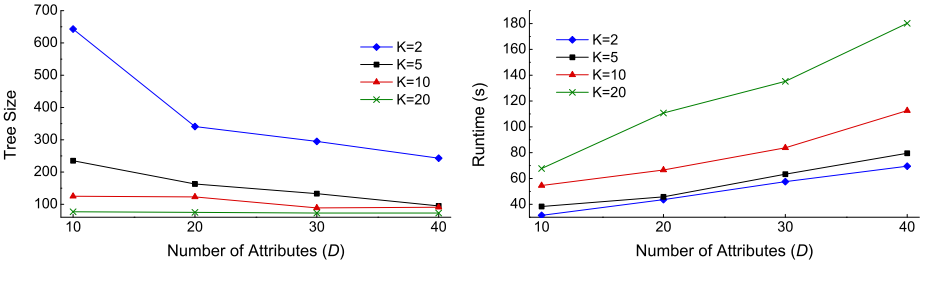


Fig. 3. Tree size and runtime curves on Random RBF dataset

The runtime curves with the number of attributes (D) and the number of classes (K) are also plotted. We observe that the runtime increases linearly with the number of attributes and the number of classes. This is because the runtime of constructing CF vectors, computing the heuristic evaluation function, and classifying unlabeled examples are all linear with the number of attributes and the number of classes. This is desirable. It can be concluded that CFDTu scales linearly to higher dimensionality and class labels.

5 Conclusions

In this paper, we address the issue of uncertain stream data classification. We build a decision tree model from data streams having uncertain attribute values which are represented by the standard error model. CFDTu applies clustering algorithm on uncertain data streams, and extracts sufficient statistics into micro-clusters. Micro-clusters are used for computing the heuristic evaluation function, as well as performing classification on test examples. Our experiments on synthetic and real-world datasets show that CFDTu is highly scalable for data streams and more effective than a purely deterministic method.

Mining from data streams having uncertain class labels is another important issue. Algorithms have been proposed to solve the problem. In future, we would like to develop an uncertain model to describe such scenarios, and incorporate the model into the CFDTu method, making CFDTu more generic.

References

1. Aggarwal, C.C., Han, J., Wang, J., Yu, P.: A Framework for Clustering Uncertain Data Streams. In: Proc. 24th Int. Conf. on Data Engineering, pp. 150–159 (2008)
2. Aggarwal, C.C.: Managing and Mining Uncertain Data. Springer Publishing Company, Incorporated (2009)
3. Bi, J., Zhang, T.: Support Vector Classification with Input Data Uncertainty. In: NIPS 2004: Advances in Neural Information Processing Systems, vol. 16, pp. 161–168 (2004)
4. Bifet, A., Kirkby, R., Holmes, G., Pfahringer, B.: MOA: Massive Online Analysis (2007), http://sourceforge.net/projects/moa-datastream
5. Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., Gavalda, R.: New ensemble methods for evolving data streams. In: Proc. 15th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, pp. 139–148 (2009)
6. Bifet, A., Holmes, G., Pfahringer, B., Frank, E.: Fast Perceptron Decision Tree Learning from Evolving Data Streams. In: Zaki, M.J., Yu, J.X., Ravindran, B., Pudi, V. (eds.) PAKDD 2010. LNCS, vol. 6119, pp. 299–310. Springer, Heidelberg (2010)
7. Bifet, A., Frank, E., Holmes, G., Pfahringer, B.: Accurate Ensembles for Data Streams: Combining Restricted Hoeffding Trees using Stacking. In: JMLR: Workshop and Conference Proceedings, pp. 225–240 (2010b)
8. Cheng, R., Kalashnikov, D.V., Prabhakar, S.: Evaluating Probabilistic Queries over Imprecise Data. In: Proc. 22nd ACM SIGMOD Int. Conf. on Management of Data, pp. 73–84 (2003)

9. Chapelle, O., Scholkopf, B., Zien, A.: Semi-Supervised Learning. MIT Press, Cambridge (2006)
10. Domingos, P., Hulten, G.: Mining high-speed data streams. In: Proc. 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, pp. 71–80 (2000)
11. Gama, J., Rocha, R., Medas, P.: Accurate decision trees for mining high-speed data streams. In: Proc. 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, pp. 523–528 (2003)
12. Ge, J., Xia, Y., Nadungodage, C.H.: A Neural Network for Uncertain Data Classification. In: Proc. 14th Pacific-Asia Conf. on Knowledge Discovery and Data Mining, pp. 449–460 (2010)
13. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: Proc. 7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, pp. 97–106 (2001)
14. Jin, W., Tung, A.K.H., Han, J.: Mining top-n local outliers in large databases. In: Proc. 7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, pp. 293–298 (2001)
15. Liang, C., Zhang, Y., Song, Q.: Decision Tree for Dynamic and Uncertain Data Streams. In: JMLR: Workshop and Conference Proceedings, vol. 13, pp. 209–224 (2010)
16. Masud, M.M., Gao, J., Khan, L., Han, J., Thuraisingham, B.: A practical approach to classify evolving data streams: training with limited amount of labeled data. In: Proc. 8th Int. Conf. on Data Mining, pp. 929–934 (2008)
17. Pan, S., Wu, K., Zhang, Y., Li, X.: Classifier Ensemble for Uncertain Data Stream Classification. In: Zaki, M.J., Yu, J.X., Ravindran, B., Pudi, V. (eds.) PAKDD 2010. LNCS, vol. 6118, pp. 488–495. Springer, Heidelberg (2010)
18. Pfahringer, B., Holmes, G., Kirkby, R.: New Options for Hoeffding Trees. In: Orgun, M.A., Thornton, J. (eds.) AI 2007. LNCS (LNAI), vol. 4830, pp. 90–99. Springer, Heidelberg (2007)
19. Qin, B., Xia, Y., Prabhakar, S., Tu, Y.: A Rule-Based Classification Algorithm for Uncertain Data. In: Proc. 25th IEEE Int. Conf. of Data Engineering, pp. 1633–1640 (2009)
20. Qin, B., Xia, Y., Li, F.: DTU: A Decision Tree for Uncertain Data. In: Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.) PAKDD 2009. LNCS, vol. 5476, pp. 4–15. Springer, Heidelberg (2009)
21. Scholz, M., Klinkenberg, R.: An ensemble classifier for drifting concepts. In: Proc. 2nd Int. Workshop on Knowledge Discovery in Data Streams, pp. 53–64 (2005)
22. Street, W.N., Kim, Y.: A Streaming Ensemble Algorithm (SEA) for large-scale classification. In: Proc. 7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, pp. 377–382 (2001)
23. Tsang, S., Kao, B., Yip, K.Y., Ho, W., Lee, S.D.: Decision Trees for Uncertain Data. In: Proc. 25th IEEE Int. Conf. of Data Engineering, pp. 441–444 (2009)
24. Wang, H., Fan, W., Yu, P., Han, J.: Mining concept-drifting data streams using ensemble classifiers. In: Proc. 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, pp. 226–235 (2003)
25. Yu, H., Yang, J., Han, J.: Classifying large data sets using SVMs with hierarchical clusters. In: Proc. 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, pp. 306–315 (2003)
26. Zhang, T., Ramakrishnan, R., Livny, M.: BIRCH: an efficient data clustering method for very large databases. In: Proc. ACM SIGMOD Int. Conf. on Management of Data, pp. 103–114 (1996)

Multi-view Laplacian Support Vector Machines

Shiliang Sun

Department of Computer Science and Technology,
East China Normal University, Shanghai 200241, China
slsun@cs.ecnu.edu.cn

Abstract. We propose a new approach, multi-view Laplacian support vector machines (SVMs), for semi-supervised learning under the multi-view scenario. It integrates manifold regularization and multi-view regularization into the usual formulation of SVMs and is a natural extension of SVMs from supervised learning to multi-view semi-supervised learning. The function optimization problem in a reproducing kernel Hilbert space is converted to an optimization in a finite-dimensional Euclidean space. After providing a theoretical bound for the generalization performance of the proposed method, we further give a formulation of the empirical Rademacher complexity which affects the bound significantly. From this bound and the empirical Rademacher complexity, we can gain insights into the roles played by different regularization terms to the generalization performance. Experimental results on synthetic and real-world data sets are presented, which validate the effectiveness of the proposed multi-view Laplacian SVMs approach.

Keywords: graph Laplacian, multi-view learning, reproducing kernel Hilbert space, semi-supervised learning, support vector machine.

1 Introduction

Semi-supervised learning or learning from labeled and unlabeled examples has attracted considerable attention in the last decade [1,2,3]. This is partially motivated by the fact that for many practical applications collecting a large number of unlabeled data is much less involved than collecting labeled data considering the expensive and tedious annotation process. Moreover, as human learning often occurs in the semi-supervised learning manner (for example, children may hear some words but do not know their exact meanings), research on semi-supervised learning also has the potential to uncover insights into mechanisms of human learning [4].

In some machine learning applications, examples can be described by different kinds of information. For example, in television broadcast understanding, broadcast segments can be simultaneously described by their video signals and audio signals which can be regarded as information from different properties or different "views". Multi-view semi-supervised learning, the focus of this paper, attempts to perform inductive learning under such circumstances. However, it

J. Tang et al. (Eds.): ADMA 2011, Part II, LNAI 7121, pp. 209–222, 2011.

should be noted that if there are no natural multiple views, artificially generated multiple views can still work favorably [5].

In this paper we are particularly interested in multi-view semi-supervised learning approaches derived from support vector machines (SVMs) [6]. As a state-of-the-art method in machine learning, SVMs not only are theoretically well justified but also show very good performance for real applications. The transductive SVMs [7], S^3VMs [8,9] and Laplacian SVMs [10] have been proposed as extensions of SVMs from supervised learning to single-view semi-supervised learning. For multi-view learning, there are also several extensions of SVMs proposed such as the co-Laplacian SVMs [11] and SVM-2K [12].

Regularization theory is an important technique in mathematics and machine learning [13,14]. Many methods can be explained from the point of view of regularization. A close parallel to regularization theory is capacity control of function classes [15]. Both regularization and capacity control of function classes can play a central role in alleviating over-fitting of machine learning algorithms.

The new method, multi-view Laplacian SVMs, proposed in this paper can also be explained by regularization theory and capacity control of function classes. It integrates three regularization terms respectively on function norm, manifold and multi-view regularization. As an appropriate integration of them and thus the effective use of information from labeled and unlabeled data, our method has the potential to outperform many related counterparts. The different roles of these regularization terms on capacity control will be unfolded later as a result of our empirical Rademacher complexity analysis. Besides giving the bound on the generalization error, we also report experimental results of the proposed method on synthetic and real-world data sets.

The layout of this paper is as follows. Section 2 introduces the objective function of the proposed approach with concerns on different regularization terms, and its optimization. Theoretical insights on the generalization error and the empirical Rademacher complexity are covered by Section 3. Then, experimental results are reported in Section 4. Finally, conclusions are drawn in Section 5.

2 Multi-view Laplacian SVMs (MvLapSVM)

2.1 Manifold Regularization

Let $x_1, \ldots, x_{l+u} \in \mathbf{R}^d$ denote a set of inputs including l labeled examples and u unlabeled ones with label space $\{+1, -1\}$. For manifold regularization, a data adjacency graph $W_{(l+u)\times(l+u)}$ is defined whose entries measure the similarity or closeness of every pair of inputs. We use a typical construction of W: $W_{ij} = 0$ for most pairs of inputs, and for neighboring x_i, x_j the corresponding entry is given by

$$W_{ij} = \exp(-\|x_i - x_j\|^2/2\sigma^2), \tag{1}$$

where $\|x_i - x_j\|$ is the Euclidean norm in $\mathbf{R}^d$.

The manifold regularization functional acting on any function $f : \mathbf{R}^d \to \mathbf{R}$ is defined as follows [16]

$$M_{reg}(f) = \frac{1}{2} \sum_{i,j=1}^{l+u} W_{ij}(f(x_i) - f(x_j))^2. \tag{2}$$

It is clear that a smaller $M_{reg}(f)$ indicates a smoother function f. Define vector $\mathbf{f} = (f(x_1), \ldots, f(x_{l+u}))^\top$. Then

$$\begin{aligned} M_{reg}(f) &= \sum_{i=1}^{l+u}(\sum_{j=1}^{l+u} W_{ij}) f^2(x_i) - \sum_{i,j=1}^{l+u} W_{ij} f(x_i) f(x_j) \\ &= \mathbf{f}^\top (V - W)\mathbf{f}, \end{aligned} \tag{3}$$

where matrix V is diagonal with the ith diagonal entry $V_{ii} = \sum_{j=1}^{l+u} W_{ij}$. The matrix $L \triangleq V - W$, which is arguably positive semidefinite, is called the graph Laplacian of W. In our empirical studies in Section 4, a normalized Laplacian $\bar{L} = V^{-1/2} L V^{-1/2}$ is used because this normalized one often performs as well or better in practical tasks [10].

2.2 Multi-view Regularization

For multi-view learning, an input $x \in \mathbf{R}^d$ can be decomposed into components corresponding to multiple views, such as $x = (x^1, \ldots, x^m)$ for an m-view representation. A function f_j defined on view j only depends on x^j, while ignoring the other components $(x^1, \ldots, x^{j-1}, x^{j+1}, \ldots, x^m)$.

For multi-view semi-supervised learning, there is a commonly acceptable assumption that a good learner can be learned from each view [17]. Consequently, these good learners in different views should be consistent to a large extent with respect to their predictions on the same examples. We also adopt this assumption and use the regularization idea to wipe off those inconsistent learners. Given the $l+u$ examples, the multi-view regularization functional for m functions $f_1, \ldots, f_m$ can be formulated as

$$V_{reg}(f_1, \ldots, f_m) = \sum_{j>k,k=1}^{m} \sum_{i=1}^{l+u} [f_j(x_i) - f_k(x_i)]^2. \tag{4}$$

Clearly, a smaller $V_{reg}(f_1, \ldots, f_m)$ tends to find good learners in each view.

2.3 MvLapSVM

As is usually assumed in multi-view learning, each view is regarded to be sufficient to train a good learner. Therefore, we can write the final prediction as $f = \frac{1}{m} \sum_{i=1}^{m} f_i$. For MvLapSVM, in this paper we concentrate on the two-view

case, that is $m = 2$. In this scenario, the objective function for MvLapSVM is defined as

$$\begin{aligned}\min_{f_1\in\mathcal{H}_1, f_2\in\mathcal{H}_2} \ & \frac{1}{2l}\sum_{i=1}^{l}[(1-y_i f_1(x_i))_+ + (1-y_i f_2(x_i))_+] + \\ & \gamma_1(\|f_1\|^2 + \|f_2\|^2) + \frac{\gamma_2}{(l+u)^2}(\mathbf{f_1}^\top L_1 \mathbf{f_1} + \\ & \mathbf{f_2}^\top L_2 \mathbf{f_2}) + \frac{\gamma_3}{(l+u)}\sum_{i=1}^{l+u}[f_1(x_i) - f_2(x_i)]^2, \end{aligned} \tag{5}$$

where $\mathcal{H}_1, \mathcal{H}_2$ are the reproducing kernel Hilbert spaces [18,19] in which f_1, f_2 are defined, nonnegative scalars $\gamma_1, \gamma_2, \gamma_3$ are respectively norm regularization, manifold regularization and multi-view regularization coefficients, and vector $\mathbf{f_1} = (f_1(x_1), ..., f_1(x_{l+u}))^\top$, $\mathbf{f_2} = (f_2(x_1), ..., f_2(x_{l+u}))^\top$.

2.4 Optimization

We now concentrate on solving (5). As an application of the representer theorem [20,21], the solution to problem (5) has the following form

$$f_1(x) = \sum_{i=1}^{l+u}\alpha_1^i k_1(x_i, x),\ f_2(x) = \sum_{i=1}^{l+u}\alpha_2^i k_2(x_i, x). \tag{6}$$

Therefore, we can rewrite $\|f_1\|^2$ and $\|f_2\|^2$ as

$$\|f_1\|^2 = \boldsymbol{\alpha}_1^\top K_1 \boldsymbol{\alpha}_1,\ \|f_2\|^2 = \boldsymbol{\alpha}_2^\top K_2 \boldsymbol{\alpha}_2, \tag{7}$$

where K_1 and K_2 are $(l+u)\times(l+u)$ Gram matrices respective from view $\mathcal{V}^1$ and $\mathcal{V}^2$, and vector $\boldsymbol{\alpha}_1 = (\alpha_1^1, ..., \alpha_1^{l+u})^\top$, $\boldsymbol{\alpha}_2 = (\alpha_2^1, ..., \alpha_2^{l+u})^\top$. In addition, we have

$$\mathbf{f_1} = K_1\boldsymbol{\alpha}_1,\ \mathbf{f_2} = K_2\boldsymbol{\alpha}_2. \tag{8}$$

To simplify our formulations, we respectively replace $\frac{\gamma_2}{(l+u)^2}$ and $\frac{\gamma_3}{(l+u)}$ in (5) with γ_2 and γ_3. Thus, the primal problem can be reformulated as

$$\begin{aligned}\min_{\boldsymbol{\alpha}_1,\boldsymbol{\alpha}_2,\boldsymbol{\xi}_1,\boldsymbol{\xi}_2} \ & F_0 = \frac{1}{2l}\sum_{i=1}^{l}(\xi_1^i + \xi_2^i) + \gamma_1(\boldsymbol{\alpha}_1^\top K_1\boldsymbol{\alpha}_1 + \\ & \boldsymbol{\alpha}_2^\top K_2\boldsymbol{\alpha}_2) + \gamma_2(\boldsymbol{\alpha}_1^\top K_1 L_1 K_1\boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_2^\top K_2 L_2 K_2\boldsymbol{\alpha}_2) + \\ & \gamma_3(K_1\boldsymbol{\alpha}_1 - K_2\boldsymbol{\alpha}_2)^\top(K_1\boldsymbol{\alpha}_1 - K_2\boldsymbol{\alpha}_2) \\ \text{s.t.} \ & \begin{cases} y_i(\sum_{j=1}^{l+u}\alpha_1^j k_1(x_j, x_i)) \geq 1 - \xi_1^i, \\ y_i(\sum_{j=1}^{l+u}\alpha_2^j k_2(x_j, x_i)) \geq 1 - \xi_2^i, \\ \xi_1^i,\ \ \xi_2^i \geq 0, \quad i = 1, \ldots, l\ , \end{cases} \end{aligned} \tag{9}$$

where $y_i \in \{+1, -1\}$, $\gamma_1, \gamma_2, \gamma_3 \geq 0$. Note that the additional bias terms are embedded in the weight vectors of the classifiers by using the example representation of augmented vectors.

We present two theorems concerning the convexity and strong duality (which means the optimal value of a primal problem is equal to that of its Lagrange dual problem [22]) of problem (9) with proofs omitted.

Theorem 1. *Problem (9) is a convex optimization problem.*

Theorem 2. *Strong duality holds for problem (9).*

Suppose $\lambda_1^i, \lambda_2^i \geq 0$ are the Lagrange multipliers associated with the first two sets of inequality constraints of problem (9). Define $\boldsymbol{\lambda}_1 = (\lambda_1^1, ..., \lambda_1^l)^\top$ and $\boldsymbol{\lambda}_2 = (\lambda_2^1, ..., \lambda_2^l)^\top$. It can be shown that the Lagrangian dual optimization problem with respect to $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ is a quadratic program. Classifier parameters $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2$ used by (6) can be solved readily after we get $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$.

3 Theoretical Analysis

In this section, we give a theoretical analysis of the generalization error of the MvLapSVM method in terms of the theory of Rademacher complexity bounds.

3.1 Background Theory

Some important background on Rademacher complexity theory is introduced as follows.

Definition 1 (Rademacher complexity, [15,23,24]). *For a sample $S = \{x_1, \ldots, x_l\}$ generated by a distribution $\mathcal{D}_x$ on a set X and a real-valued function class $\mathcal{F}$ with domain X, the empirical Rademacher complexity of $\mathcal{F}$ is the random variable*

$$\hat{R}_l(\mathcal{F}) = \mathbb{E}_{\boldsymbol{\sigma}}[\sup_{f \in \mathcal{F}} |\frac{2}{l} \sum_{i=1}^{l} \sigma_i f(x_i)| | x_1, \ldots, x_l],$$

where $\boldsymbol{\sigma} = \{\sigma_1, \ldots, \sigma_l\}$ are independent uniform $\{\pm 1\}$-valued (Rademacher) random variables. The Rademacher complexity of $\mathcal{F}$ is

$$R_l(\mathcal{F}) = \mathbb{E}_S[\hat{R}_l(\mathcal{F})] = \mathbb{E}_{S\boldsymbol{\sigma}}[\sup_{f \in \mathcal{F}} |\frac{2}{l} \sum_{i=1}^{l} \sigma_i f(x_i)|].$$

Lemma 1 ([15]). *Fix $\delta \in (0, 1)$ and let $\mathcal{F}$ be a class of functions mapping from an input space Z (for supervised learning having the form $Z = X \times Y$) to $[0, 1]$. Let $(z_i)_{i=1}^l$ be drawn independently according to a probability distribution*

$\mathcal{D}$. Then with probability at least $1-\delta$ over random draws of samples of size l, every $f \in \mathcal{F}$ satisfies

$$\mathbb{E}_{\mathcal{D}}[f(z)] \leq \hat{\mathbb{E}}[f(z)] + R_l(\mathcal{F}) + \sqrt{\frac{\ln(2/\delta)}{2l}}$$
$$\leq \hat{\mathbb{E}}[f(z)] + \hat{R}_l(\mathcal{F}) + 3\sqrt{\frac{\ln(2/\delta)}{2l}},$$

where $\hat{\mathbb{E}}[f(z)]$ is the empirical error averaged on the l examples.

Note that the above lemma is also applicable if we replace $[0,1]$ by $[-1,0]$. This can be justified by simply following the proof of Lemma 1, as detailed in [15].

3.2 The Generalization Error of MvLapSVM

We obtain the following theorem regarding the generalization error of MvLapSVM, which is similar to one theorem in [12]. The prediction function in MvLapSVM is adopted as the average of prediction functions from two views

$$g = \frac{1}{2}(f_1 + f_2). \tag{10}$$

Theorem 3. *Fix $\delta \in (0,1)$ and let $\mathcal{F}$ be the class of functions mapping from $Z = X \times Y$ to $\boldsymbol{R}$ given by $\tilde{f}(x,y) = -yg(x)$ where $g = \frac{1}{2}(f_1+f_2) \in \mathcal{G}$ and $\tilde{f} \in \mathcal{F}$. Let $S = \{(x_1,y_1),\cdots,(x_l,y_l)\}$ be drawn independently according to a probability distribution $\mathcal{D}$. Then with probability at least $1-\delta$ over samples of size l, every $g \in \mathcal{G}$ satisfies*

$$P_{\mathcal{D}}(y \neq sgn(g(\boldsymbol{x}))) \leq \frac{1}{2l}\sum_{i=1}^{l}(\xi_1^i + \xi_2^i) + 2\hat{R}_l(\mathcal{G}) + 3\sqrt{\frac{\ln(2/\delta)}{2l}},$$

where $\xi_1^i = (1 - y_i f_1(x_i))_+$ and $\xi_2^i = (1 - y_i f_2(x_i))_+$.

Proof. Let $H(\cdot)$ be the Heaviside function that returns 1 if its argument is greater than 0 and zero otherwise. Then it is clear to have

$$P_{\mathcal{D}}(y \neq sgn(g(\mathbf{x}))) = \mathbb{E}_{\mathcal{D}}[H(-yg(\mathbf{x}))]. \tag{11}$$

Consider a loss function $\mathcal{A} : \mathbf{R} \to [0,1]$, given by

$$\mathcal{A}(a) = \begin{cases} 1, & \text{if } a \geq 0; \\ 1+a, & \text{if } -1 \leq a \leq 0; \\ 0, & \text{otherwise.} \end{cases}$$

By Lemma 1 and since function $\mathcal{A} - 1$ dominates $H - 1$, we have [15]

$$\mathbb{E}_{\mathcal{D}}[H(\tilde{f}(x,y)) - 1] \leq \mathbb{E}_{\mathcal{D}}[\mathcal{A}(\tilde{f}(x,y)) - 1]$$
$$\leq \hat{\mathbb{E}}[\mathcal{A}(\tilde{f}(x,y)) - 1] + \hat{R}_l((\mathcal{A}-1)\circ\mathcal{F}) + 3\sqrt{\frac{\ln(2/\delta)}{2l}}.$$

Therefore,

$$\begin{aligned} &\mathbb{E}_{\mathcal{D}}[H(\tilde{f}(x,y))] \\ \leq\ &\hat{\mathbb{E}}[\mathcal{A}(\tilde{f}(x,y))] + \hat{R}_l((\mathcal{A}-1)\circ\mathcal{F}) + 3\sqrt{\frac{\ln(2/\delta)}{2l}}. \end{aligned} \tag{12}$$

In addition, we have

$$\begin{aligned} \hat{E}[\mathcal{A}(\tilde{f}(x,y))] &\leq \frac{1}{l}\sum_{i=1}^{l}(1-y_i g(x_i))_+ \\ &= \frac{1}{2l}\sum_{i=1}^{l}(1-y_i f_1(x_i)+1-y_i f_2(x_i))_+ \\ &\leq \frac{1}{2l}\sum_{i=1}^{l}[(1-y_i f_1(x_i))_+ + (1-y_i f_2(x_i))_+] \\ &= \frac{1}{2l}\sum_{i=1}^{l}(\xi_1^i+\xi_2^i), \end{aligned} \tag{13}$$

where ξ_1^i denotes the amount by which function f_1 fails to achieve margin 1 for (x_i, y_i) and ξ_2^i applies similarly to function f_2.

Since $(\mathcal{A}-1)(0)=0$, we can apply the Lipschitz condition [23] of function $(\mathcal{A}-1)$ to get

$$\hat{R}_l((\mathcal{A}-1)\circ\mathcal{F}) \leq 2\hat{R}_l(\mathcal{F}). \tag{14}$$

It remains to bound the empirical Rademacher complexity of the class $\mathcal{F}$. With $y_i \in \{+1,-1\}$, we have

$$\begin{aligned} \hat{R}_l(\mathcal{F}) &= \mathbb{E}_{\boldsymbol{\sigma}}[\sup_{f\in\mathcal{F}}|\frac{2}{l}\sum_{i=1}^{l}\sigma_i\tilde{f}(x_i,y_i)|] \\ &= \mathbb{E}_{\boldsymbol{\sigma}}[\sup_{g\in\mathcal{G}}|\frac{2}{l}\sum_{i=1}^{l}\sigma_i y_i g(x_i)|] \\ &= \mathbb{E}_{\boldsymbol{\sigma}}[\sup_{g\in\mathcal{G}}|\frac{2}{l}\sum_{i=1}^{l}\sigma_i g(x_i)|] = \hat{R}_l(\mathcal{G}). \end{aligned} \tag{15}$$

Now combining (11)∼(15) reaches the conclusion of this theorem. □

3.3 The Empirical Rademacher Complexity $\hat{R}_l(\mathcal{G})$

In this section, we give the expression of $\hat{R}_l(\mathcal{G})$ used in Theorem 3. $\hat{R}_l(\mathcal{G})$ is also important in identifying the different roles of regularization terms in the MvLapSVM approach. The techniques adopted to derive $\hat{R}_l(\mathcal{G})$ is analogical to and inspired by those used for analyzing co-RLS in [19,24].

The loss function $\hat{L} : \mathcal{H}^1 \times \mathcal{H}^2 \rightarrow [0, \infty)$ in (5) with $\hat{L} = \frac{1}{2l}\sum_{i=1}^{l}[(1 - y_i f_1(x_i))_+ + (1 - y_i f_2(x_i))_+]$ satisfies

$$\hat{L}(0, 0) = 1. \tag{16}$$

We now derive the regularized function class $\mathcal{G}$ from which our predictor g is drawn.

Let $Q(f_1, f_2)$ denote the objective function in (5). Substituting the predictors $f_1 \equiv 0$ and $f_2 \equiv 0$ into $Q(f_1, f_2)$ results in an upper bound

$$\min_{f_1, f_2 \in \mathcal{H}^1 \times \mathcal{H}^2} Q(f_1, f_2) \leq Q(0, 0) = \hat{L}(0, 0) = 1. \tag{17}$$

Because each term in $Q(f_1, f_2)$ is nonnegative, the optimal function pair (f_1^*, f_2^*) minimizing $Q(f_1, f_2)$ must be contained in

$$\begin{aligned}\mathcal{H} = \{(f_1, f_2) : \gamma_1(\|f_1\|^2 + \|f_2\|^2) + \gamma_2(\mathbf{f}_{1u}{}^\top L_{1u}\mathbf{f}_{1u} + \\ \mathbf{f}_{2u}{}^\top L_{2u}\mathbf{f}_{2u}) + \gamma_3 \sum_{i=l+1}^{l+u}[f_1(x_i) - f_2(x_i)]^2 \leq 1\},\end{aligned} \tag{18}$$

where parameters γ_1, γ_2, γ_3 are from (9), $\mathbf{f}_{1u} = (f_1(x_{l+1}), ..., f_1(x_{l+u}))^\top$, $\mathbf{f}_{2u} = (f_2(x_{l+1}), ..., f_2(x_{l+u}))^\top$, and L_{1u} and L_{2u} are the unnormalized graph Laplacians for the graphs only involving the unlabeled examples (to make theoretical analysis on $\hat{R}_l(\mathcal{G})$ feasible, we temporarily assume that the Laplacians in (5) are unnormalized).

The final predictor is found out from the function class

$$\mathcal{G} = \{x \rightarrow \frac{1}{2}[f_1(x) + f_2(x)] : (f_1, f_2) \in \mathcal{H}\}, \tag{19}$$

which does not depend on the labeled examples.

The complexity $\hat{R}_l(\mathcal{G})$ is

$$\hat{R}_l(\mathcal{G}) = \mathbb{E}_{\boldsymbol{\sigma}}[\sup_{(f_1, f_2) \in \mathcal{H}} |\frac{1}{l}\sum_{i=1}^{l}\sigma_i(f_1(x_i) + f_2(x_i))|]. \tag{20}$$

To derive the Rademacher complexity, we first convert from a supremum over the functions to a supremum over their corresponding expansion coefficients. Then, the Kahane-Khintchine inequality [25] is employed to bound the expectation over $\boldsymbol{\sigma}$ above and below, and give a computable quantity. The following theorem summarizes our derived Rademacher complexity.

Theorem 4. *Suppose* $\mathcal{S} = K_{1l}(\gamma_1 K_1 + \gamma_2 K_{1u}^\top L_{1u} K_{1u})^{-1} K_{1l}^\top + K_{2l}(\gamma_1 K_2 + \gamma_2 K_{2u}^\top L_{2u} K_{2u})^{-1} K_{2l}^\top$, $\Theta = K_{1u}(\gamma_1 K_1 + \gamma_2 K_{1u}^\top L_{1u} K_{1u})^{-1} K_{1u}^\top + K_{2u}(\gamma_1 K_2 + \gamma_2 K_{2u}^\top L_{2u} K_{2u})^{-1} K_{2u}^\top$, $\mathcal{J} = K_{1u}(\gamma_1 K_1 + \gamma_2 K_{1u}^\top L_{1u} K_{1u})^{-1} K_{1l}^\top - K_{2u}(\gamma_1 K_2 + \gamma_2 K_{2u}^\top L_{2u} K_{2u})^{-1} K_{2l}^\top$, *where* K_{1l} *and* K_{2l} *are respectively the first* l *rows of* K_1 *and* K_2, *and* K_{1u} *and* K_{2u} *are respectively the last* u *rows of* K_1 *and* K_2. *Then we have* $\frac{U}{\sqrt{2}l} \leq \hat{R}_l(\mathcal{G}) \leq \frac{U}{l}$ *with* $U^2 = tr(\mathcal{S}) - \gamma_3 tr(\mathcal{J}^\top(I + \gamma_3\Theta)^{-1}\mathcal{J})$.

4 Experiments

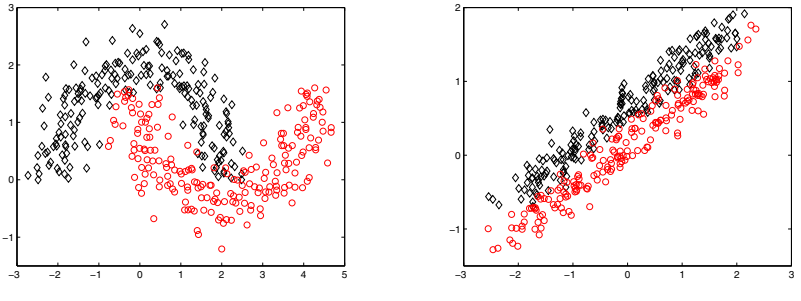
We performed multi-view semi-supervised learning experiments on a synthetic and two real-world classification problems. The Laplacian SVM (LapSVM) [10], co-Laplacian SVM (CoLapSVM) [11], manifold co-regularization (CoMR) [19] and co-SVM (a counterpart of the co-RLS in [11]) are employed for comparisons with our proposed method. For each method, besides considering the prediction function $(f_1 + f_2)/2$ for the combined view, we also consider the prediction functions f_1 and f_2 from the separate views.

Each data set is divided into a training set (including labeled and unlabeled training data), a validation set and a test set. The validation set is used to select regularization parameters from the range $\{10^{-10}, 10^{-6}, 10^{-4}, 10^{-2}, 1, 10, 100\}$, and choose which prediction function should be used. With the identified regularization parameter and prediction function, performances on the test data and unlabeled training data would be evaluated. The above process is repeated at random for ten times, and the reported performance is the averaged accuracy and the corresponding standard deviation.

4.1 Two-Moons-Two-Lines Synthetic Data

This synthetic data set is generated similarly to the toy example used in [11]. Noisy examples in two classes appear as two moons in one view and two parallel lines in the other view, and points on one moon are enforced at random to associate with points on one line (see Fig. 1 for an illustration). The sizes for labeled training set, unlabeled training set, validation set and test set are 10, 200, 100 and 100, respectively.

As in [11], a Gaussian and linear kernel are respectively chosen for the two-moons and two-lines view. The classification accuracies of different methods on this data set are shown in Table 1, where $\mathbb{T}$ and $\mathbb{U}$ means accuracies on the test data and unlabeled training data, respectively, and best accuracies are indicated in bold (if two methods bear the same accuracy, the smaller standard deviation will identify the better method).



(a) Two-moons view (b) Two-lines view

Fig. 1. Distribution of the two-moons-two-lines data

Table 1. Classification accuracies and standard deviations (%) of different methods on the synthetic data

	LapSVM	CoLapSVM	CoMR	Co-SVM	MvLapSVM
$\mathbb{T}$	91.40 (1.56)	93.40 (3.07)	91.20 (1.60)	96.30 (1.95)	**96.90** (1.70)
$\mathbb{U}$	90.60 (2.33)	93.55 (2.72)	90.90 (2.02)	96.40 (1.61)	**96.40** (1.46)

From this table, we see that methods solely integrating manifold or multi-view regularization give good performance, which indicates the usefulness of these regularization concerns. Moreover, among all the methods, the proposed MvLapSVM performs best both on the test set and unlabeled training set.

4.2 Image-Text Classification

We collected this data set from the sports gallery of the yahoo! website in 2008. It includes 420 NBA images and 420 NASCAR images, some of which are shown in Fig. 2. For each image, there is an attached short text describing content-related information. Therefore, image and text constitute the two views of this data set.

Each image is normalized to be a 32×32-sized gray image. Feature extraction for the texts is done by removing stop words, punctuation and numbers and then applying Porter's stemming [26]. In addition, words that occur in five or fewer documents were ignored. After this preprocessing, each text has a TFIDF feature [27] of 296 dimensions.

The sizes for labeled training set, unlabeled training set, validation set and test set are 10, 414, 206 and 210, respectively. Linear kernels are used for both views. The performance is reported in Table 2 where co-SVM ranks first on the test set while MvLapSVM outperforms all the other methods on the unlabeled

Fig. 2. NBA (left) and NASCAR (right) images

Table 2. Classification accuracies and standard deviations (%) of different methods on the NBA-NASCAR data

	LapSVM	CoLapSVM	CoMR	Co-SVM	MvLapSVM
$\mathbb{T}$	99.33 (0.68)	98.86 (1.32)	99.38 (0.68)	**99.43** (0.59)	99.38 (0.64)
$\mathbb{U}$	99.03 (0.88)	98.55 (0.67)	98.99 (0.90)	98.91 (0.38)	**99.54** (0.56)

training set. If we take the average of the accuracies on the test set and unlabeled training set, clearly our MvLapSVM ranks first.

4.3 Web Page Categorization

In this subsection, we consider the problem of classifying web pages. The data set consists of 1051 two-view web pages collected from the computer science department web sites at four U.S. universities: Cornell, University of Washington, University of Wisconsin, and University of Texas [17]. The task is to predict whether a web page is a course home page or not. Within the data set there are a total of 230 course home pages. The first view of the data is the words appearing on the web page itself, whereas the second view is the underlined words in all links pointing to the web page from other pages. We preprocess each view according to the feature extraction procedure used in Section 4.2. This results in 2332 and 87-dimensional vectors in view 1 and view 2 respectively [28]. Finally, document vectors were normalized to TFIDF features.

Table 3. Classification accuracies and standard deviations (%) of different methods on the web page data

	LapSVM	CoLapSVM	CoMR	Co-SVM	MvLapSVM
$\mathbb{T}$	94.02 (2.66)	93.68 (2.98)	94.02 (2.24)	93.45 (3.21)	**94.25** (1.62)
$\mathbb{U}$	93.33 (2.40)	93.39 (2.44)	93.26 (2.19)	93.16 (2.68)	**93.53** (2.04)

The sizes for labeled training set, unlabeled training set, validation set and test set are 12, 519, 259 and 261, respectively. Linear kernels are used for both views. Table 3 gives the classification results obtained by different methods. MvLapSVM outperforms all the other methods on both the test data and unlabeled training data.

5 Conclusion

In this paper, we have proposed a new approach for multi-view semi-supervised learning. This approach is an extension of SVMs for multi-view semi-supervised

learning with manifold and multi-view regularization integrated. We have proved the convexity and strong duality of the primal optimization problem, and used the dual optimization to solve classifier parameters. Moreover, theoretical results on the generalization performance of the MvLapSVM approach and the empirical Rademacher complexity which can indicate different roles of regularization terms have been made. Experimental practice on multiple data sets has also manifested the effectiveness of the proposed method.

The MvLapSVM is not a special case of the framework that Rosenberg et al. formulated in [29]. The main difference is that they require the loss functional depends only on the combined prediction function, while we use here a slightly general loss which has a separate dependence on the prediction function from each view. Their framework does not subsume our approach.

For future work, we mention the following three directions.

- **Model selection:** As is common in many machine learning algorithms, our method has several regularization parameters to set. Usually, a held out validation set would be used to perform parameter selection, as what was done in this paper. However, for the currently considered semi-supervised learning, this is not very natural because there is often a small quantity of labeled examples available. Model selection for semi-supervised learning using no or few labeled examples is worth further studying.
- **Multi-class classification:** The MvLapSVM algorithm implemented in this paper is intended for binary classification. Though the usual one-versus-rest, one-versus-another strategy, which converts a problem from multi-class to binary classification, can be adopted for multi-class classification, it is not optimal. Incorporating existing ideas of multi-class SVMs [30] into the MvLapSVM approach would be a further concern.
- **Regularization selection:** In this paper, although the MvLapSVM algorithm obtained good results, it involves more regularization terms than related methods and thus needs more assumptions. For some applications, these assumptions might not hold. Therefore, a probably interesting improvement could be comparing different kinds of regularizations and attempting to select those promising ones for each application. This also makes it possible to weight different views unequally.

Acknowledgments. This work was supported in part by the National Natural Science Foundation of China under Project 61075005, and the Fundamental Research Funds for the Central Universities.

References

1. Chapelle, O., Schölkopf, B., Zien, A.: Semi-supervised Learning. MIT Press, Cambridge (2006)
2. Culp, M., Michailidis, G.: Graph-based Semi-supervised Learning. IEEE Transactions on Pattern Analysis and Machine Intelligence 30, 174–179 (2008)
3. Sun, S., Shawe-Taylor, J.: Sparse Semi-supervised Learning using Conjugate Functions. Journal of Machine Learning Research 11, 2423–2455 (2010)

4. Zhu, X.: Semi-supervised Learning Literature Survey. Technical Report, 1530, University of Wisconsin-Madison (2008)
5. Sun, S., Jin, F., Tu, W.: View Construction for Multi-view Semi-supervised Learning. In: Liu, D., Zhang, H., Polycarpou, M., Alippi, C., He, H. (eds.) ISNN 2011, Part I. LNCS, vol. 6675, pp. 595–601. Springer, Heidelberg (2011)
6. Vapnik, V.N.: Statistical Learning Theory. Wiley, New York (1998)
7. Joachims, T.: Transductive Inference for Text Classification using Support Vector Machines. In: Proceedings of the 16th International Conference on Machine Learning, pp. 200–209 (1999)
8. Bennett, K., Demiriz, A.: Semi-supervised Support Vector Machines. Advances in Neural Information Processing Systems 11, 368–374 (1999)
9. Fung, G., Mangasarian, O.L.: Semi-supervised Support Vector Machines for Unlabeled Data Classification. Optimization Methods and Software 15, 29–44 (2001)
10. Belkin, M., Niyogi, P., Sindhwani, V.: Manifold Regularization: A Geometric Framework for Learning from Labeled and Unlabeled Exampls. Journal of Machine Learning Research 7, 2399–2434 (2006)
11. Sindhwani, V., Niyogi, P., Belkin, M.: A Co-regularization Approach to Semi-supervised Learning with Multiple Views. In: Proceedings of the Workshop on Learning with Multiple Views, International Conference on Machine Learning (2005)
12. Farquhar, J., Hardoon, D., Meng, H., Shawe-Taylor, J., Szedmak, S.: Two View Learning: SVM-2K, Theory and Practice. Advances in Neural Information Processing Systems 18, 355–362 (2006)
13. Tikhonov, A.N.: Regularization of Incorrectly Posed Problems. Soviet Mathematics Doklady 4, 1624–1627 (1963)
14. Evgeniou, T., Pontil, M., Poggio, T.: Regularization Networks and Support Vector Machines. Advances in Computational Mathematics 13, 1–50 (2000)
15. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, Cambridge (2004)
16. Belkin, M., Niyogi, P.: Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. Neural Computation 15, 1373–1396 (2003)
17. Blum, A., Mitchell, T.: Combining Labeled and Unlabeled Data with Co-training. In: Proceedings of the 11th Annual Conference on Computational Learning Theory, pp. 92–100 (1998)
18. Aronszajn, N.: Theory of Reproducing Kernels. Transactions of the American Mathematical Society 68, 337–404 (1950)
19. Sindhwani, V., Rosenberg, D.: An RKHS for Multi-view Learning and Manifold Co-regularization. In: Proceedings of the 25th International Conference on Machine Learning, pp. 976–983 (2008)
20. Kimeldorf, G., Wahba, G.: Some Results on Tchebycheffian Spline Functions. Journal of Mathematical Analysis and Applications 33, 82–95 (1971)
21. Rosenberg, D.: Semi-Supervised Learning with Multiple Views. PhD dissertation, Department of Statistics, University of California, Berkeley (2008)
22. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
23. Bartlett, P., Mendelson, S.: Rademacher and Gaussian Complexities: Risk Bounds and Structural Results. Journal of Machine Learning Research 3, 463–482 (2002)
24. Rosenberg, D., Bartlett, P.: The Rademacher Complexity of Co-regularized Kernel Classes. In: Proceedings of the 11th International Conference on Artificial Intelligence and Statistics, pp. 396–403 (2007)

25. Latala, R., Oleszkiewicz, K.: On the Best Constant in the Khintchine-Kahane Inequality. Studia Mathematica 109, 101–104 (1994)
26. Porter, M.F.: An Algorithm for Suffix Stripping. Program 14, 130–137 (1980)
27. Salton, G., Buckley, C.: Term-Weighting Approaches in Automatic Text Retrieval. Information Processing and Management 24, 513–523 (1988)
28. Sun, S.: Semantic Features for Multi-view Semi-supervised and Active Learning of Text Classification. In: Proceedings of the IEEE International Conference on Data Mining Workshops, pp. 731–735 (2008)
29. Rosenberg, D., Sindhwani, V., Bartlett, P., Niyogi, P.: Multiview Point Cloud Kernels for Semisupervised Learning. IEEE Signal Processing Magazine, 145–150 (2009)
30. Hsu, C.W., Lin, C.J.: A Comparison of Methods for Multiclass Support Vector Machines. IEEE Transactions on Neural Networks 13, 415–425 (2002)

New Developments of Determinacy Analysis

Rein Kuusik and Grete Lind

Department of Informatics, Tallinn University of Technology
Raja 15, Tallinn 12618, Estonia
kuusik@cc.ttu.ee, grete@staff.ttu.ee

Abstract. This paper deals with development of Determincy Analysis (DA), a method for knowledge discovery. There are three approaches to DA that give different results. The first method finds exactly one system of non-overlapping rules, all with equal length. The second, step by step approach enables to find very many different systems of non-overlapping rules where the rules have different length. The third approach can find one system of overlapping rules. Analysis of these approaches showed that shorter rule systems contain rules which are not contained in other rules. It means that we have to find only all these rules (so called determinative set of rules, DSR) and describe given data table on their basis. This paper presents a new DA approach based on the DSR and algorithm for finding them. Several new tasks are formulated: to generate rule systems with specific features or the shortest or minimal rule system etc.

Keywords: Determinacy Analysis, Data discovery, Rule system, Overlapping rules, Determinative set of rules.

1 Introduction to DA and the Problem Statement

In this paper we deal with the developments of the knowledge discovery method called Determinacy Analysis (DA), the system of methods for the analysis of rules that was created at the end of 1970s. Its approach combines mathematical statistics and logic [1-2].

DA has been arisen from the analysis of frequency tables. The goal of DA is to describe a given selection of objects (class Y) (Who/what are they? How can we describe them? What distinguishes them from others?), not to classify an unknown object. Therefore it is necessary to know which attributes are more determining than the others, not by attribute's descriptive power (for that purpose we could use, for example, entropy like in ID3 or its developments), but from the viewpoint of typical associations of attributes and attributes that are not necessary for describing the system of objects (Y).

Known methods of DA can extract one or more rule systems where the rules do not overlap. The two such kind approaches to DA are: 1) step by step method, 2) DAS.

The initial data table and a feature Y (as a certain class) are given. The goal is to describe Y (possibly) completely by the non-overlapping (possibly) accurate rules.

J. Tang et al. (Eds.): ADMA 2011, Part II, LNAI 7121, pp. 223–236, 2011.

In step by step approach described in [3] the extracted rules can have different lengths while they do not overlap. Attributes are added into the rules one by one. If a rule is accurate it is not expanded by adding the next factor (i.e. attribute with certain value). At the same time the completenesses of found accurate rules are summed up. Reaching 100% the coverage is found. The user decides about the order in which the attributes are included into the rules, from the beginning until the situation when all objects of the class are covered. The order of attributes is essential, different orders lead to the different results.

DAS extracts the rules with equal length, all of them contain the same attributes – this is a simple way to get a set of non-overlapping rules.

Both approaches have problems with the amount of rules, zero factors in rules and the length of rules.

The approach of DA which can find one rule system consisting of overlapping rules has been developed [4]. But there is a problem with the best criteria selection in the process of choosing the next attribute for the rule, there is no guarantee that the extracted rule system will be the shortest one (with the lowest number of rules).

We deal with these problems, analyze them and describe a new approach to DA which can solve these problems.

1.1 Determination and Its Characteristics

DA-technology assists in obtaining regularities, explanations and description rules. DA has been used in sociology [5], linguistics [6], medicine [7], and other areas (for the complete list of references see [8] or [9]).

The overview of determinacy analysis is based on [1-2], [9-10].

The main idea behind DA is that a rule can be found based on the frequencies of joint occurrence or non-occurrence of events. Such rule is called a determinacy or determination, and the mathematical theory of such rules is called determinacy analysis [9].

If it is observable that an occurrence of X is always followed by an occurrence of Y, this means that there exists a rule "If X then Y", or $X \rightarrow Y$. Such correlation between X and Y is called a *determination* (from X to Y). Here X is a *determinative (determining)* and Y is a *determinable*.

The determinative (X) consists of one or more factors. A factor is an attribute with its certain value. Each attribute can give as many factors as many different values it has. The factors coming from the same attribute are not contained in the same X.

Each rule has two characteristics: accuracy and completeness[1].

The accuracy of the determination $X \rightarrow Y$ shows to what extent X determines Y. It is defined as a proportion of occurrences of Y among the occurrences of X: $A(X \rightarrow Y) = n(X\ Y) / n(X)$, where $A(X \rightarrow Y)$ is the accuracy of determination, n(X) is the number of objects having feature X and $n(X\ Y)$ is the number of objects having both features X and Y.

[1] In the beginning (in [1], for example) "accuracy" (Russian "*точность*") was called "intensity" and "completeness" ("*полнота*") was called "capacity" ("*емкость*").

The completeness of the determination X→Y shows which part of the objects having Y can be explained by the determination X→Y. It is a percentage of occurrences of X among the occurrences of Y: C(X→Y) = n(X Y) / n(Y), where C(X→Y) is the completeness of determination, n(Y) is the number of objects having feature Y and n(X Y) is the number of objects having both features X and Y.

Both accuracy and completeness can have values from 0 to 1. Value 1 shows maximal accuracy or completeness, 0 means that rule is not accurate or complete at all. Value between 0 and 1 shows quasideterminism.

If all objects having feature X have also feature Y then the determination is (maximally) accurate. In case of an accurate determination A(X→Y) = 1 (100%). Most of the rules are not accurate. In case of an inaccurate rule A(X→Y) < 1. In order to make a determination more (or less) accurate the complementary factors are added into the left part of the rule. Adding the factor Z into the rule X→Y we get a rule XZ→Y.

The contribution of factor Z *to accuracy* of rule XZ→Y is measured by the increase of accuracy ΔA(Z) caused by addition of factor Z into the rule X→Y: ΔA(Z) = A(XZ→Y) – A(X→Y). The contribution to accuracy falls into interval from -1 to 1.

If ΔA(Z)>0 then Z is a positive factor. Addition of a positive factor makes the rule more accurate, sometimes the resultant rule is (maximally) accurate. If ΔA(Z)<0 then Z is a negative factor. Addition of a negative factor decreases the rule's accuracy, sometimes until zero. If ΔA(Z)=0 then Z is a zero (or inessential) factor. Addition of a zero factor does not change the rule's accuracy. *An accurate rule* contains no negative factors, all factors are positive or zero factors [10].

If C(X→Y)=1 (100%) then the rule X→Y is (maximally) complete. It means that Y is always explained by X. In case of an incomplete rule C(X→Y)<1, it means that X does not explain all occurrences of Y.

The contribution of factor Z *to completeness* of rule XZ→Y is measured by the increase of completeness ΔC(Z) by addition of factor Z into the rule X→Y: ΔC(Z) = C(XZ→Y) – C(X→Y). The contribution of whatever factor to completeness is negative or zero [10].

The system of rules is a set of rules in the form of $S_q = \{x_i \rightarrow y \mid i=1,2,...,q\}$, where q is the number of rules. Every system is characterised by average accuracy, summarised completeness and summarised capacity (the number of objects covered by the rules).

The system of rules $S_q = \{x_i \rightarrow y \mid i=1,2,...,q\}$ is *additive* when x_i-s pairwise do not overlap. The completeness and capacity of the additive system are just summed up completenesses and capacities of the rules. The accuracy of the additive system is not additive (i.e. equal to the sum of rules' accuracies), it is found as a weighted average.

A system is called *complete* if its completeness is 1. A s*ystem* is called *accurate* if its accuracy is 1. A system is accurate when all of its rules are accurate.

The rank of a rule is a dimension of its left side. A rule in the form of $z_1z_2...z_r \rightarrow y$ is called a rule of rank r ($r \geq 1$). A system of rules in the form of $z_1z_2...z_r \rightarrow y$ is called a system of rules of rank r by variables z_1, z_2, ..., z_r relative to the feature y. Every system of rules of rank $r \geq 1$ by fixed set of r variables is additive [10].

The theory of DA covers also the non-additive systems of (overlapping) rules, the first developments, an algorithm for finding such systems is published in [4].

1.2 Example

The following example illustrates the step by step approach. The data from [11] will be used in example (see Table 1). This data table (object-attribute system) contains 14 objects described by 5 attributes – the weather conditions of Saturday morning. The last attribute shows the object's belonging to a certain class (feature Y) - suitability for some unspecified activity (P – positive instances i.e. suitable conditions, N – negative/unsuitable). Attributes Outlook and Temperature have both three alternative values, attributes Humidity, Windy (and Class) have two possible values each.

We will describe objects belonging to class "N". This class consists of five objects (n(Y)=5). Attributes will be added into the rules in the following (freely chosen) order: Outlook, then Humidity, then Windy and last Temperature.

Table 1. Saturday morning data

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
Overcast	cool	normal	true	P
Sunny	mild	high	false	N
Sunny	cool	normal	false	P
Rain	mild	normal	false	P
Sunny	mild	normal	true	P
Overcast	mild	high	true	P
Overcast	hot	normal	false	P
Rain	mild	high	true	N

Rules containing attribute Outlook only are given in Table 2.

Table 2. The rules consisting of attribute Outlook

Outlook	n(X)	n(XY)	A	C	∑C
Sunny	5	3	3/5	3/5	
overcast	4	0	0	0	
Rain	5	2	2/5	2/5	

The rule with Outlook.overcast has zero accuracy (i.e. does not exist) in given class and will not be expanded. Two other rules have accuracy between 0 and 1, thus we expand them by adding the next attribute (Humidity) into rules (see Table 3).

Table 3. The rules consisting of attributes Outlook and Humidity

Outlook	Humidity	n(X)	n(XY)	A	C	∑C
Sunny	**High**	**3**	**3**	**1**	**3/5**	**3/5**
Sunny	Normal	2	0	0	0	
Rain	High	2	1	1/2	1/5	
Rain	Normal	3	1	1/3	1/5	

The rule Outlook.sunny&Humidity.high→Class.N is accurate (A=1) and needs no additional factors, it is included into the result. It covers 60% of the objects of the class (C=3/5). The completenesses C of the accurate rules are summed up (ΣC), with hope to reach the 100% coverage (by accurate rules). The rule with Outlook.sunny and Humidity.normal has zero accuracy in given class and will not be expanded.

Two last rules are expanded by adding attribute Windy (see Table 4).

Table 4. The rules consisting of attributes Outlook, Humidity and Windy

Outlook	Humidity	Windy	n(X)	n(XY)	A	C	∑C
Rain	**high**	**true**	**1**	**1**	**1**	**1/5**	**4/5**
Rain	high	false	1	0	0	0	
Rain	**normal**	**true**	**1**	**1**	**1**	**1/5**	**1**
Rain	normal	false	2	0	0	0	

From here we find two accurate rules, both with completeness 20%. Now the sum of (accurate) completenesses is 100% which means that the class "N" is (fully) covered by the (accurate) rules. At the same time we can see that two other rules (with Windy.false) have zero accuracies and there is no reason to expand them.

As the found rules cover the class "N" completely there is no need to use the next attribute (Temperature), so the class is described without using it.

Class "N" is covered by three rules (denote this system of rules by *S1*):

1. Outlook.sunny & Humidity.high → Class.N (C = 60%);
2. Outlook.rain & Humidity.high & Windy.true → Class.N (C = 20%);
3. Outlook.rain & Humidity. normal & Windy.true → Class.N (C = 20%).

This is one possible description for class "N" beginning from the attribute Outlook. In the "language" of DA it means that these are saturdays with sunny outlook and high humidity or rainy outlook, high humidity and windy weather or rainy outlook, normal humidity and windy weather. The user has to know what to do with this knowledge. If "N" means "not suitable for certain activity", for example, then if the user is satisfied with such description then (s)he accepts it. If this knowledge is not satisfying then the user can change the order of attributes or add some new attributes.

For example, when the used order of attributes is Temperature, Outlook, Windy and Humidity, then we get four rules (in which attribute Humidity is not used):

1. Temperature.hot & Outlook.sunny → Class.N (C=40%);
2. Temperature.mild & Outlook.sunny & Windy.false → Class.N (C=20%);
3. Temperature.mild & Outlook.rain & Windy.true → Class.N (C=20%);
4. Temperature.cool & Outlook.rain & Windy.true → Class.N (C=20%).

1.3 Problems

In order to describe the problems we first present some results (rule systems) using all approaches of DA on the data from [11] (see Table 1). The purpose is to determine Class "N" by rules containing attributes Outlook, Humidity and Windy. Analysis is given from two aspects: interpretation of results and methods' procedural logic. Analyst is also interested in finding of several shorter rule systems to describe Y.

- The additive system consisting of accurate rules of different rank (number of factors) got using step by step approach can be *S1* consisting of 3 rules (see section 2.2) or (*S2*):

1. Outlook.rain & Windy.true → Class.N (C = 40%);
2. Outlook.sunny & Windy.true & Humidity.high → Class.N (C = 20%);
3. Outlook.sunny & Windy.false & Humidity.high → Class.N (C = 40%).

S1 is found when attributes are included in the order: first Outlook, then Humidity, then Windy. In case of *S2* the order is: first Outlook, then Windy and then Humidity. It is interesting to mention that the first rule of *S1* corresponds to two longer rules of *S2* containing values of attribute Windy as zero factors. Also, the first rule of *S2* is divided into two longer rules of S1 containing values of attribute Humidity as zero factors. In both cases the system is accurate and complete and the rules do not cover the same objects. As we can see the different rule systems describe the objects practically differently giving a different knowledge about the subset of objects. The user decides how many rule systems is enough. If the description given by the rule systems is not good enough then some other set of attributes can be chosen or just some attributes can be exchanged. As we can see, the more the rule system contains the rules, that are not contained in other rules, the shorter the rule system is.

- The additive system with a fixed set of factors (rank r=3) got using DAS (*S3*):

1. Outlook.sunny & Humidity.high & Windy.true → Class.N (C = 20%);
2. Outlook.sunny & Humidity.high & Windy.false → Class.N (C = 40%);
3. Outlook.rain & Humidity.high & Windy.true → Class.N (C = 20%);
4. Outlook.rain & Humidity. normal & Windy.true → Class.N (C = 20%).

Compared to the result of step-by-step approach the first rule of *S1* is divided into two and these two contain values of attribute Windy as zero factors and similarily the first rule fo *S2* is divided into two rules containing zero factors (values of Humidity).

- The non-additive system with overlapping rules got using algorithm described in [4] consists of 2 rules (*S4*):

1. Outlook.sunny & Humidity.high → Class.N (C = 60%);
2. Outlook.rain & Windy.true → Class.N (C=40%).

Although this algorithm can find overlapping rules, in this case the rules do not overlap, each instance of class "N" is covered exactly once; there are no zero factors. As we see it is the shortest rule system.

Looking at these examples we can see that an additive system containing rules with different lengths (both *S1* and *S2*) has lesser number of rules than an additive system with a fixed rank and set of attributes (*S3*). *S3* contains two longer rules from *S1* and two longer rules from *S2*. The set of overlapping rules (*S4*) has the least number of rules and also the shortest rules, it contains the shortest rule from *S1* and the shortest rule from *S2*. Also we can say that shorter rule systems consist of rules which are not contained in other rules.

The task to find an additive system of rules can be understood in several ways:

1. In case of DAS-approach, all the rules contain all given attributes.
2. In case of step by step approach, it is possible to find a system of rules of different length. Different order of inclusion of attributes can give different systems of rules.

In the first case, on the assumption of the essence of the DAS approach, there is exactly one solution. If each rule has to contain all (given) attributes, then we cannot take away any factors from any rules and consequently we cannot change such system of rules. We loose also a great amount of knowledge because of not knowing specific information about attributes' associations (by 2, 3 or more attributes). Determining a different set of attributes (for description) we get a different system which is also the only possible solution for that set of attributes (in case of the same requirement – to contain all attributes). In this case the purpose is to find the most suitable set of attributes i.e. exclude redundant attributes (like Temperature in our example) and include all necessary attributes to avoid contradictions – a situation where the set of attributes is not sufficient to distinguish different classes. A suitable set can be found by testing different sets of attributes.

In the second case – non-overlapping rules (i.e. additive system) of different length (rank) – the constitution of the system depends on the order in which the factors are included into the rules. It might be hard to find a compact system of such rules. In this case there is a huge amount of different orders of attributes in a given set of attributes by 1, 2, 3, ..., M attributes. It is not real to extract all these rule systems and to analyze them. It would be good to allow non-fixed order of including factors.

The recent approach to DA emanates from logic: if one rule system does not satisfy then the next one will be considered etc. Such treatment comes from the fact that the rules of one rule system are derived from the uniform order of attributes and thus they are mutually related. It means that we cannot change the order of attributes in the emergent rule system. This is an important restriction, according to it the best description in a sense can form from single rules of different rule systems. One solution here is to generate all rule systems and according to some criteria find from them the best cover i.e. a rule set (with completeness 100%) consisting of the shortest rules or a rule

set with the least number of rules. This turns out to be very labor-consuming because all the possible sets and orders (permutations) of attributes should be found.

DA development for overlapping rules can automatically generate one rule system, but there is no guarantee that the extracted rule system is the shortest (with a lowest number of rules) (or several rule systems when changing criteria for choosing the factor for making extract).

Altogether we can say that 1) shorter rule systems mostly consist of rules which are not contained in other rules and 2) the existing approaches to DA do not include an effective method for finding shorter rule systems.

2 New Approach

In the previous section we mentioned that shorter rule systems consist of rules which are not contained in other rules. It means that for constructing shorter rule systems we need only rules of such type. But how to find them?

Next we present a new approach of DA which gives a solution to this problem. At first we define a new concept "Determinative set of rules", then describe an algorithm that can find it and describe how we can use this rule set for further analysis.

2.1 Basis of the New Approach

Let a table X(N,M) be given and a set B of all possible rules describing (only) the class Y and each rule in B is presented only once.

The Determinative set of rules (DSR) for class Y consists of all rules which are not contained in other rules of B.

B = {Ri}, i=1, 2,..., K, where K is a number of all possible rules describing (only) the Class Y. $Ri \neq Rj$, $i \neq j$.
DSR = {Ru}. $Ru \in$ DSR if there $/\exists Ri \in B$, $Ru \subset Ri$, $i \neq u$. DSR $\subseteq$ B

It means that DSR does not contain subrules of its rules. To get DSR from B we have to throw out all subrules of rules. We call this process „rule set compression".

Example. Let B contain 4 rules (all possible rules for Y = (Class =1)):

1. r1: IF T1=1 & T2=1 THEN CLASS=1
2. r2: IF T1=1 & T3=2 THEN CLASS=1
3. r3: IF T2=1 THEN CLASS=1
4. r4: IF T3=2 THEN CLASS=1

As we see, the rule r1 is contained in r3 and r2 is contained in r4. According to the definition DSR_B = {r3, r4}.

The main features of DSR are:

1. there are no redundant attributes (zero factors) in rules,
2. the same object in the class Y can be described by several rules.

On the basis of DSR we can form and solve next tasks, for example, to find

1. the shortest rules (by the number of attributes in the rule),
2. the longest rules (by the number of attributes in the rule in DSR),
3. the rules with specific features (for example, all rules of the rank r in DSR),
4. the shortest rule system (i.e. the rule system with the smallest number of rules),
5. the rule system which consists of rules with minimal ranks,
6. all the rule systems we can form on the basis of DSR.

The tasks 1-3 are easily solvable, but tasks 4-6 are essentially system covering tasks and they are NP-complex tasks.

2.2 Description of the Algorithm

Here we describe the algorithm realizing the new approach to DA. The findable set of rules is DSR together with some redundant rules which are eliminated afterwards.

This is a depth-first-search algorithm that makes subsequent extracts of objects containing certain factors. At each level first of all the rules (of that extract) are detected and then factors for making extracts of the next level are selected one by one.

The algorithm uses frequency tables for X_t (all objects of current extract) and Y_t (objects belonging to observable class of current extract), Fx_t and Fy_t accordingly. If there are equal frequencies in both frequency tables for some factor then this factor completes a rule. The rule includes also the factors chosen on the way to that extract.

The selection criteria for choosing the next factor are based on frequencies, the maximal frequency in Fy_t. In case of equal maximal frequencies in Fy_t the one having lower frequency in Fx_t is preferred. If only one attribute (of the extract) has free (unused) value(s) (indicated by frequencies over zero in Fy) then it is not practical to make a next (further) extract because there would be no free factors to distinguish objects of different classes in that extract. If there are no free factors (i.e. no frequencies over zero) then obviously it is not possible to make a next extract. In both cases the algorithm backtracks to the previous level.

Each factor that is used for making an extract or completing a rule is set to zero in the corresponding Fy. Each Fy (except for the initial level) inherits all zeroes of the previous level (we call it “bringing zeroes down”). These zeroing techniques prevent many redundant extracts and rules.

```
Algorithm
Determine tables X and Y
S0. t:=0; U_t:=∅
S1. Find frequencies in tables X_t and Y_t: Fx_t, Fy_t
    If t>0 then
        For each factor A such that Fy_{t-1}(A)= 0
            Fy_t(A):= 0
S2. For each factor A such that Fy_t(A)=Fx_t(A)
        output rule {U_i}&A, i=0,…,t; Fy_t(A):= 0
S3. If not enough free factors for making extract then
```

```
        If t=0 then Goto End
        Else t:=t-1; Goto S3
S4. Choose a new (free) factor U_t
    Fy_t(A):= 0;
    t:=t+1; extract subtable of objects containing U_t;
    Goto S1
End. System of rules is found
```

2.3 Example

In the examples we use the same Saturday morning's data [11] as in section 1.2 (see Table 1), but this time in a numerical representation. The coding used for attributes and their values is shown in Table 5. The initial data table is given in Table 6. Let X is X(14,4), X_{ij} = 1,...,3 and Y=5.2 {Y_i: 1,2,6,8,14} (i.e. class "N"). The frequencies of attributes' values for X and Y (Fx and Fy accordingly) are given in Table 7.

Table 5. The coding used for Saturday morning data

Attribute	**Outlook**	**Temperature**	**Humidity**	**Windy**	**Class**
Code	1	2	3	4	5
1	sunny	cool	high	true	P
2	overcast	mild	normal	false	N
3	rain	hot			

Table 6. The initial data table

	1	2	3	4	5
1	1	3	1	2	2
2	1	3	1	1	2
3	2	3	1	2	1
4	3	2	1	2	1
5	3	1	2	2	1
6	3	1	2	1	2
7	2	1	2	1	1
8	1	2	1	2	2
9	1	1	2	2	1
10	3	2	2	2	1
11	1	2	2	1	1
12	2	2	1	1	1
13	2	3	2	2	1
14	3	2	1	1	2

Table 7. The frequencies for X and Y

Fx	1	2	3	4	Fy	1	2	3	4
1	5	4	7	6	1	3	1	4	3
2	4	6	7	8	2	0	2	1	2
3	5	4	0	0	3	2	2	0	0

There are no equal frequencies (over zero) in initial frequency tables. A factor that will be the basis for making an extract is chosen by the biggest value in Fy. This is 3.1 (attribute 3 with value 1) with frequency =4. Extract (subtable of the table X) by 3.1 is shown in Table 8 and the corresponding frequency tables in Table 9.

Table 8. Extract by 3.1 (Humidity.high)

	1	2	3	4	5
1	1	3	1	2	2
2	1	3	1	1	2
3	2	3	1	2	1
4	3	2	1	2	1
8	1	2	1	2	2
12	2	2	1	1	1
14	3	2	1	1	2

Table 9. The frequencies of extracted data

Fx	1	2	3	4	Fy	1	2	3	4
1	3	0	7	3	1	3	0	4	2
2	2	4	0	4	2	0	2	0	2
3	2	3	0	0	3	1	2	0	0

For 1.1 (Outlook.sunny) the frequencies in Fx and Fy are equal which means that all objects (of extract by 3.1) having 1.1 belong to Y. Hence we get a rule R1: 3.1&1.1=3 (Humidity.high&Outlook.sunny). The frequency after "=" shows that the rule covers three objects (namely 1, 2 and 8) – this is additional information. The frequency of 1.1 is set to zero in current Fy to avoid using it as a basis for making next extract(s).

Now we have to choose a factor for making a next extract. In Fy four different factors have the maximal frequency (=2): 2.2; 2.3; 4.1; 4.2. In Fx two of them (2.3 and 4.1) have frequency 3 and two have frequency 4 (2.2; 4.2). Choice is made from those which have lower frequency in Fx. The first one with frequency 3 (in Fx) is 2.3. Extract by (3.1 and) 2.3 is in Table 10 and corresponding frequencies in Table 11.

Table 10. Extract by 3.1 (Humidity.high) and 2.3 (Temperature.hot)

	1	2	3	4	5
1	1	3	1	2	2
2	1	3	1	1	2
3	2	3	1	2	1

Table 11. The frequencies of extracted data

Fx	1	2	3	4	Fy	1	2	3	4
1	2	0	3	1	1	0	0	2	1
2	1	0	0	2	2	0	0	0	1
3	0	3	0	0	3	0	2	0	0

Every frequency table for Y (Fy) inherits all zeroes of the previous level, to avoid repetitious extracts and redundant rules ("bringing zeroes down"). Therefore the actual frequency of 1.1 in Fy (=2) is replaced by 0. If this frequency was not set to zero we would find the rule 3.1&2.3&1.1=2 which is a subrule of the already found rule 3.1&1.1. From the current extract we find the rule R2: 3.1&2.3&4.1=1 (Humidity. High & Temperature.hot&Windy.true). The frequency of 4.1 is set to zero after that. Now there is only one frequency over zero in Fy (4.2=1). Making an extract by only free attribute (factor) cannot give a rule because there are no attributes to use for distinguishing between different classes at the next level. So the algorithm goes back to the previous level (extract by 3.1 – see Table 8).

The frequency table Fy of that level (see Table 9) has got two zeroes meanwhile: 1.1 has been set to zero when the rule with it was extracted and 2.3 has been set to zero due to being the basis for extract. The next (unused) factor with frequency 2 in Fy and frequency 3 in Fx is 4.1. The next extract is made by 4.1. This extract contains objects 2, 12 and 14. The frequencies are shown in Table 12.

Table 12. The frequencies of extract by 3.1 (Humidity.high) and 4.1 (Windy.true)

Fx	1	2	3	4	Fy	1	2	3	4
1	1	0	3	3	1	0	0	2	2
2	1	2	0	0	2	0	1	0	0
3	1	1	0	0	3	1	0	0	0

From that level we get the rule R3: 3.1&4.1&1.3=1 (Humidity.high&Windy.true& Outlook.rain). After zeroing the frequency of 1.3 in Fy there is only one unused frequency over zero in Fy (2.2=1) and algorithm backtracks to the previous level. The frequency table Fy of extract by 3.1 (Table 8) has now one more zero (4.1), its current state is given in Table 13.

Table 13. Frequnecy tables for extract by 3.1

Fx	1	2	3	4	Fy	1	2	3	4
1	3	0	7	3	1	0	0	4	0
2	2	4	0	4	2	0	2	0	2
3	2	3	0	0	3	1	0	0	0

Factors 2.2 and 4.2 have equal frequencies in both tables (2 in Fy and 4 in Fx). First 2.2 is chosen to make the next extract. This extract (containing objects 4, 8, 12 and 14), its sub-extract by 1.3 (objects 4 and 14) and also extract by 4.2 (objects 1, 3, 4, 8) give no rules.

Algorithm is back at the initial level. The only thing that is changed compared to the state shown in Table 7 is that the frequency of 3.1 is set to zero in the initial Fy.

The work continues the same way. Below the whole tree of extracts made during the work is given. Making extract by 3.1 and its sub-extracts have been described already (see tables 7-13).

After "=" are shown frequencies Fy/Fx of extract. Rn after "⇒" indicates a found rule. On the right the objects contained in extract are listed.

```
3.1 (Humidity.high)=4/7 ⇒ R1          1,2,3,4,8,12,14
  & 2.3 (Temperature.hot)=2/3 ⇒ R2    1,2,3
  & 4.1 (Windy.true)=2/3 ⇒ R3         2,12,14
  & 2.2 (Temperature.mild)=2/4        4,8,12,14
    & 1.3 (Outlook.rain)=1/2          4,14
  & 4.2 (Windy.false)=2/4             1,3,4,8
1.1 (Outlook.sunny)=3/5 ⇒ R4          1,2,8,9,11
  & 4.2 (Windy.false)=2/3 ⇒ R5        1,8,9
  & 2.2 (Temperature.mild)=1/2        8,11
4.1 (Windy.true)=3/6 ⇒ R6, R7         2,6,7,11,12,14
  & 2.1 (Temperature.cool)=1/2        6,7
  & 2.2 (Temperature.mild)=1/3        11,12,14
2.3 (Temperature.hot)=2/4             1,2,3,13
1.3 (Outlook.rain)=2/5                4,5,6,10,14
  & 2.1 (Temperature.cool)=1/2        5,6
  & 2.2 (Temperature.mild)=1/3        4,10,14
2.2 (Temperature.mild)=2/6            4,8,10,11,12,14
4.2 (Windy.false)=2/8                 1,2,3,4,5,8,9,10,13
2.1 (Temperature.cool)=1/4            5,6,7,9
```

As a result 7 rules have been found:

R1: 3.1&1.1=3 (Humidity.high&Outlook.sunny→Class.N)
R2: 3.1&2.3&4.1=1 (Humidity.high&Temperature.hot&Windy.true→Class.N)
R3: 3.1&4.1&1.3=1 (Humidity.high&Windy.true&Outlook.rain→Class.N)
R4: 1.1&2.3=2 (Outlook.sunny&Temperature.hot→Class.N)
R5: 1.1&4.2&2.2=1 (Outlook.sunny &Windy.false&Temperature.mild→Class.N)
R6: 4.1&1.3=2 (Windy.true&Outlook.rain→Class.N)
R7: 4.1&2.3=1 (Windy.true&Temperature.hot→Class.N)

Two of them are redundant: R2 is a subrule of R7 (3.1 (Humidity.high) is a zero factor here), and R3 is a subrule of R6 (due to redundant factor 3.1 (Humidity.high) R3 covers one object less than R6). Subrule of some rule is always found before the rule itself. Subrules that could be found afterwards are prevented by zeroing techniques.

After the rule set compression we got DSR: R1, R4, R5, R6 and R7. On the basis of these five rules (DSR) we can solve several tasks mentioned in section 2.1.

3 Conclusion

This paper gives an overview of Determinacy Analysis, introducing the theory and different approaches, proposes a new method and the corresponding algorithm for finding determinative set of overlapping rules and several ideas how to use it. The algorithm is based on frequency tables which makes it easy to detect a potential zero factor free DSR rule. Finally we can say the following.

1. Somebody might say that the finding of DSR is very laborious, especially in cases of large amounts of data. If so, user can decide, what is the purpose of the work. If the purpose is quick one-time information gathering for a data set under analysis, then the use of DSR-based DA approach maybe not the best one. But if the purpose is to describe the data set and through that discover a new knowledge, then this DA approach is a good solution, because it enables the post analysis of DSR to discover more new knowledge. There is no need to experiment with several attributes and their different orders to get more knowledge.
2. It is mostly desired that the rules are relatively short and/or the number of rules has not to be very large, because of easier interpretation of them. On the other hand, if we want to describe the data, we must know how several attributes act together, what are the typical co-occurrences of attributes, what attributes do not act with others etc. The earlier methods of DA described in the paper can't discover such kind of knowledge, but the new one presented here creates a good basis for that.

The post-analysis of rules will be the topic of the next paper.

References

1. Chesnokov, S.V.: Determination-Analysis of Social-Economic Data in Dialogical Regime. Preprint. All-Union Institute for Systems Research, Moscow (1980) (in Russian)
2. Chesnokov, S.V.: Determinacy Analysis of Social-Economic Data, Nauka, Moscow (1982) (in Russian)
3. Lind, G., Kuusik, R.: Some Ideas for Determinacy Analysis Realisation. In: Proceedings of the 11th IASTED International Conference on Artificial Intelligence and Soft Computing, pp. 185–190. ACTA Press (2007)
4. Kuusik, R., Lind, G.: Some Developments of Determinacy Analysis. In: Cao, L., Zhong, J., Feng, Y. (eds.) ADMA 2010, Part I. LNCS, vol. 6440, pp. 593–602. Springer, Heidelberg (2010)
5. Chesnokov, S.V.: Determinacy Analysis of Social-Economic Data. Sociological Studies 3, 179–189 (1980) (in Russian)
6. Luelsdorff, P.A., Chesnokov, S.V.: Determinacy Form as the Essence of Language. Prague Linguistic Circle Papers 2, 205–234 (1996)
7. Chesnokov, S.V.: Determinacy Analysis and the Search for Diagnostic Criteria in Medicine (the Case of Comprehensive Ultrasonography). Ultrasonic Diagnostics 4, 42–47 (1996) (in Russian)
8. Main scientific works of S.V. Chesnokov (in Russian), `http://www.context.ru/publications`
9. DA-system 4.0, version 4.0 for Windows 95, Windows 98 and Windows NT. Questions and Answers. DA-system and Technology of Data Analysis. "Context" (1999) (in Russian)
10. Chesnokov, S.V.: Determinacy Analysis of Socio-Economic Data. Illustrative Materials to Lectures. Lecture 2: Rules. Lecture 3: Systems of Rules. Lomonosov Moscow State University, Faculty of Economics, Moscow (2002) (unpublished, in Russian)
11. Quinlan, J.R.: Induction of Decision Trees. Machine Learning 1, 81–106 (1986)

On Mining Anomalous Patterns in Road Traffic Streams

Linsey Xiaolin Pang[1,4], Sanjay Chawla[1], Wei Liu[2], and Yu Zheng[3]

[1] School of Information Technologies, University of Sydney, Australia
[2] Dept. of Computer Science and Software Engineering, University of Melbourne, Australia
[3] Web Search and Mining Group, Microsoft Research Asia, Beijing, China
[4] NICTA, Sydney, Australia
qlinsey@it.usyd.edu.au, sanjay.chawla@sydney.edu.au, wei.liu@unimelb.edu.au, yuzheng@microsoft.com

Abstract. Large number of taxicabs in major metropolitan cities are now equipped with a GPS device. Since taxis are on the road nearly twenty four hours a day (with drivers changing shifts), they can now act as reliable sensors to monitor the behavior of traffic. In this paper we use GPS data from taxis to monitor the emergence of unexpected behavior in the Beijing metropolitan area. We adapt likelihood ratio tests (LRT) which have previously been mostly used in epidemiological studies to describe traffic patterns. To the best of our knowledge the use of LRT in traffic domain is not only novel but results in very accurate and rapid detection of anomalous behavior.

Keywords: Spatio-temporal outlier, persistent, emerging, upper-bounding.

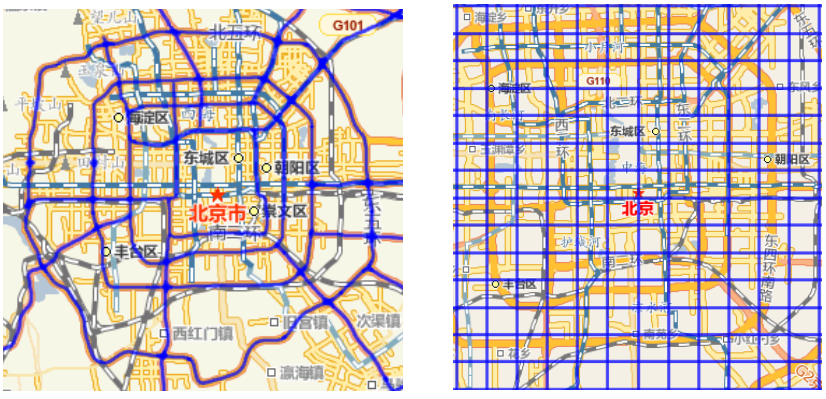
1 Introduction

Thousands of taxis ply the roads of large metropolitan cities like New York, London, Beijing and Tokyo every day. Most taxis are on the road twenty four hours a day with drivers changing shifts. Many of these taxis are now equipped with GPS and their spatio-temporal coordinates are available. Thus if a city is partitioned into a grid then at a given time we can estimate the count of the number of taxis in the grid cells. Over time, the cell counts will settle into a pattern and vary periodically. For example, during morning rush hour more taxis will be concentrated in business districts than at other times of the day. Similarly taxi counts near airports will synchronize with aircraft arrival and departure schedules. Occasionally there will be a departure of the cells counts from periodic behavior due to unforeseen events like vehicle breakdowns or onetime events like big sporting events, fairs and conventions.

Our objective is to identify contiguous set of cells and time intervals which have the largest statistically significant departure from expected behavior.

Once such regions and time intervals have been discovered then experts can begin identifying events which may have caused the unexpected behavior. This

J. Tang et al. (Eds.): ADMA 2011, Part II, LNAI 7121, pp. 237–251, 2011.



(a) Road Network of Beijing (b) Grid Map

Fig. 1. An example of the traffic network of Beijing. Based on the longitude and latitude, the entire city is partitioned into a grid map. Subfigure (a) is partitioned into subfigure (b).

in turn can help make provisions to manage future traffic behavior. Similar problems appear in many other domains. For example, government healthcare agencies are interested in detecting emergence of disease patterns which deviate from expected behavior.

The number of contiguous regions and time intervals is very large. For example, if the spatial grid corresponds to a $n \times n$ matrix and there are T time intervals, then there are potentially $O(n^2T)$ spatio-temporal cells and $O(n^4T^2)$ cubic regions. The huge amount of spatio-temporal data, such as taxi count across different grid regions within different time steps from minutes to hours to days, requires an efficient approach to detect spatial-temporal outliers for predicting abnormal events and implementing traffic control measures in advance. For this motivation, we apply road network of Beijing and partition it into grid to find outliers (Fig. 1).

In a paper of particular relevance to our work, the LRT framework [15] states the computation cost for single statistic value as well as enumerating all the spatial regions to be expensive. To avoid performing statistical computations for every region, it provides a pruning strategy based on classical likelihood test statistic. In this paper, we extend the LRT framework to detect abnormal traffic pattern. More specifically, **the contributions** are: (1) A general and efficient pattern mining approach for spatio-temporal outlier detection is proposed. (2) In this work, persistent and emerging outlier detection statistical models are provided. (3) We give our proof that the upper-bounding strategy of LRT is applicable to "persistent" and "emerging" outlier detection models. (4) We conducted experiments on synthetic data to verify the extended pruning approach and show the significant improvement of searching when data set size is large; We also performed real data validation in the detection of emerging taxi count trend due to some major events.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 illustrates the statistic background and upper-bounding methodology for pruning. Section 4 proposes our approach, in which the statistic detection models are provided. The upper-bounding and pruning mechanism in this framework based on our proof are presented in section 5. Computational complexity is also discussed in this section. Section 6 shows the experiments and case studies results. Finally, section 7 concludes this work.

2 Background

2.1 Statistical Background

We provide a brief but self-contained introduction for finding the most anomalous region (rectangle) in a spatial setting . We also explain a pruning strategy which can cut down the number of rectangles that need to be checked. The basic tool to find the anomalous region is the Likelihood Ratio Test (LRT).

Given a data set X, the distribution $P(X, \theta)$, a null hypothesis H_0 and an alternate hypothesis H_1, the LRT is the ratio

$$\lambda = \frac{\sup_\theta\{L(\theta|X)|H_0\}}{\sup_\theta\{L(\theta|X)|H_1\}}$$

where $L()$ is the likelihood function. In a spatial setting the null hypothesis is that the statistical aspects of the phenomenon of interest in a region (that is currently being tested) are no different from their complement. Thus if a region is indeed anomalous then the the alternate hypothesis will most likely be a better fit and the denominator of the λ will have a higher value for θ which is a maximum likelihood estimator. A remarkable fact about λ is that under mild regularity conditions, the asymptotic distribution of $\Lambda \equiv -2 \log \lambda$ follows a χ^2_k distribution with k degrees of freedom, where k is the number of free parameters[1] (see below). Thus regions whose Λ value likes in the tail of χ^2 distribution are likely to be anomalous.

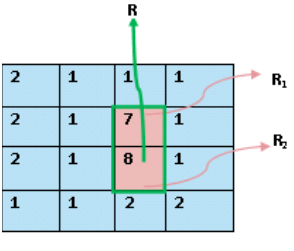


Fig. 2. An example of (4×4) grid to illustrate the LRT calculation and the upper-bounding methodology

[1] If the χ^2 distribution is not applicable then Monte Carlo simulation can be used to ascertain the p-value.

2.1.1 Upper-Bounding Methodology

The basic idea of upper-bounding methodology in LRT [15] is: if region R is composed of two non-overlapping regions R_1 and R_2, then $L(\theta_R|X_R) \leq L(\theta'_{R_1}|X_{R_1}) \times L(\theta'_{R_2}|X_{R_2})$, where θ_R is calculated by performing $MLE_1(R, f(G))$; θ'_{R_1} and θ'_{R_2} are computed directly by performing $MLE_0(f(R_1))$ and $MLE_0(f(R_2))$. And the formula is equivalent to: $logL(\theta_R|X_R) \leq \log L(\theta'_{R_1}|X_{R_1}) + \log L(\theta'_{R_2}|X_{R_2})$. Therefore, the log likelihood of any given region R can be upper-bounded.

2.1.2 Example

We generate a grid of 4×4 in Table 2. The number of successes (k_c) generated by poisson model $P_o(b_c p)$ is displayed in each cell. The baseline b_c in each cell is set to be 10. The success rate of p is 0.5 for the region R and p is 0.1 for the rest of cells. The significant level α=0.05. Here we refer the success rate of p as test parameter.

(1) The likelihood function of each cell is: $f(p|c) = \frac{(b_c p)^k e^{-(b_c p)}}{k!}$

(2) The likelihood of any given region R is: $L(p|R) = \Pi_{c_i \in R} \frac{(b_{c_i} p)^{k_i} e^{-(b_{c_i} p)}}{k_i!}$.

(3) The MLE_0 of p for a region R (denoted as $\hat{p}$), which is composed of cell $c_1, c_2, ..., c_t$, is calculated as: $\hat{p} = \frac{\sum_{i=1}^{t} k_i}{\sum_{i=1}^{t} b_i}$. Thus $\hat{p}_R = \frac{(7+8)}{(10+10)} = 0.75$. Similarly, $\hat{p}_{\bar{R}}$, $\hat{p}_{R_1}$, $\hat{p}_{R_2}$ and $\hat{p}_G$ are obtained as: 0.14, 0.7 ,0.8 and 0.21.

(4) Λ of region R are calculated by the above definition:

$$\Lambda_R = (-2)\log\left(\frac{0.75^{15} \times e^{-0.75\times 20} \times 0.14^{19} \times e^{-0.14\times 140}}{0.21^{19+15} \times e^{-0.21\times 340}}\right) = 20.79$$

From above steps, we get the exact log likelihood value of region R: $\log L(p|R) = -19.31$; and the exact log likelihood of R_1 and R_2: $\log L(p|R_1) = -9.49$, $\log L(p|R_2) = -9.78$ separately.

We know the critical value of $\chi^2(\alpha)$=3.84. Obviously, 20.79 is greater than 3.84. Therefore region R is treated as a potential outlier. As a verification of the upper-bounding strategy, we can see that the sum of log likelihood of R_1 and R_2 is -19.27, which is greater than the exact log likelihood value of R with -19.31.

3 Proposed Framework

3.1 Preliminaries

Definition 1. *KP: It refers to "key parameter", denoted as $KP\{\theta_1, \theta_2, .., \theta_i, .., \theta_n\}$. θ_i is a parameter coming from the key parameter set. For instance, in epidemiology, if we concern about the trend of the disease rate in a spatio-temporal view, the disease rate is KP. For simplicity, we only consider one parameter from the key parameter set in our work (denoted as KP).*

Definition 2. *PSTO (Persistent Spatial-Temporal Outlier): The KP is consistent throughout its duration.*

Definition 3. *ESTO (Emerging Spatial-Temporal Outlier): The KP is non-decreasing throughout its duration until it reaches the peak.*

Definition 4. *Spatio-Temporal Data Cube: A cube is the minimum spatial-temporal unit with relation to spatial and temporal perspective. The data in each cube is based on a statistical model characterized by a probability density function.*

Definition 5. *Temporal Unit: It is the time length that we apply test statistic on every time step.*

Definition 6. *Temporal Interval: It is the period that the user defines to detect outlier.*

3.2 Statistical Models

Definition 7. *PSTO Model (Persistent Spatio-Temporal Outlier Model): It is used to detect persistent spatio-temporal outliers. The null hypothesis H_0 assumes that the KP is consistent for all regions over time. The alternative hypothesis H_1 assumes that KP has a higher value in region $r_i \in \mathrm{R}$ than the value outside of region $r_j \in \text{G-R}$ (i.e. $\bar{R}$), but the value in region $r_i \in \mathrm{R}$ is consistent over time. We calculate the likelihood ratio test as follows:*

$$D(R) = \begin{cases} \frac{\Pi_{r_i \in R} L(\theta_r | X_R) \Pi_{r_i \in \bar{R}} L(\theta_{\bar{r}} | X_{\bar{r}})}{\Pi_{r_i \in G} L(\theta_G | X_G)} & \text{for } \theta_r \geq \theta_{\bar{r}}\,, \\ 1 & \text{otherwise.} \end{cases}$$

This formula is the classical LRT statistic. We first calculate the MLE of θ_r and $\theta_{\bar{r}}$ to maximize the numerator and the MLE of θ_G to maximize the denominator. Then the ratio is the score we use to evaluate the "anomalousness" of a given spatio-temporal region.

Definition 8. *ESTO Model (Emerging Spatio-Temporal Outlier Model): This model is used to detect emerging spatio-temporal outliers. The null hypothesis H_0 assumes that the KP is consistent for all regions over time. The alternative hypothesis H_1 assumes that KP is non-decreasing with every time step over region $r_i \in \mathrm{R}$ and higher than $r_j \in \bar{R}$. We calculate the likelihood ratio test as follows:*

$$D(R) = \begin{cases} \frac{Max_{\theta_{\bar{r}} \leq \theta_{t_{min}} \leq \ldots \leq \theta_T} \Pi_{r_i \in R} L(\theta_r^t | X_r^t) \Pi_{r_i \in \bar{R}} L(\theta_{\bar{r}}^t | X_{\bar{r}}^t)}{\Pi_{r_i \in G} L(\theta_G^t | X_G^t)} & \text{for } \theta_{\bar{r}} \leq \theta_{t_{min}} \leq \ldots \leq \theta_T, \\ 1 & \text{otherwise.} \end{cases}$$

This formula is derived from the classical LRT statistic and designed for the emerging scenario. User needs to find a solution to maximize the numerator with the increasing KP. For instance, Barlow [2] provide an approach to solve the constrained maximum likelihood estimation on the reliability growth model in which the relative risk is non-decreasing over time. Or EM algorithm can be performed to estimate the key parameter.

4 Upper-Bounding Strategy and Pruning Mechanism for Proposed Framework

4.1 Upper-Bounding Strategy

(1) In $PSTO$ model, the upper-bounding strategy explained in section 2.2.2 can be extended directly to spatio-temporal dimension.
(2) In $ESTO$ model, KP is assumed to vary at different time step; we show below that the upper-bounding strategy is still applicable to this model.

Lemma 1. *Let region $R = R_{t1} \cup R_{t2}$ for non-overlapping time interval $t1$ and $t2$, we have:*

$$L(\theta_R|X_R) \leq L(\theta'_{R_{t1}}|X_{R_{t1}}) \times L(\theta'_{R_{t2}}|X_{R_{t2}}) \quad (1)$$

, where $\theta_R = \theta_{R_{t1}} \cup \theta_{R_{t2}}$ and $X_R = X_{R_{t1}} \cup X_{R_{t2}}$

Proof. We know $L(\theta_R|X_R) = L(\theta_{R_{t1}}|X_{R_{t1}}) \times L(\theta_{R_{t2}}|X_{R_{t2}})$. Using the LRT upper-bounding basic concepts, we know that $\theta_{R_{t1}}$ is chosen under more strict complete parameter space and $\theta'_{R_{t1}}$ is chosen under loosen null parameter space. That means performing MLE_0 on a sub-interval of R has loosen the constraints comparing with performing MLE_1 on R. Thus, we have $L(\theta_{R_{t1}}|X_{R_{t1}}) \leq L(\theta'_{R_{t1}}|X_{R_{t1}})$ and $L(\theta_{R_{t2}}|X_{R_{t2}}) \leq L(\theta'_{R_{t2}}|X_{R_{t2}})$. Therefore, $L(\theta_R|X_R) \leq L(\theta'_{R_{t1}}|X_{R_{t1}}) \times L(\theta'_{R_{t2}}|X_{R_{t2}})$.

Lemma 2. *Let region $R = R1 \cup R2$ for non-overlapping spatial region $R1$ and $R2$,we have:*

$$L(\theta_{R1}, \theta_{R2}|X_{R1}, X_{R2}) \leq L(\theta'_{R1_{t1}}, \theta'_{R1_{t2}}|X_{R1_{t1}}, X_{R1_{t2}}) \times L(\theta'_{R2_{t1}}, \theta'_{R2_{t2}}|X_{R2_{t1}}, X_{R2_{t2}}) \quad (2)$$

,where R, $R1$, $R2$ are composed of (t1,t2) time steps respectively. Here we just use two time steps to illustrate. It is applicable to any t time steps.

Proof. For each time step i, we have:$L(\theta_{R_{ti}}|X_{R_{ti}}) \leq L(\theta'_{R1_{ti}}|X_{R1_{ti}}) \times L(\theta'_{R2_{ti}}|X_{R2_{ti}})$

$$L(\theta_{R1}, \theta_{R2}|X_{R1}, X_{R2}) = L(\theta_{R1}|X_{R1}) \times L(\theta_{R2}|X_{R2})$$
$$L(\theta_{R1_{t1}}, \theta_{R1_{t2}}|X_{R1_{t1}}, X_{R1_{t2}}) = L(\theta_{R1_{t1}}|X_{R1_{t1}}) \times L(\theta_{R1_{t2}}|X_{R1_{t2}})$$
$$L(\theta_{R2_{t1}}, \theta_{R2_{t2}}|X_{R2_{t1}}, X_{R2_{t2}}) = L(\theta_{R2_{t1}}|X_{R2_{t2}}) \times L(\theta_{R2_{t2}}|X_{R2_{t2}})$$

Therefore we get

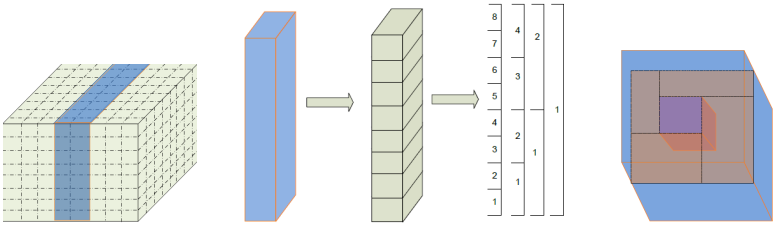
$$L(\theta_{R1}, \theta_{R2}|X_{R1}, X_{R2}) \leq L(\theta'_{R1_{t1}}, \theta'_{R1_{t2}}|X_{R1_{t1}}, X_{R1_{t2}}) \times L(\theta'_{R2_{t1}}, \theta'_{R2_{t2}}|X_{R2_{t1}}, X_{R2_{t2}})$$

From lemma 1 and lemma 2, we know that the upper-bounding strategy is applicable to emerging model ($ESTO$).

4.2 Precomputation and Pruning Mechanism

4.2.1 Precomputation for Region R

We recursively split the region into two sub-regions of the same size, starting from the biggest cuboid enclosed by two planes from time view, ending at the lowest resolution of the spatial-temporal grid. Fig. 3b shows the split approaches



(a) R (b) R (c) $\bar{R}$

Fig. 3. Precomputation of any given spatial-temporal region R and tiling of $\bar{R}$. Subfigure (a) shows a $8 \times 8 \times 8$ spatial-temporal grid; subfigure(b) shows: one of the cuboids from spatial precomputed set is split from temporal dimension and results in 15 smaller cuboids. Subfigure(c) is the radial method to tile $\bar{R}$.

for a sub-cuboid highlighted as blue from the temporal dimension in a $8 \times 8 \times 8$ grid (Fig. 3a). The likelihood of any given region can be upper-bounded by this pre-computed set via the tiling of LRT.

4.2.2 Precomputation for the Complement of Region R (i.e. $\bar{R}$)

By considering all of the intersection points, we connect each intersection point on the 3-dimensional grid with the eight corners of the grid. This produces eight diagonals, each of which creates one cuboid in the precomputed set. Since there are $O(n^4)$ intersection points, there are $O(n^4)$ cuboids in the precomputed set. After we get the precomputed set, for any given region $\bar{R}$, we use the radial and sandwich methods in LRT to get the upper-bounded likelihood value of $\bar{R}$. It involves of a total of twelves times tiling in 3-dimension view. Fig. 3c shows the tiling in radial way.

4.2.3 Computational Complexity

In the brute-force approach, there are a total of $O(n^6)$ regions that need to be searched. Our approach reduces the cost by precomputing two likelihood data set with size of $O(n^4)$. The likelihood of every region is upper-bounded and the real likelihood is calculated only for a number of regions. Furthermore, In our implementation, we have already ranked the top-k regions according to the likelihood ratio values. Therefore, the performance won't be affected no matter which significance testing method is applied.

The process of outlier detection is shown in Algorithm 1. The inputting parameters are: data grid (G), probability density function (f), maximum likelihood estimation function under different parameter space (MLE_0, MLE_1), likelihood function (L), number of top regions to be returned (K) and the significance level (α). In this process, step 1 and 2 perform precomputations; Step 5 to step 8 obtains the upper-bounded likelihood value of current cuboid for each iteration. During each iteration, the chi-squared distribution is applied to prune normal regions. Finally, it outputs top-k anomalous regions.

Algorithm 1. Top k spatio-temporal outlier detection

Input: a spatial-temporal grid G, f, MLE_0, MLE_1, L, k and α
Output: top-k anomalous spatio-temporal regions.

1: Precompute the $O(n^4)$ cuboids for upper-bounding any given cuboid R.
2: Precompute the $O(n^3)$ cuboids for upper-bounding any given cuboid $\bar{R}$
3: Let $\theta_0 = MLE_0(f(G))$.
4: **for** Each cuboid R in the grid **do**
5: Get the upper-bounded value for log L($\theta_R|X_R$)
6: Get the upper-bounded value for log L($\theta_{\bar{R}}|X_{\bar{R}}$)
7: Combine the results of step 3, 5, 6 to an upper bound for Λ_R
8: Check upper-bounded value of Λ_R from chi-square distribution
9: **if** The Λ_R is in the α level and less than the kth best **then**
10: Prune R
11: **else**
12: Compute real Λ_R;
13: **if** Λ_R is in the top k, **then**
14: Remember R
15: **end if**
16: **end if**
17: **end for**

5 Experiments, Results and Analysis

We report on experiments conducted where we have used Algorithm 1 to test for accuracy, pruning ability and performance. K was set to 1. In this section all experiments were carried out on synthetic data. In Section 5.3, we will demonstrate the usefulness of our approach on a real data set.

We tested four variants of the outlier detection:

(1) brute-force persistent spatio-temporal outliers (bpsto)
(2) brute-force emerging spatio-temporal outliers (besto)
(3) pruning-based persistent spatial-temporal outliers (ppsto)
(4) pruning-based emerging spatio-temporal outliers (pesto)

5.1 Evaluations on Synthetic Data

We generated data set on a grid size varying from $(4 \times 4 \times 4)$ to $(128 \times 16 \times 16)$. Fifty separate trials were carried out for each scenario (see below) and we measured three aspects: (a) pruning rate (b) accuracy, and (c) running time. The significance level was set at $\alpha = 0.05$.

5.1.1 Scenario I

The null hypothesis holds. The baseline b_c is generated relatively uniformly by a normal distribution ($\mu = 10^4, \sigma = 10^3$) and a fixed success rate p of 0.001. The number of successes k_c is generated from Po ($b_c p$). Results are shown in Table 1.

Table 1. Average Pruning Rate and Accuracy in Scenario I and II

(a) Scenario I

$Test$	$Pruning(\%)$	$Accuracy(\%)$
$4 \times 4 \times 4$	100	no false alarm
$8 \times 8 \times 8$	100	no false alarm
$16 \times 16 \times 16$	99.9	0.1 false alarm

(b) Scenario II

$Test$	$Pruning(\%)$	$Accuracy(\%)$
$4 \times 4 \times 4$	100	no false alarm
$8 \times 8 \times 8$	99.99	0.01 false alarm
$16 \times 16 \times 16$	100	no false alarm

Table 2. Average Pruning Rate in Scenario III

$Test$	$16 \times 16 \times 16$	$32 \times 16 \times 16$	$64 \times 16 \times 16$	$32 \times 32 \times 32$	$128 \times 16 \times 16$
ppsto (%)	95.27	97.35	97.64	97.47	96.74
pesto (%)	98.37	98.46	98.69	99.11	99.23

5.1.2 Scenario II

The null hypothesis holds. The only difference with scenario I is that the data in a random selected cuboid area with size of $(5 \times 4 \times 3)$ is generated by a normal distribution with different parameter setting $(\mu = 10^5, \sigma = 5 \times 10^3)$. Results are shown in Table 1.

Analysis: The results of Scenario I and II show that the we achieve a high pruning rate and almost no false alarm even when we perturb the distribution of one region. This is as expected and demonstrates that the algorithm is well calibrated. By a high pruning rate we mean that we can rule out the outliers by just checking the LRT upper bound derived from the tiling. If the upper bound value is less than the critical value then the true LRT value of the region cannot be anomalous.

5.1.3 Scenario III

The alternative hypothesis holds. It is similar to the null model except that the data of a randomly selected cuboid area of size $(5 \times 4 \times 3)$ is generated from a Poisson distribution with $p = 3, 6, 9, 18, 36$ for emerging case and $p = 3$ for persistent case. The data not within the cuboid area was also from a Poisson distribution with $p = 1$. Results are shown in Table 2.

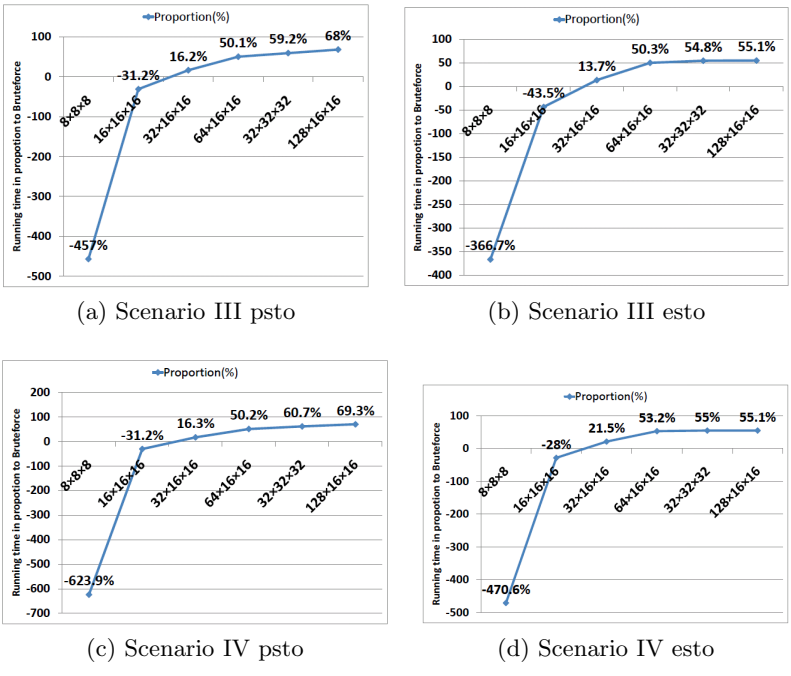
5.1.4 Scenario IV

The alternative hypothesis holds. It is similar to scenario III except that data of a randomly selected cuboid area of size $(5 \times 4 \times 3)$ was generated from a Poisson distribution with $p = 10, 50, 250, 1250, 6250$ for emerging case and $p = 10$ for persistent case. Results are shown in Table 3.

Analysis: For Scenario III and IV the anomalous regions were correctly identified while maintaining a high pruning rate. Also there were no regions declared as false positives.

Table 3. Average Pruning Rate in Scenario *IV*

Test	16 × 16 × 16	32 × 16 × 16	64 × 16 × 16	32 × 32 × 32	128 × 16 × 16
ppsto (%)	79.27	97.51	97.77	97.22	96.68
pesto (%)	95.57	97.40	96.78	94.70	95.23



(a) Scenario III psto (b) Scenario III esto

(c) Scenario IV psto (d) Scenario IV esto

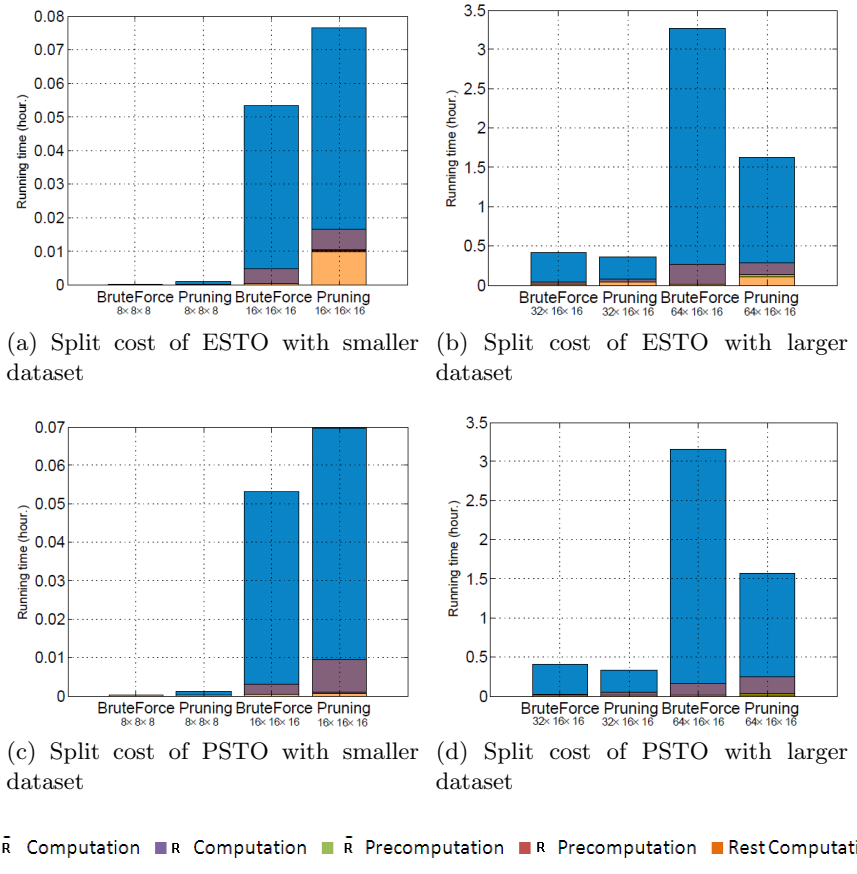
Fig. 4. The proportion of running time of pruning vs. brute-force approach. It shows that outlier pruning searching is significantly improved when the dataset size starts from 32 × 16 × 16 in these four different scenarios.

5.1.5 Running Time

We analyze the running time with and without pruning for Scenario III and IV. Figure 4 shows that as the size of the spatial and temporal region increase, the effect of pruning becomes prominent. For the largest data tested, the pruning mechanism resulted in a savings of nearly 50% compared to the brute-force approach. We have also calculated the cost of a single likelihood calculation as the dimension of the grid size increases. For the 8 × 8 × 8 data set, the cost of the likelihood calculation using the brute-force approach is 0.01ms while with pruning it increases to 0.08ms. However, for the larger data sets (e.g., $128 \times 16 \times 16$) the cost of a single likelihood calculation goes does from 0.30ms for the brute-force approach to around 0.16ms with pruning. Another observation is that the cost of the single likelihood calculation is nearly similar for data sets of the same size but different dimensions, for example $128 \times 16 \times 16$ and $32 \times 32 \times 32$.

We have also analyzed and compared the running of the different components both for the brute-force and pruning approaches. The results are shown in Figure 5. The brute-force approach has the following components:

(1) The cost of the likelihood calculation for each region R (R Computation).
(2) The cost of the likelihood calculation for the complement of each region R, denoted as $\bar{R}$ ($\bar{R}$ Computation).



(a) Split cost of ESTO with smaller dataset

(b) Split cost of ESTO with larger dataset

(c) Split cost of PSTO with smaller dataset

(d) Split cost of PSTO with larger dataset

Fig. 5. The running time of comparable parts of brute-force vs. pruning approach in scenario III. It shows that the pruning searching is faster with the larger dataset. Although the tiling of every $\bar{R}$ takes longest time in pruning searching, the cost is small compared to the likelihood calculation of every $\bar{R}$ in brute force searching.

The pruning approach is more complex and involves the following components:

(1) The cost of computing the likelihood for each element of the tiling set T_R. This will be used to upper bound the likelihood value for an arbitrary spatio-temporal region. (R precomputation)

(2) The cost of computing the likelihood for each element of the tiling set $T_{\bar{R}}$ ($\bar{R}$ precomputation).
(3) The cost of upper-bounding the likelihood of R. This involves first expressing R as a union of subregions and then each subregion as a union of tiles from T_R.
(4) The cost of upper-bounding the likelihood of $\bar{R}$ ($\bar{R}$ Computation). This involves first expressing $\bar{R}$ as union of subregions and then each subregion as a union of tiles from $T_{\bar{R}}$. Each $\bar{R}$ region can be expressed as a union of six subregions and there are two types of tiling methods: sandwich and radial. We calculate the likelihood value using both tiling methods and then select the tightest upper-bound. As is clear from Figure 5, this is the most expensive part of the calculation. However, as the data set size increases, the overheads of the tiling give way to its more efficient reuse resulting in considerable savings.

5.2 Case Study: Beijing Taxi GPS Data

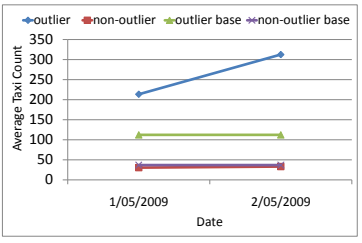
We illustrate the use of the Pesto method on a real data set [4], [8], [3]. The data set consists of three months of GPS trajectories collected from 33,000 taxis in Beijing between 01/03/2009 and 31/05/2009. We search for the most anomalous region and the time period and then provide a possible explanation for the anomaly.

Case I: All (8×8) grid were tested between $9:00:00am$ and $10:00:00am$ for sixteen days. We choose 20 days of data to calculate the baseline probabilities.

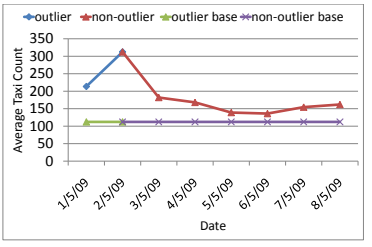
Result I: The period from 01/05/2009 to 02/05/2009 emerged as a top outlier at the position of $(0, 1)$ and $(1, 1)$ on the grid. This period corresponds to the Labor day public holiday ("Golden Week"). Usually the holiday duration is seven days (from May 1st to May 7th) , but starting from 2009, the holiday period was truncated between May 1st and May 3rd. To celebrate the holidays it appears that many people visited Happy Valley, the biggest amusement park in Beijing. The 3rd International Fashion festival was also held in that location. Our results coincide with the fact that taxis enjoy good business on public holidays and there is usually an increase in the number of taxis near tourist spots. The results are shown in Fig. 6a, 6b, 7a. We can see that the number of taxis increased from 1st May to 2nd May and then decreased from 3rd of May onwards.

Case II: All (8×8) grids were tested. The temporal unit is from $3:15:00pm$ to $4:30:00pm$ every day for 8 days. Twelve days of data was used to calculate the baseline.

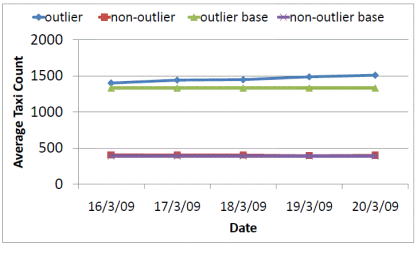
Result II: The region highlighted as blue on the map was detected as an emerging outlier from 16/03/2009 to 20/03/2009. It is one of the city express road called Tonghuihe North Road. From 01/03/2009 to 13/0/20093, the 11th National People's Congress (i.e. NPC) was held in Beijing, which is the annual meeting of the highest legislative body of the People's Republic of China. Nearly



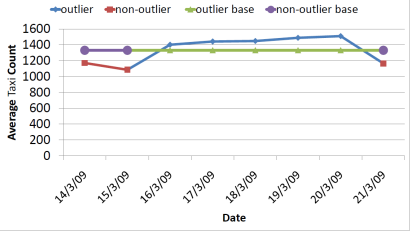
(a) The average taxi counts within outlier regions vs. non-outlier regions from 01/05/2009 to 02/05/2009



(b) The average taxi counts within outlier regions from 01/05/2009 to 08/05/2009



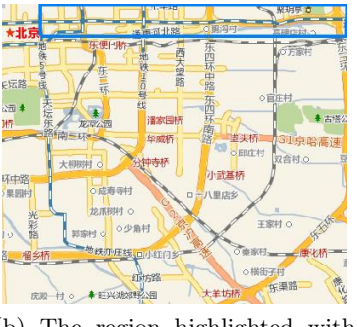
(c) The average taxi counts within outlier regions vs. non-outlier regions from 16/03/2009 to 20/03/2009



(d) The average taxi counts within outlier regions from 14/03/2009 to 21/03/2009

Fig. 6. Comparison of outlying and non-outlying regions in $8 \times 8 \times 8$ grid. It shows: (a) the average taxi counts within outlier regions is non-decreasing compared to non-outlier regions which share the same emerging period with outlier. (b) the average taxi counts within outlier regions throughout the emerging period is non-decreasing compared to the outlier regions for the rest of the period.

(a) The region highlighted with blue borders on the map is the outlier region of Case I. The icon shows the exact location of Happy Valley.



(b) The region highlighted with blue borders is the outlier of Case II. It is the city express road of Beijing. (i.e. Tonghuihe North Road)

Fig. 7. Outlier Locations from our two case studies on Beijing Map

3000 deputies from all over China attended the Congress. During this period, the traffic authorities in Beijing imposed temporary restriction measures on vehicles to control traffic flow. Most people choose to take bus or subway instead of driving or taking taxi to commute to work. The number of taxi traveling on Tonghuihe North road increased until most of the deputies left Beijing. The results are shown in Fig. 6c, 6d, 7b.

6 Related Work

Among the various methods for discovering outlier, the spatial and space-time scan statistic, introduced by Kulldorff [9,10,11,14], has been the most widely adopted. However, it is originally designed for poisson and bernoulli data. Later on, the different variations of ordinal, exponential and normal models are proposed [12,6,7,13]. They have been implemented in the software (SaTScan)[1]. In the space-time scan statistic of Kulldorff, the key parameter is assumed as consistent over time. The technique simply applies time as one more dimension. Niell et al. [5] points out the distinct feature of time aspect and proposes a modified test statistic to detect localized and globalized emerging cluster . Tango et al. [16] also proposes a space-time scan statistic based on negative binomial model by taking into account the possibility of nonnegligible time-to-time variation of poisson mean. Wu et al. [15] proposes a generic framework called LRT for any underlying statistics model. It uses the classic likelihood ratio test (LRT) statistic as a scoring function to evaluate the "anomalousness" of a given spatial region with respect to the rest of the spatial region. Moreover a generic pruning strategy was proposed that can greatly reduce the number of likelihood ratio tests. However, it is used for spatial anomaly detection without considering the temporal property. Wei et al. [3] propose an approach to discover casual relationships among spatio-temporal outliers. Here, we only focus on detecting spatial-temporal outliers.

7 Conclusions

In this paper, we proposed an efficient pattern mining approach catered for spatio-temporal traffic data, which is able to detect "persistent outliers" and "emerging outliers". We also derived a upper-bounding strategy for the two statistic models. Experiments show that the performance of computational time is greatly improved when the dataset size is very large, and we can still find the correct outliers. In our case studies, our model is able to detect regions with emerging number of taxis that can be validated by known major traffic events.

References

1. http://www.SatScan.org
2. Barlow, R.E., Scheuer, E.M.: Reliability Growth During A Development Testing Program. Technometrics (1966)

3. Liu, W., Zheng, Y., Chawla, S., Yuan, J., Xie, X.: Discovering spatio-temporal causal interactions in traffic data streams. In: 17th SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2011, pp. 1010–1018 (2011)
4. Chen, Z., Shen, H.T., Zhou, X., Zheng, Y., Xie, X.: Searching trajectories by locations: an efficiency study. In: Proceedings of the 29th ACM SIGMOD International Conference on Management of Data (SIGMOD 2010), pp. 255–266 (2010)
5. Neill, D.B., Moore, A.W., Sabhnani, M., Daniel, K.: Detection of emerging space-time clusters. In: Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (KDD 2005), pp. 218–227 (2005)
6. Jung, I., Kulldorff, M., Klassen, A.C.: A spatial scan statistic for ordinal data. Stat Med, 1594–1607 (2007)
7. Jung, I., Kulldorff, M., Richard, O.J.: A spatial scan statistic for multinomial data. Stat Med, 1910–1918 (2010)
8. Yuan, J., Zheng, Y., Zhang, C.Y., Xie, W.L., Xie, X., Sun, G.Z., Huang, Y.: T-drive: Driving directions based on taxi trajectories. In: Proceedings of the 18th ACM SIGSPATIAL Conference on Advances in Geographical Information Systems, pp. 99–108
9. Kulldorff, M.: A spatial scan statistic. Comm. in stat. Theory and Methods, 1481–1496 (1997)
10. Kulldorff, M.: Spatial scan statistics: models, calculations, and applications. In: Glaz, J., Balakrishnan, N. (eds.) Scan Statistics and Applications. Birkhauser (1999)
11. Kulldorff, M., Nagarwalla, N.: Spatial disease clusters: detection and inference. Statistics in Medicine, 799–810 (1995)
12. Huang, L., Kulldorff, M., Gregorio, D.: A Spatial Scan Statistic for Survival Data. International Biometrics Society, 109–118 (2007)
13. Huang, L., Tiwari, R., Kulldorff, M., Zou, J., Feuer, E.: Weighted normal spatial scan statistic for heterogenous population data. American Statistical Association (2009)
14. Kulldorff, M., Athas, W., Feuer, E., Miller, B., Key, C.: Evaluating cluster alarms: a space-time scan statistic and cluster alarms in los alamos. American Journal of Public Health 88(9), 1377–1380 (1998)
15. Wu, M., Song, X., Jermaine, C., Ranka, S., Gums, J.: A LRT Framework for Fast Spatial Anomlay Detection. In: Proceedings of the 15th ACM SIGKDD international Conference on Knowledge Discovery and Data Mining (KDD 2009), pp. 887–896 (2009)
16. Tango, T., Takahashi, K., Kohriyama, K.: A Space–Time Scan Statistic for Detecting Emerging Outbreaks. International Biometrics Society, 106–115 (2010)

Ontology Guided Data Linkage Framework for Discovering Meaningful Data Facts

Mohammed Gollapalli[1], Xue Li[2], Ian Wood[3], and Guido Governatori[4]

[1,2] School of Information Technology & Electricial Engineering, University of Queensland, Brisbane, Australia
{mohammed,xueli}@itee.uq.edu.au
[3] School of Mathematics & Physics, University of Queensland, Brisbane, Australia
i.wood1@uq.edu.au
[4] National ICT Australia (NICTA), Queensland Research Laboratory, Brisbane, Australia
guido.governatori@nicta.com.au

Abstract. Making sensible queries on databases collected from different organizations presents a challenging task for linking semantic equivalent data facts. Current techniques primarily focused on performing pair-wise attribute matching and paid little attention towards discovering probabilistic structural dependencies by exploiting the ontological domain knowledge of tables, attributes and tuples to construct hierarchical cluster mapping trees. In this paper, we present Ontology Guided Data Linkage (OGDL) framework for self-organizing heterogeneous data sources into homogeneous ontological clusters through multi-faceted classification. Through the evaluation on real-world data, we demonstrate the robustness and accuracy of our system.

Keywords: Data linkage, ontology matching, clustering, table attributes.

1 Introduction

The ability to extract meaningful data facts from multiple sources is a significant problem of current data linkage techniques. When attempting to perform data linkage where there are inconsistencies in the data, domain knowledge is necessary. For example, the database from an organization such as The World Bank [10] knows nothing of a database from another organization, for example the US Federal Government Archives [11]. Without common domain knowledge, extracting meaningful data into a usable form becomes a tedious process. Additionally, a similar problem exists for database migrations and other data intensive tasks that involve disparate databases without common schemas. Furthermore, performing data linkage on heterogeneous databases requires discovering all possible primary and foreign key relationships that might exist between different attribute pairs on a global spectrum.

In conducting our research, we investigated on real world data collected from a variety of sources [10 to 16], anticipating that the lack of a common domain would

J. Tang et al. (Eds.): ADMA 2011, Part II, LNAI 7121, pp. 252–265, 2011.

indicate the shortage of multi-domain experimentation in similar research. Examination of real-world data reveals that the relational database schemas that are invariant in time hold valuable information (tables, fields and tuples) and can aid in identifying semantically similar objects with which we can derive hierarchical relationships up to the instance level in the dataspace approach [19]. Unfortunately, schema-based techniques lack the ability to allow computational linkage using domain information and are highly expensive when dealing with unrelated databases.

A key step in integrating databases is to identify the semantic correspondences among the attribute pairs [1, 6]. Therefore, we primarily focused our research in finding semantic data coordination using ontology matching principles. Ontology matching is the process of finding correspondences between semantically related entities of different ontologies [1]. Eusenat *et al* in [1] has shown that such methods can be highly effective when combined with other methodologies. In this paper we will consider a new type of data linkage problem, namely, the problem of exploiting hidden relationships between tables, attributes and tuples - based on the knowledge discovery at different levels of data abstraction such as ontology level, schema level and instance level where semantic information is used as data abstraction principles to perform data linkage. Constructing a novel system that embodies this approach raises a number of challenges. Our solution to handle these challenges will introduce a variety of approaches through extending existing techniques and proposing a multi-strategy ontology guided data linkage framework.

Performing attribute pair-wise matching between different data source tables is suitable when dealing with small databases. However, this could become a costly approach when applied on large volumes of data. The computational expense derived from performing such a pair-wise attribute matching is perhaps the main drawback from existing data linkage methods and is what restricts its accuracy. In order to reduce the number of pair-wise comparisons, we employ "ontological clustering" technique which represents data into much smaller datasets by modeling huge amounts of information given as input into easily analyzable ontological models.

During the course of our research, we also found that the classification of data into a single order doesn't give the flexibility in defining variable semantic mappings. For instance, organizations can maintain different rules and standards in storing their business data. To make things worse, databases might be poorly designed without any data models. Furthermore, platform independent databases have also been emerging in recent years in order to attract the global market. Hence, the semantic flow of data and their relationships might not be in a fixed direction when dealing with variable data sources. In order to increase the chances of discovering correlations between clusters, we employ "multi-faceted" mapping strategy. The overarching objective is to represent tables, attributes and tuples ontological domain information in multiple facets (arrangements), instead of any predetermined order, in an anticipation of capturing the flow of meaningful semantic data and to construct self-expanding

hierarchical semantic trees crucial for data linkage. We described methods for constructing three very different kinds of representations: sequential facet, parallel facet and mixed facet. Sequential facet aims to classify data based on the ontological findings of table level clusters, followed by attribute level clusters and then tuple level clusters. Parallel facet classifies data based on equal chance of finding pairs within table level clusters; within attribute level clusters and within tuple level clusters without prioritizing any sequence order. Mixed facet classifies data through combined cross references of table level, attribute level and tuple level clusters. The obtained results can be easily narrowed down in discovering candidate keys, primary keys, foreign keys, and partially related keys the results of which can be integrated with IBM or Microsoft's QBE (Query-by-Example) tools for making sensible queries in order to discover meaningful data facts.

The first contribution of this paper is the handling of data uncertainties. The data collected from different data sources are often noisy. In order to perform a successful data linkage, data needs to be organized in a way that different sets of data can be well- represented. The data uncertainty process introduces a series of steps to organize noisy datasets into a uniform representation before the data linkage process starts. The second contribution we have made is the introduction of OGDL framework. OGDL creates variable clusters within each sample set based on the ontological essence and self-expands through multi-faceted cluster mapping strategy applied on a global spectrum. The framework results can be further narrowed down in creating schema structures the results of which can be integrated in data mining tools for knowledge discovery. Finally, the third contribution is the extensive evaluation of accuracy, performance and scalability tests carried out in order to evaluate OGDL framework against real-world databases. The experimental results indicate that our framework discovers precise data relationships by an order of magnitude.

This paper is organized as follows: In section 2 we discuss previous work related to this research; in section 3 we propose our new framework; in section 4 we present our experimental results; and in section 5 we discuss our conclusions and recommendations for future work.

2 Related Work

Michel Gagnon [2] proposed local to global ontology mapping approach for integration of data sources. While this technique helps in getting a global ontology picture, it doesn't make use of schema mapping strengths and doesn't serve the need to understand the tuples. [3, 4] have proposed global schema mapping as a resolution for data linkage. However, schema mapping by itself is not sufficient [1] and cannot perform an ideal solution to understand the semantic structures of unrelated databases.

CORDS [5] is a substantial contribution to query-based approaches; CORDS discovers correlations and soft functional dependencies through an automated process using column pairs. CORDS create column groups through a series of

processes that include enumeration of candidate pairs, elimination of unlikely candidates, and statistical analysis to identify correlations. Unfortunately, CORDS is built primarily on relational databases. Secondly, CORDS performs pairwise attribute matching instead of structural level approach which is highly expensive when applied on large volumes of unrelated databases and doesn't serve discovering semantic mappings.

iDisc [6] creates database representations through a multi-process learning technique. Base clusters are used to uncover topical clusters which are then aggregated through meta-clustering. The advantage of iDisc framework is that it supports extending additional clusters and representations. However, the downside of iDisc approach is that it doesn't support reasoning based mapping which could be seen as a cumbersome approach. Furthermore, it doesn't consider building ontological structure mapping trees.

3 Ontology Guided Data Linkage (OGDL) Framework

Ontology Guided Data Linkage (OGDL) represents a new framework allowing managing large volumes of data sources by extracting their ontology and consequently by accounting for a "summarised version" of the source data. This aim is achieved by means of globally describing ontological clusters of heterogeneous raw data as a means to perform data linkage. We propose a "knowledge based" multi-faceted cluster mapping technique which aims at extracting possible relationships between related data clusters on a structure level.

When integrating large volumes of data our framework intends to create a feasible method for discovering related data as part of bottom-up system handled process and allowing for user-friendly queries on a top-down information extraction procedure. The main intuition behind our methodology is that end-users, performing semantic queries will not have knowledge of meaningful data relationships unless relationships can be established and related information is presented. Fig. 1 shows the architecture of our framework. When the user creates a query or intends to visualize relationships found through our technique on a graph, the results delivered include not only the data requested by the user but also directly related data.

Our approach is user-friendly, and is designed for use primarily in business intelligence and data discovery purposes. Our application can be used by data managers, researchers, or analysts, as it doesn't require data structure or complex query knowledge. The series of steps performed as part of our OGDL framework are aimed at discovering data linkage across big-size databases with minimal user involvement. In other words, our proposed framework in this paper directly addresses high computational overheads through multi-layer strategy which highly reduces the amount of data considered for comparison at subsequent stages and self-expands in constructing ontology guided data linkage structures.

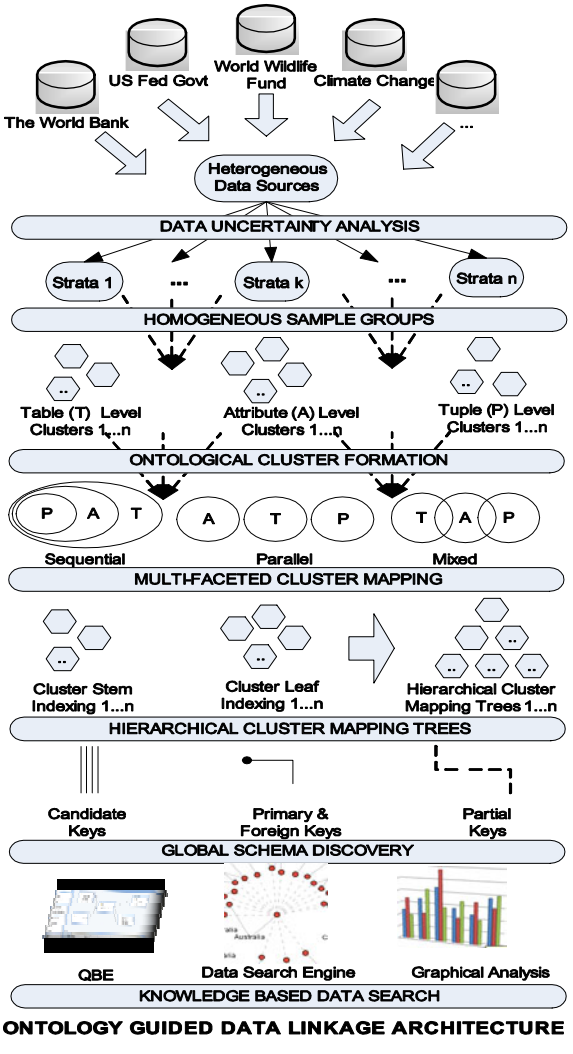


Fig. 1. The general architecture of Ontology Guided Data Linkage (OGDL) framework. The framework takes different datasets as input and performs data uncertainties analysis process which helps cleaning and organising data into homogeneous strata groups. The strata samples are used to form different levels of clusters. The framework then performs cluster stem-and-leaf joins through multi-faceted cluster mapping technique the results of which are further analysed to construct hierarchical cluster mapping trees. The ontological structures are summarised to form data linkage; discovering candidate, primary, partial, and foreign key relationships. The final results obtained can be integrated in knowledge based data analysis tools in which sensible queries can be made to discover meaningful data facts.

3.1 Data Uncertainties Analysis

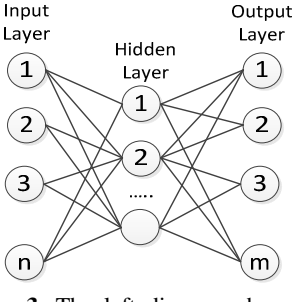
When dealing with large volumes of data (numeric, categorical, string based etc.) obtained from different sources, we are highly exposed to different types of "data

uncertainties" which could be one of the biggest obstacles to perform successful data linkage. In an effort to improve the quality of data in such cases we employ series of preliminary steps based on the principles as shown in fig. 2 ensuring best optimal results. An important advantage of "data uncertainties" process is that this is a one off system handle process aimed at cleaning, classifying and organizing observational data. It has to be noted that these steps are aimed at approximating the quality of data and doesn't guarantee overcoming all of the data uncertainties.

Fig. 2. These are three steps involved in the data uncertainties analysis process. The ontological classification is used to extend Neural-Networks technique to categorize attributes into homogeneous clusters based on the ontological matching weights. The stratified sampling is applied in each of the ontological classification group. Outlier detection step cleans the sample collection with any extreme values.

Ontological Classification. Ontological Classification is the process of categorizing the given data into predefined classes based on similarities of their ontologies. Neural Networks [1] attempt to produce intelligent behaviour by clustering attributes into categories using attribute properties. In our research, we have introduced an extended version of Neural Networks [1] through the process of ranking ontological characteristics (see Fig. 3). We employed Spearman's Rank correlation [18] to determine relationship strengths based on ranking prediction weights given by different participating judges. The *Judges* in our case are the pattern matching agents for recognizing say "m" number of pattern clusters for the given "n" number of features. The extended Neural Networks takes advantage of ontological measurement variables from the input layer and rank them using the provided classified definitions in the hidden layer and finally clusters ontologically matching attributes into homogeneous groups in the output layer. Assume that group of n entity values are arranged in the order of merit in possession of characteristics $c_1, c_2, \ldots c_n$. Let $\rho(c_i, c_j)$ be the rank defined between two characteristics c_i and c_j. The ontological classification towards classifying attributes is determined as shown in (1) where the pair of attributes having the nearest common attribute features is the maximum $\rho\,(c_i, c_j)$.

$$\rho\,(c_i, c_j) = 1 - \frac{\sum_{i=1}^{n} d_i^2}{2n\sigma_i^2} = 1 - \frac{6\,\sum_{i=1}^{n} d_i^2}{n\,(n^2-1)} |\sum_i d_i^2 = \sum_i [(x_i - \bar{x}) - (y_i - \bar{y})]^2 \quad (1)$$



Classification C1
Survey_Date
StartDate
EndDate
DOB
Election Date
Payment Date
BirthDate
DeathDate
.....

Classification C2
Service_Code
Time_Code
Editby
COUNTRYCODE
Region
Indicator
.....

Classification Cn
Foot_Note
Comment
MailingAddress
Charity
Description
IndicatorNames
Notes
.....

Fig. 3. The left diagram shows the Neural Networks design and the right diagram shows example of attribute clusters formed in The World Bank database [10].

Stratified Sampling. Sampling is the statistical practice concerned with the selection of an unbiased or random subset of individual observations within a population (data) of individuals [18]. In the case where there are an unmanageable number of records, the computational requirements may become unacceptably high and thus sampling methodologies will need to be considered. In order to yield knowledge about relatively large volumes of data and to make predictions based on statistical inference we employ stratified sampling process. In a stratified random sampling, the actual data is divided into homogenous groups, called as strata's. Each strata differs from the other strata but each of the strata is homogenous within itself [18]. The advantage of stratified sampling is that it provides greater precision often requiring smaller samples. We take advantage of results discovered from the ontological classification process in order to create these strata's. The below equation shows how the variable is calculated for a stratum i taken as a reference for n total number of strata's. The number of samples in each stratum is a function $f(i)$ of its standard deviation st_{dev} i.e. homogeneity: the more homogeneous is the stratum (small standard deviation), the lower number of samples.

$$f(i) = n \cdot \frac{st_{dev(i)}}{\mathbf{Max}\{st_{dev}\}} \quad No.\ Samples(i) = \begin{cases} 0.5 \cdot n, & f(i) > 0.5 \cdot n \\ f(i), & 0.5 \cdot n \geq f(i) \geq 0.1 \cdot n \\ 0.1 \cdot n, & f(i) < 0.1 \cdot n \end{cases} \tag{2}$$

Outlier Detection. Outliers are basically 'erroneous' observations that are not in alignment with the overall population (data). These erroneous values often pose serious hurdles for obtaining stable estimates and making correct predictions. In order to reduce the number of computationally-expensive entries, we used Box Plot statistical approach [18] to detect and overcome all the erroneous observations. The box-plot outliers are extreme values located outside the lower and upper quartile (the normal distribution curve). An important advantage of using Box Plot is that it allows calculating all extreme values based on the specified percentage threshold from the median value. Another advantage of using Box Plot is that all the results can be graphically depicted. For non-scaled data, outlier process considers only string lengths, and not the actual data itself. This restriction vastly reduces the complexity of the algorithm and scales well to the large number of attribute sizes found in the real-world databases.

3.2 Multi-layer Cluster Formation

We define ontological clustering as the process of globally describing entries at different levels into much lower number of elements in order to reduce the computational expense required during the actual data linkage. The resulting cluster sets represent the most effective way in achieving probabilistic matching with high-accuracy at low enough computational cost. Cluster formation requires comparison of various sets of possible semantic pairs. To reduce the number of comparisons required

in determining entity similarities, we propose extended version of *tri-grams* [1] providing the best balance of accuracy and computational requirements. The extended tri-grams are formed through the generation and transformation of each gram into its constituent scale based "hash code". The hash codes generated for similar substrings are similar [20] which are used to calculate the correlation between entity pairs. The extended tri-grams are calculated as $\sqrt{\sum_{\forall x} |f(hc[x])_\alpha - f(hc[x])_\beta|}$ where $f(x)_\alpha$ and $f(x)_\beta$ are the number of occurrences of "hash codes" of substring x in entities α and β.

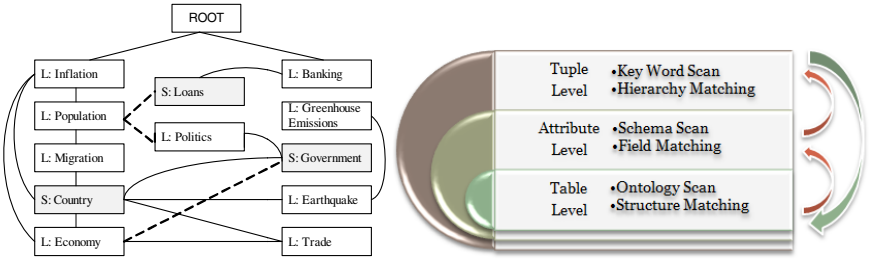


Fig. 4. The left diagram shows example of tuple level cluster with stems and leaf pairsformed in the World Bank Statistical Indicators data [10] and the right diagram shows the repetition of cluster formation process at different levels.

In order to discover the flow of semantic information in multiple dimensions, the framework performs clustering process at different levels (see Fig.4) to capture different sets of ontological pairs. Clusters $\tau(\partial_n)$ are formed at the table level τ, in which case entities are collection of table names along with their ontological resource to represent pairing tables in a cluster. Consider a set of tables $\tau_1, \tau_2, \ldots \tau_n$ where each table is associated with an ontological structure such as $\tau_i \rightarrow \bigcup \tau(\partial)_i$, $\tau_j \rightarrow \bigcup \tau(\partial)_j \ldots$ then we define tables are similar if their ontologies are found similar $\forall(\tau_1, \tau_2, \ldots \tau_n) \mid \tau_i \approx \tau_j \; if \; (\bigcup \tau(\partial)_i \approx \bigcup \tau(\partial)_j)$. The results obtained from table level clusters are used to perform structural level matching. Similar technique is employed at attribute level $\bar{a}(\partial_n)$ in which case entities are collection of attribute names along with their ontological resource to represent pairing attributes in a cluster. Consider a set of attributes $\bar{a}_1, \bar{a}_2, \ldots \bar{a}_n$ where each attribute is associated with an ontological structure such as $\bar{a}_i \rightarrow \bigcup \bar{a}(\partial)_i$, $\bar{a}_j \rightarrow \bigcup \bar{a}(\partial)_j \ldots$ then we define attributes are similar if their clusters are found similar $\forall(\bar{a}_1, \bar{a}_2, \ldots \bar{a}_n) \mid \bar{a}_i \approx \bar{a}_j \; if \; (\bigcup \bar{a}(\partial)_i \approx \bigcup \bar{a}(\partial)_j)$. The results obtained from attribute level clusters are used to perform field matches during the schema scans. Clusters are formed at the tuple level, in which case entities are collection of strata's formed in the data uncertainties stage. Consider a set of strata's $\varphi_1, \varphi_2, \ldots \varphi_n$ where each stratahas associated sample entities such as $\varphi_i \rightarrow \bigcup \varphi(\partial)_i$, $\varphi_j \rightarrow \bigcup \varphi(\partial)_j \ldots$ then we define tuples are similar if their strata's are found similar $\forall(\varphi_1, \varphi_2, \ldots \varphi_n) \mid \varphi_i \approx \varphi_j \; if \; (\bigcup \varphi(\partial)_i \approx \bigcup \varphi(\partial)_j)$. Tuple level clusters are used to perform key word scans during hierarchical data matching.

Algorithm 1. Ontological Cluster Formation

Input: S_i, Set of entities (table sets, attribute sets or tuple sets) for $i = 1,2,3 \ldots n$, $\{S_1, S_2, S_3, \ldots . S_i\}$
Output: C_n: $(\rho_n \rightarrow \gamma_n)$, Clusters with stem-and-leaf mappings for $n = 1,2,3 \ldots n$

1. **Let** $C_n := \{\emptyset\}$ Set of data clusters for $n = 1,2,3 \ldots n$
2. **for all** S_i **do**
3. **for all** $(\gamma, \partial) \in S_i$ **do** *// collect linguistic resource ∂ for entity element γ*
4. *Extri-grams* $(\gamma, \partial) :=$ *Extri-grams*$(\gamma, \partial) \cup \{S_i\}$ *// generate extended tri-grams*
5. **for all** $A_i, A_{i+1} \in$ *Extri-grams* $(\gamma, \partial \rightarrow S_i)$, $A_i \neq \emptyset, A_{i+1} \neq \emptyset$ **do**
6. **if:** *correlation* $\Delta(A_i, A_{i+1}) \leq \sigma(A_i, A_{i+1})$ *// calculate and verify density*
7. **If:** $C_i(A_i, A_{i+1}) \in C_n$ with $C_i \neq \emptyset$ *// verify if cluster already exists*
8. $C_n(A_i, A_{i+1}) := C_i(A_i, A_{i+1}) \cup \{C_n\}$ *//merge with existing cluster*
9. **else:**
10. $C_n += C_i(A_i, A_{i+1})$ *// form a new cluster*
11. *// create ontological cluster mapping*
12. **for all** $C_i(x, y) \in C_n, where\ i = 1,2,3 \ldots n$ **do**
13. *stem*$(\rho_i) := \{\emptyset\}$, *leaf* $(\rho_i \rightarrow \gamma_n, c_i) := \{\emptyset\}$
14. ***for all*** $\gamma_n \in \{\rho_i, [x, y]\}$ **do**
15. C_n: $(\rho_n \rightarrow \gamma_n) += C_i(\rho_n \rightarrow \gamma_n)$ *//map stems and leaves*
16. **return** C_n

Fig. 5. The algorithm for creating different levels of ontological clusters. The algorithm takes different sets of data as input and discovers data ontology through extended tri-grams the results of which are grouped into each cluster. The resulting clusters are further analyzed in order to form stem-and-leaf mappings within each cluster.

The ontological clustering algorithm is defined in Fig.5.For each of the input (table level, attribute level, and tuple level entities), linguistic resource (step 3) is associated and extended tri-grams are generated for similarity matching (step 4). Comparisons are made across various data inputs through the transformed tri-grams and correlations of variable sets Δ are determined (step 5-6) through the accepted threshold σ. The results are saved into a new or existing cluster (step 7-10) holding homogeneous cluster sets. In order to handle unmanageable number of cluster entities through this process, we employ stem-and-leaf mappings (step 12-15). Stem-and-Leaf mapping is the process of marking relationships between entities within each cluster. Stems are the representatives of a cluster which are formed as a result of defining unique ontological definition based on the highest correlation ranking whereas leaves are the representation of semantic concepts within the classified cluster. Fig.4 shows example of stem-and-leaf mapping formed in the World Bank Database [10]. The stems $\{Country, Loans, and\ Government\}$ are associated with their corresponding leaf structures. One of the primary tasks of this process is that it reduces the unmanageable number of cluster pairs and maps direct and indirect relationships which can be used as a basis for matching clusters which is detailed in the next section.

3.3 Multi-faceted Cluster-Mapping

Discovering different sets of ontological clusters are not sufficient unless hierarchical relationships are established between clusters the results of which can be queried to retrieve meaningful data facts. We introduce multi-faceted cluster mapping strategy in

order to capture the structural relationships between different ontological clusters in different arrangements, an advantage when compared over existing techniques. Different level of clusters discovered in the previous section (section 3.2) are arranged into sequential, parallel and mixed facets as shown in Fig. 6 and relative mappings with their relationship strengths are determined and preserved in ARFF format [17].

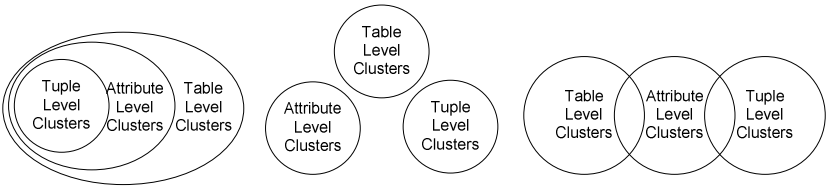


Fig. 6. Diagram showing multi-faceted arrangements in mapping clusters. The left diagram shows sequential cluster mapping. The middle diagram shows parallel cluster mapping and the right diagram shows mixed cluster mapping.

Sequential facet (arrangement) capture relationships between clusters based on the ontological findings of table level clusters $\tau(C_1, C_2, C_3, \dots C_n)$ followed by attribute level clusters $\bar{a}(C_1, C_2, C_3, \dots C_n)$ and then tuple level clusters $\varphi(C_1, C_2, C_3, \dots C_n)$ which is defined as $\forall\tau\,(\tau \in \bar{a} \cup \varphi \leftrightarrow \tau \in \varphi \cup \bar{a})$. Parallel facet attempts to find independent matches within table level, attribute level and tuple level clusters ignoring the order of precedence. Fig.7 shows example of sequential and parallel facet. In contrast to sequential and parallel, mixed facet discovers relationships by cross-referencing all the sets of clusters formed. Our experiments reveal that mixed facet is crucial in pairing time-series based data. For example, in the World Bank [10] database; the tuple values from Health Nutrition and Population Stats table had {1961M01, 1962M2, 1963M3,…} which required matching with the Global Economic Monitor Terms of Trade table attribute names {1960, 1961, 1962,…}. In order to rank the correlation between ontological clusters, we used Intra-class correlation (ICC)

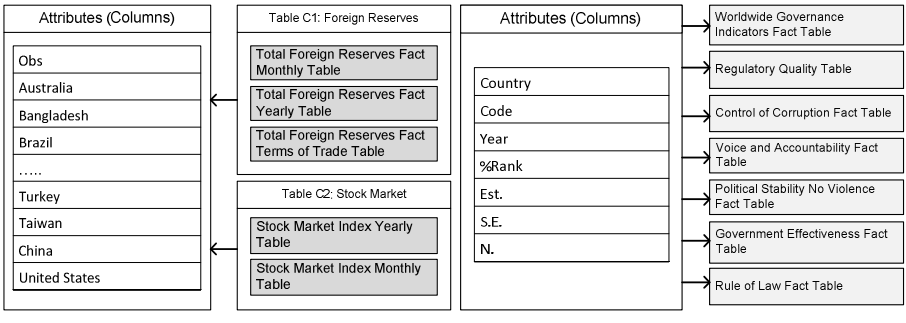


Fig. 7. Example of sequential (left) and parallel (right) facet matches from The World Bank database [10]. In a sequential facet, if table level clusters match we assume attributes match. However, in a parallel facet, if table level clusters don't match, the attributes can still match as shown in the diagram.

technique [18]. Let $\vec{\gamma}$ and $\overrightarrow{\delta}$ be the ontological vectors of two cluster entities γ and δ of cluster space ϑ, the cluster relationships are determined through the correlation calculation of Ψ as shown below:

$$\Psi(\gamma,\delta) = \frac{\sum_{\sigma\in|\vartheta|}\overrightarrow{\gamma_\sigma} \times \overrightarrow{\delta_\sigma}}{\sqrt{\sum_{\sigma\in|\vartheta|}\overrightarrow{\gamma^2{}_\sigma} \times \sum_{i\in|\vartheta|}\overrightarrow{\delta^2{}_\sigma}}} \tag{3}$$

3.4 Generating Global Schema

The proposed framework takes advantage of relationship densities formed in the previous step (section 3.3) in order to create global schema the results of which are used to discover meaningful data facts. Creating schema structure requires identifying candidate keys, primary keys and foreign key relationships. OGDL framework performs incremental pair-wise comparison between attributes having semantic cluster mappings (see section 3.3) and determines possible candidate keys. Suppose we have A_1, A_2, A_3, and A_4 attributes, the candidate keys are discovered by applying incremental pair wise cluster comparisons matching up to counts of Ω-item set (where Ω is the parameter specified by the user) $\{A_1, A_2\}, \{A_1, A_3\}, \ldots \{A_3, A_4\}, \ldots \{A_2, A_3 A_4\}$ as shown in Fig.8. The potential primary/foreign key relationships are formed by means of accounting the density of relationships between cluster pairs. For this, we relied on the multiplicity property which is defined as the maximum number of child table cluster entities⋃ A able to link with the parent table cluster entities ⋃ Bin order to form parent-child relationship.

$$if\ A \subseteq B, then \bigcup A \subseteq \bigcup B \tag{4}$$

Fig.8 shows example of parent-child (primary-foreign) relationships formed within the African development tables of The World Bank database [10]. In case of not finding primary-foreign key relationships between clusters, if the matching clusters tend to partially match, we consider them as partially related keys because of their partial level of significance. Despite having relatively weak relationships, our experiments have shown that these partial relationships which have been constantly ignored in current techniques hold valuable data facts.

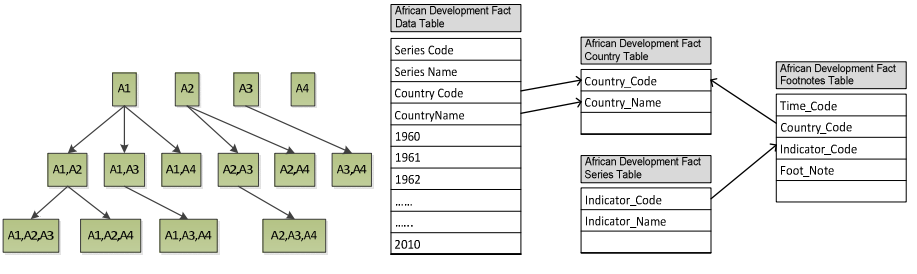


Fig. 8. The left diagram shows incremental pair-wise comparisons applied on cluster pairs in order to determine primary key in a table. The right diagram shows example of parent-child relationships formed in the African development indicators data (The World Bank [10]).

4 Empirical Evaluation

We have evaluated the proposed OGDL framework on several real-world databases. Our goals were to evaluate the accuracy, scalability and performance of OGDL and OGDL components. We report the evaluation of our framework using different datasets whose characteristics are listed in Table 1. These databases are publicly available which were used as the "gold standard" for our experimental study.

Table 1. Database Setup

Database #	Tables #	Attributes #	Rows #
The World Bank Data Catalog [10]	4512	67680	2.0 M
The US Federal Govt. Data Catalog [11]	3155	43339	1.4 M
The World Wildlife Fund Data Catalog [12]	21	472	55 K
The Adventure Works Database [13]	254	2608	24 K
National Climatic Data Center [14]	255	1350	15 K
Queensland Govt. Wildlife & Ecosystems [15]	102	259	1.2 K
Medical Data Sets [16]	129	390	33 K

Accuracy Metrics. Accuracy is the process of ensuring that the results obtained are closely associated to expected true values. The analysis of the accuracy in our application has to be focused on the primary keys, foreign keys and partial keys discovery. The approach we took in discovering attribute relationships was to run our prototype against relational databases already having tables in third normal form with primary and foreign keys. We manually deleted all the existing relationships and executed OGDL to see if it finds deleted relationships. The results were highly accurate as shown in Fig.9. Even though, OGDL didn't find expected number of relationships, it categorized a new set of partial key relationships where the missing relationships could be found. The reason we got large number of partial keys is due to the fact that, OGDL relies on sampling technique (see section 3.1) in order to reduce the computational expense in dealing with large volumes of data and employs probabilistic cluster matching technique at different levels. The conclusion from our research is that our proposed approach matches 87% of the cases.

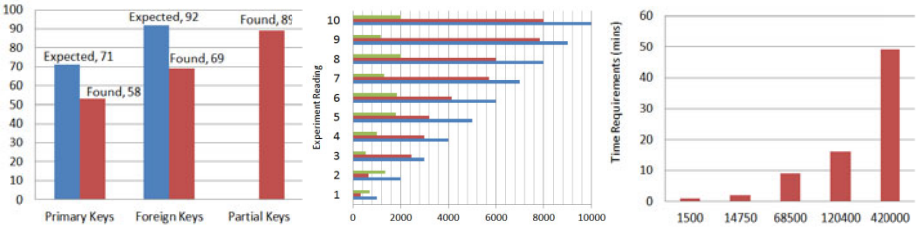


Fig. 9. The first graph (from left to right) shows the accuracy test results carried against the relational databases in order to discover primary, foreign and partial keys. The second graph displays the cluster counts formed in different attribute sets. The third graph displays the performance metric results i.e. showing the time requirements for discovering relationships under different data input sizes.

Scalability Metrics. Scalability tests are aimed at assessing the workload required, system results throughput and the ability to handle varied sets of data. In our experimental study, we focused scalability based on different sets of clusters formed versus attribute counts as shown in Fig.9. We observe the number of clusters discovered by OGDL is very close to the number of attributes. This further indicates the effectiveness of OGDL clustering technique. The OGDL clustering process represents data into much smaller datasets by accurately modeling huge amounts of information given as input and by relying on easily-analyzable sets of cluster.

Performance Metrics. Performance is the throughput gained against time and resources. The analysis of performance in our application has been considered based on the time requirements under different input size conditions as shown in Fig.9. It has to be noted that the conditions being accounted for were the ones from the primary key, foreign key and partial key findings. The conclusion we made is that our application takes advantage from the reduction in the amount of data carried out from the clustering process. For the smaller dataset, the computation time is quite close, while computation becomes relatively slower as the size of the dataset size increase. The reason for this increase is due to the large number of attributes considered in our experimental study (see Table 1). Nevertheless, due to the peculiarities of its constituent process (big time consumption, analysis to be performed just once), OGDL is accurate on both relational and non-relational test data, and gracefully scales up to large reference table sizes and reduce memory requirements without compromising on accuracy.

5 Conclusion and Future Work

Many organizations provide their data sets for free share [10 to 16]. Their data sets are mostly extracted as text from their internal relational databases, where every record is unique and mostly are in third normal form. For querying these data sets without prior knowledge of their schemas, this paper proposed an approach of ontology-guided data linkage for sharing these open sourced data sets. In summary, we presented a thorough analysis of handling data uncertainties. We then presented a novel approach to form ontological clusters and establish hierarchical relationships between clusters using multi-faceted technique. We also showed how the results obtained from our framework can be easily narrowed down to create global schema. Using real-world datasets from different domains, we established the overall accuracy and robustness of our system and confirmed that the development of OGDL is highly significant. Furthermore, we explained how our results can be integrated with IBM or Microsoft's QBE (Query-by-Example) tool's in order to perform semantic queries, which is part of our future work in consideration. For future work, we are also considering development of faceted data search engine system in order to facilitate newly emerging faceted browsing. In summary, our approach can help different anticipating bodies towards extracting and discovering required meaningful data facts without prior knowledge of raw data, having little or no documentation and without waiting for a long delay at run time.

References

1. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer, Heidelberg (2007)
2. Gagnon, M.: Ontology-based Integration of Data Sources. In: IEEE 10th International Conference on Information Fusion, Que, Canada, pp. 1–8 (2007)
3. Bonifati, A., Mecca, G., Pappalardo, A., Raunich, S., Summa, G.: Schema Mapping Verification: The Spicy Way. In: EDBT 2008, Nantes, France, March 25-30, pp. 1289–1293 (2008)
4. Radwan, A., Popa, L., Stanoi, I.R., Younis, A.: Top-K Generation of Integrated Schemas Based on Directed and Weighted Correspondences. In: SIGMOD, Providence, Rhode Island, USA, June 29-July 2, pp. 641–654 (2009)
5. Llyas, I.F., Markl, V., Haas, P., Brown, P., Aboulnaga, A.: CORDS: Automatic Discovery of Correlations and Soft Functional Dependencies. In: SIGMOD 2004, France, pp. 647–658 (2004)
6. Wu, W., Reinwald, B., Sismanis, Y., Manjrekar, R.: Discovering Topical Structures of Databases. In: SIGMOND, June 9-12, pp. 1019–1030 (2008)
7. Pudi, V., Krishna, P.R.: Data Mining. OXFORD University Press (2009)
8. Fuxman, A., Hernandez, M.A., Ho, H., Miller, R.J., Papotti, P., Popa, L.: Nested Mappings: Schema Mapping Reloaded. In: VLDB, Seoul, Korea, pp. 67–78 (2006)
9. Pottinger R., Bernstein P.A.: Schema Merging and Mapping Creation for Relational Sources. In: EDBT 2008, Nantes, France, pp. 73–84 (2008)
10. The World Bank, Data Catalog, http://data.worldbank.org/topic
11. The US Federal Govt., Data Catalog, http://www.data.gov/catalog
12. The World Wildlife Fund, http://www.worldwildlife.org/science/data/item1872.html
13. The Adventure Works Database, http://sqlserversamples.codeplex.com/
14. National Climatic Data Center, http://www.ncdc.noaa.gov/oa/ncdc.html
15. Qld.Wildlife & Ecosystems, http://www.derm.qld.gov.au/wildlife-ecosystems/index.html
16. Medicare Databases, http://www.medicare.gov/download/downloaddb.asp
17. ARFF, WEKA, University of Waikato, http://weka.wikispaces.com/XML
18. Gupta, S.C., Kapoor, V.K.: Fundamentals of Mathematical Statistics, 11th edn. Sultan Chand & Sons (2009)
19. Franklin, M.J., Halevy, A.Y., Maier, D.: From Databases to Dataspaces: a new abstraction for information management. SIGMOD, Record 34(4), 27–33 (2005)
20. MSDN, Hashing, http://msdn.microsoft.com/en-us/library/system.object.gethashcode

Predicting New User's Behavior in Online Dating Systems

Tingting Wang[1], Hongyan Liu[2], Jun He[1,*], Xuan Jiang[1], and Xiaoyong Du[1]

[1] Key Labs of Data Engineering and Knowledge Engineering, Ministry of Education, China
School of Information, Renmin University of China, 100872, China
{wtt526,hejun,jx,duyong}@ruc.edu.cn
[2] Department of Management Science and Engineering, Tsinghua University, 100084, China
liuhy@sem.tsinghua.edu.cn

Abstract. Predicting new user's reaction behavior to its recommended candidate partner correctly is critical to improve recommendation accuracy in online dating systems. However, new user (cold start) problem and data sparseness problem in the online dating system make this task very challenging. In this paper, we propose a hybrid method called crowd wisdom based behavior prediction to solve the two problems and achieve good prediction accuracy. By this method, old users who have been recommended partners before are first separated into groups. Users in each group have similar preference for partners. Then, we propose a novel measure to combine a group user's collective behavior to predict one user's behavior, which can solve the data sparseness problem. By calculating the probability a new user belongs to each group and utilizing the group's behavior we can solve the new user problem. Based on these strategies, we develop a behavior prediction algorithm for new users. Experimental results conducted on a real online dating dataset show that our proposed method performs better than other traditional methods.

Keywords: online dating, recommendation, clustering, classification.

1 Introduction

With the advance of web and information technology, more and more people make friends through online dating systems. In this kind of systems, each registered user provides both its own personal information and its basic match requirement about its partner. Based on the basic requirement, we can find matches among all of users registered. But most of these matches are usually not satisfactory. Therefore, if we can predict which match is better based on historical matching records, we can increase users' satisfactory and can improve the online dating website's viscosity, and hence increase profit of the online dating company. However, this is a very challenging problem because of cold start problem and data sparseness problem. In this paper, we study how to predict which user will be preferred by new users (also

* Corresponding author.

J. Tang et al. (Eds.): ADMA 2011, Part II, LNAI 7121, pp. 266–277, 2011.

called target users) given a list of candidate users who meet the target users' basic requirements. In this study, preference is reflected by two kinds of user behaviors. One is *click*, and the other is *sending message* (*message* or *msg* for short). If the target user is interested in a particular candidate recommended to him or her based on basic information such as name, age and photo, the target user may click the candidate person for further personal information, which is the so-called *click* behavior. After that, if the user wants to interact with the candidate, the user can send message to the candidate by clicking a *message* button to get the contact information, which refers to the *msg* behavior. In the following, we use *click* and *message* as nouns and verbs interchangeably. As sending message means higher preference than click, we give more weights to *message* than to click if we correctly predict this behavior. The online dating company gets payment for each message behavior. In other words, our task is to predict which one will be clicked and which one will be sent message by a target user, given a list of candidates meeting the target user's basic requirement.

To address this issue, two problems must be considered. One is new user problem, which means that this user has not been recommended any user before. It's hard to know its preference by existing behavior data. Both basic content-based and collaborative filtering methods [10, 11, 12] cannot deal with problem well. The other is data sparseness problem. Two actual similar persons maybe have no or little intersection of their candidate users. It's not enough to only use the actions of its similar users done to its candidate users, because the behavior information is usually very sparse. In our experiments, user behavior data only includes about 1% user pair action rating, missing all other user pairs action ratings. Clustering users by user profile information has been used to settle cold start problem [8, 9]. But methods to deal with both new user and data sparseness problem are still necessary. In this paper, we focus on how to solve these two problems at the same time. To solve the data sparseness problem, we propose to use a group user's collective behavior to predict one user's behavior. We divide users into groups based on their historical behaviors and calculate the probability the new user belonging to each group to solve the new user problem. Based on the two basic methods we develop an algorithm to predict new user's reaction to recommended user. We conduct experiments to evaluate the performance of the proposed algorithm as well as two other methods. Experimental results demonstrate the effectiveness of our proposed algorithm.

The rest of the paper is organized as follows. In section 2, problem definition and basic prediction methods are given. The detailed algorithm is introduced in section3. Experiment setup and results are described in section 4. In section 5, related works is discussed. Lastly, conclusions are drawn in section 6.

2 Problem Definition and Basic Prediction Methods

Let U be a set of users registered in an online dating system. Each user has a unique ID assigned by the system. From the website, we can get historical matching dataset about user recommendation and user behavior information, which is in the form of tuples (*uida*, *uidb*, and *action*), where action has three values, *rec*, *click*, and *message*. *Rec* means the online dating system recommended user with ID of *uidb* to user uida, providing name, age and photo of *uidb* to *uida*, *click* means *uida* clicked *uidb* for

further personal information, and *message* means that uida sent message to *uidb*. Both *uida* and *uidb* are registered users in the website. We call those who have received recommendations old users, and those who have never been received any recommendation new users or target users. Each user has two kinds of information, one is their personal information such as *age*, *height* and *income*. The other kind is match requirement information for their partners, including the requirements about age, height, income and so on. In a word, we save these information into three tables, *behavior*(*uida*, *uidb*, *action*), *personal*(*uid*, *gender*, *age*, *work_location*, *photo*, *income*, *education*, …) and *requirement*(*uid*, *min_age*, *max_age*, *min_height*, *max_height*, …), which as a whole we call dataset *D*. Examples of these data are given in Tables 1, 2 and 3 respectively.

Table 1. Behavior table

uida	*uidb*	*action*
100033	375879	rec
100033	381720	rec
……	381720	rec
……	417848	rec
……	417848	click
100033	327685	msg
381720	372934	rec

Table 2. Personal information

Uid	*gender*	*age*	*work_location*	……
100033	1	27	10	……
381720	0	23	10	……
……				……
……				
327685	0	23	10	……

Table 3. Requirement information

uid	*min_age*	*max_age*	*min_height*	*max_height*	……
100033	22	28	160	178	……
381720	25	30	175	180	……
……					
……					
327685	25	30	175	178	……

Given dataset *D*, let U_t be the set of users who have never been recommended any candidate user yet. For any target user $u \in U_t$ and a set $U_c=\{u_i \mid u_i \in U$ and u_i meets user u's match requirements$\}$ of candidate users, our mining task is to predict which user in U_c user u will only click for more information, to which user user u will further send message and which user user u will have no action.

Note that although user u has never been recommended any user yet, i.e, u doesn't occur in the column *uida* of table behavior, it may occur in column *uidb*. For any

candidate user $u_i \in U_c$, u_i may occur in both column *uida* and *uidb*. In fact, most of candidate users occur in the behavior table.

To fulfill this task, we can regard action as class label and build classification model to do prediction. To this end, we need to build a classifier training dataset from the dataset D. The challenge to do this is that different training set may have different performance in terms of prediction accuracy. The hard thing is to find important attributes which are relevant to user's behavior.

We can assign values to actions and regard the value as rating of product (regard each candidate as a product), for example, for user u and candidate u_i, if u message u_i, $r(u, u_i)=2$, if u click u_i, $r(u, u_i)=1$ and otherwise, $r(u, u_i)=0$. In this way, prediction of the behavior is equivalent to predict the action value or target user's rating of particular candidate user. Therefore, we can map this mining task to recommendation task, which can be solved by collaborative filtering, content-based recommendation and hybrid methods. But recommendation methods suffer from cold start problem and date sparseness problem, which are just what we want to solve in this paper. If we assume two users have similar preference if they are liked by same users, we can find similar users of target user u, and based on the actions these similar users done to user u's candidate user u_i, we can predict the action user u could take towards candidate user u_i. Similarly, if we assume users who have similar personal and requirement information will have similar preference, we can also handle the new user problem. We will describe the details of these methods in the experiment section. But due to data sparseness problem, it's hard to find users similar to user u. Also, the assumption may not be correct, as people liked by same persons may not like similar persons. Therefore, we need to develop other method.

By exploring the data in the behavior table, we find that users are naturally clustered into groups according to their preferred candidate users. If we can successfully assign a target user to one of these groups, we can solve the cold start problem. And then we can make use of the action information of the whole group to predict its action, which can solve data sparseness problem, as it's hard to find similar users based on behavior information due to new user and data sparseness problem. We call this method *crowd wisdom based behavior prediction*. To implement this method, we have three questions to answer.

1) How to cluster users in training behavior data into groups?
2) How to assign target users to groups?
3) How to combine behavior information to make predictions?

We will discuss the first two questions in the next section. Now we focus on the third question.

Suppose we have k groups $G_s = \{G_1, G_2, \ldots, G_k\}$, target user u's candidate user u_i belongs to group G_i (u_i is one of the recommended users in this group), $1 \leqslant i \leqslant k$, and u has probability of $p(G_i \mid u)$ belonging to group G_i. If $p(G_i \mid u)$ is high, we want to use the possibility of u_i's being clicked or messaged in this group to predict how u_i could be clicked or messaged by user u. To do this, we develop a measure called *msgrate* to measure the possibility of u_i's being clicked or messaged in group G_i, which is calculated by Equation 1.

$$msgrate(u_i, G_i) = \frac{\alpha \cdot \sum_{u_j \in G_i} rec(u_j, u_i) + \beta \cdot \sum_{u_j \in G_i} click(u_j, u_i) + \gamma \cdot \sum_{u_j \in G_i} msg(u_j, u_i)}{\sum_{u_j \in G_i} rec(u_j, u_i) + \sum_{u_j \in G_i} click(u_j, u_i) + \sum_{u_j \in G_i} msg(u_j, u_i) + \varepsilon} \quad (1)$$

In Equation 1, $rec(u_j, u_i)$ is the number of time user u_i is recommended to user u_j as candidate match in group G_i, and accordingly, $click(u_j, u_i)$ and $msg(u_j, u_i)$ is the numbers of times user u_j gives user u_i the action "click" and "message" respectively. Parameters α, β and γ give the different actions different weights to show their effect. And ε is a smoothing factor, which is used to avoid zero value or a very small value of the denominator when user u_i has never or seldom been recommended in group G_i.

Intuitively, Equation 1 means that if a user has more chances to be clicked or messaged, the *msgrate* will be high, which means this user has high possibility to be clicked or messaged by other similar users. The denominator is a kind of normalization to relieve bias for frequently recommended users. We then combine $p(G_i \mid u)$ and $msgrate(u_i, G_i)$ to predict how likely user u would click or message u_i. The detail is described in the next section.

For different groups, the four parameters, α, β, γ and ε, could be different. Assuming that these parameters are independent, in our experiment, we get the approximate optimal values of these parameters by enumerating some possible values in a given interval. For α and ε, we set the interval to be [0, 1], and β and γ [0, 30]. The increase step starting from 0 for interval [0, 1] is 0.01, and for interval [0, 30] is 1. Each time, we fix three of them and change the value of one parameter to get a good value. To judge if a combination is good, we separate 20% tuple from the training behavior data as testing data. Finally, in our experiment, for a male, we give α, β, γ and ε the values 0.04, 15, 13, 0.57, and for a female, the values are 0.03, 25, 13, 0.35 respectively.

From probabilistic point of view, given a group G_i, we can estimate the probability that user *u clicks* or *messages* user u_i in this group by the following equations.

$$p(action = click\ or\ msg \mid uida = u, uidb = u_i) \quad (2)$$

$$= \frac{p(uida = u, uidb = u_i \mid action = click\ or\ msg) \times p(action = click\ or\ msg)}{p(uida = u, uidb = u_i)} \quad (3)$$

$$= \frac{p(uida = u \mid action = click\ or\ msg) \times p(uidb = u_i \mid action = click\ or\ msg) \times p(action = click\ or\ msg)}{p(uida = u) \times p(uidb = u_i)} \quad (4)$$

$$\propto \frac{p(uidb = u_i \mid action = click\ or\ msg)}{p(uidb = u_i)} \propto \frac{p(uidb = u_i, action = click\ or\ msg)}{p(uidb = u_i)} \quad (5)$$

According to naïve Bayesian theory, the equation (2) can be rewritten as the equation (3). Moreover, we assume that the occurrence of the two users is independent given click or message action. Then, the equation (3) can be rewritten as the equation (4). Furthermore, considering our aim is to compare the scores a target user gives to different candidate users, the ranking is unrelated to the target user itself. So the above probability is expressed like equation (5). From the above equations we can conclude that a target user is more likely to click or message candidate users who are

more likely to be clicked or messaged by other users, which is consistent with our measure *msgrate*.

3 Algorithm *BehvPred*

In this section we discuss how to implement crowd wisdom based behavior prediction method to predict users' behavior in online dating websites. We first answer the three questions raised in Section 2, and then we describe our algorithm developed based on the answers to these questions.

3.1 Clustering Users

It is hard to find a new user's right similar users according to their common behaviors, because the new one doesn't have any actions before. We can use user's profile information (personal information and requirement information) find its similar users. But users with similar profile may not have similar action towards same users. Here we use both the action information and profile information to find a group of users similar to a particular target user. To do this, we first need to cluster old users into groups according to their actions to recommended users.

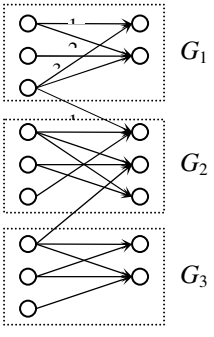


Fig. 1. Illustration of the result of graph partitioning using n-cut

To model the action relationship between users, we construct a bipartite graph $G(V_1, V_2, E)$, where V_1 is a set of old users, and V_2 is a set of users recommended to users in V_1. E is a set of edges. Each $edge(u_i, u_j)$ starting from user $u_i \in V_1$ to user $u_j \in V_2$ represents *click* and/or *msg* actions between them. Each $edge(u_i, u_j)$ is associated a weight computed by Equation 6.

$$weight(u_i, u_j) = w_1 \times click(u_i, u_j) + w_2 \times msg(u_i, u_j) \tag{6}$$

In our experiments, we give w_1 and w_2 values 1 and 2 respectively. We use graph partitioning algorithm *N-cut* [6, 7] to cut the bipartite graph G into 8 groups. Graph partitioning algorithms separate the nodes of a graph into n groups, with the number of edges connecting nodes cross different groups minimized. We tried different number of groups such as 4, 8 and 16. Comparing different number of groups, the

similarity within groups and dissimilarity between different groups come to maximum when the group number is 8. Before partitioning, we regard the whole graph as one group, and after partitioning, let G_s be the set of 8 sub-graphs, $G_s=\{G_1, G_2, \ldots, G_8\}$, where each subgraph is still a bipartite graph $G_i(V_{i1}, V_{i2}, E_1)$, $i \in \{1, \ldots, 8\}$.

3.2 Assigning Users to Groups

To assign new users to groups, we need to compute the probability a user u belongs to each groups. To this end, we use classification technique. After clustering step, each old user has a group label, which is also a class label. So, we construct a training dataset which consists of all of old users and each user is described by both personal information and requirement information, 34 features in total. Then, feature selection algorithm is used to find important key features. Correlation-based feature selection method in Weka is used for evaluating the attributes' importance, and *BestFirst* search method defines the search order. After this step, 34 features are reduced to 17 features. Finally, for each target user u, we use naïve bayes to compute the probability u belongs to each group given its personal and requirement attribute values.

3.3 Combing Group Behavior

For a target user u and one of its candidate user u_i, there are two cases. One is that user u_i occurs in the right side of one subgraph $G_i \in G_s$. The other is that user u_i doesn't occur in any groups.

For the first case, suppose user u_i occurs in group G_i such that $u_i \in V_{i2}$. Then, as discussed in Section 2, we can combine $p(G_i \mid u)$ and $msgrate(u_i, G_i)$ to predict how likely user u would click or message u_i. The likelihood score is computed by Equation 7.

$$score(u,u_i) = \begin{cases} p(G_i \mid u) \times msgrate(u_i, G_i) & \text{if } p(G_i \mid u) \geq \theta \\ msgrate(u_i, G) & otherwise \end{cases} \quad (7)$$

In the above Equation, θ is a threshold to choose a group user u is similar to. In our experience, we set $\theta = 0.01$. If user u is not similar to any group, we use the *msgrate* of user u_i in the whole training data D corresponding to the whole graph G.

For the candidates who have never been recommended to any old user before, the above formula cannot process. This is the new item problem in the recommendation systems. As we have user profile information, we can use the profile information to find similar users. Every attribute is nominal(categorical). We use Manhattan distance after normalization to measure the dissimilarity between users.

Suppose the most similar user is u_l. Then we use $score(u, u_l)$ to approximate $score(u, u_i)$. We can also find more than one similar users, and use weighted average of the scores to approximate. In our experiments, we use one most similar user.

3.4 Algorithm

We develop an algorithm, *BehvPred*, to integrate the three steps discussed in the above. The major steps of the proposed algorithm are summarized in Figure 2.

Algorithm ***BehvPred***: behavior prediction
Input: training dataset D and bipartite graph G, target user u and U_t, a set of u's candidate users, similarity threshold θ
Output: a ranked list of candidate users
Begin
1. use n-cut algorithm to cluster the users of D into k groups
2. Build Bayes classifier based on the clustering results
3. for each candidate user $u_i \in U_t$
4. If there exists a subgraph $G_i(V_{i1}, V_{i2}, E_1)$ such that $u_i \in V_{i2}$
5. calculate $msgrate(u_i, G_i)$ by Equation 1
6. else
7. calculate $msgrate(u_i, G)$
8. for each group G_i
9. calculate probability of user u belonging to each group, $p(G_i | u)$
10. calculate $score(u, u_i)$ by Equation 7
11 rank users in U_t by descending order of score.
End

Fig. 2. Major steps of algorithm *BehvPred*

4 Experiments

4.1 Dataset

We conduct experiments on a real dataset obtained from an online dating system. This dataset includes the aforementioned three tables, behavior, personal and requirement. There are 8599012 tuples in behavior table and 548393 persons in both personal and requirement tables. There are 24 attributes in table personal and 10 attributes in table requirement. In the behavior table, 95% percent of tuples have action of *rec*. To evaluate the performance of different methods, we divide the behavior table into two parts, training behavior data(80%) and test behavior data(20%). Other statistics about this dataset are given in Table 4.

Table 4. Some figures about datasets

items	value
# of distinct uidas in training behavior data	12000
# of distinct uidbs in training behavior data	55029
# of "rec" actions in training behavior data	6659813
# of "click" actions in training behavior data	149394
# of "msg" actions in training behavior data	38273
# of distinct uidbs in test data	25168
# of distinct uidbs in both test behavior data and training behavior data	25103

4.2 Evaluation Criteria

To evaluate the performance of different methods, for each target user and ten recommended candidate users, each method ranks the ten candidates. Those who are mostly likely to be messaged are ranked in the top, and those who are possibly be clicked follows. To measure how accurate the rank is, we use a measure called *NDCG* (normalized discounted cumulative gain). For a target user u, its $NDCG(u, n)$ is computed as follows.

$$NDCG(u,n) = \frac{DCG(u,n)}{IdealDCG(u,n)} \tag{8}$$

$$DCG(u,n) = \sum_{i=1}^{n} \frac{2^{r(u,u_i)} - 1}{\log_2(1+i)} \tag{9}$$

$$NDCG@n = \frac{\sum_{u \in U_t} NDCG(u,n)}{|U_t|} \tag{10}$$

Suppose we set n =10, which means that we consider the top 10 recommended users for every target user. Assume the rank of ten candidates of target user u returned by one of the prediction methods is $u_1 > u_2 > \ldots > u_{10}$. In equation 9, $r(u, u_i)$ is the value of the action user u took for the ith ranked candidate u_i. Value of action *rec* is 0, *click* is 1 and *msg* is 2. *IdealDCG* is the *DCG* for the correct rank. In this way, we can compute a *NDCG* score for each target user. Then we use the average *NDCG* of all of target user $u \in U_t$ by Equation 10 to measure the accuracy of the prediction method.

4.3 Classification Method and Collaborative Filtering Method

In our experiments, we compare classification method and collaborative filtering method with our proposed method.

For classification method, we first use feature selection method to select important features. And then, three classification methods including *Naïve Bayes*, *Decision Tree* and *SVM* are used to build classifiers. To build the training dataset, for each tuple of the training behavior data, both attributes *uida* and *uidb* are replaced by attributes about the users' personal information and requirements information. Besides, another two attributes are added to it. One is click-probability of user *uidb*, which is the probability that the user is clicked by any other user. The other is message-probability of user *uidb*, which is the probability that the user is messaged by any other user. As a result, 17 attributes are remained for classification.

Collaborative filtering method predicts the rating scores (here the action) by finding the users who have similar behavior with the target user. Unfortunately, each target user is a new user who has no behavior information before. Therefore, we need to develop ways to handle the cold start problem. Suppose we are predicting the action target user u may take towards candidate user u_i and let U_s be a set of users

who have been recommended u_i. We have two methods. The first method is called behavior based method. It is based on the assumption that two users have similar preference if they are liked by same users. For a user $u_a \in U_s$, we calculate similarity between u and u_a by $cosine(v(u), v(u_a))$, where $v(u)$ or $v(u_a)$ is a rating vector. Suppose that there are m users who have actions to both user u and u_a in the training behavior data. Then $v(u)$ consists of ratings like $r(u_b, u)$, where u_b is one of the m users. Similarly we can obtain $v(u_a)$.

Similarly, if we assume users who have similar personal and requirement information have similar preference, we can find similar users among users in U_s by cosine similarity. This method is called profile based method.

For a target user u, some most similar users whose similarity are bigger than some threshold (we set it 0.7) are selected. Then the most similar users (let U_k be the set of these users) found by the above methods are used to predict the action user u may take towards candidate user u_i by equation 11.

$$r(u,u_i) = \frac{\sum_{u_a \in U_k} sim(u,u_a) \times r(u_a,u_i)}{\sum_{u_a \in U_k} sim(u,u_a)} \tag{11}$$

According to the rating score estimated by the following methods, for each target user u, we can get a ranked list of its candidates and compute *NDCG* for the list. By comparing these two methods, we find the profile based method performs better than the behavior based method. This may be because the assumption is not correct. In the following comparisons, we only compare the profile based method with other methods.

4.4 Comparison and Discussion

Experimental results of the five methods conducted on the real data set are shown in Table 5.

Table 5. Experiment results

methods	*NDCG*@5	*NDCG*@10	*NDCG*@20
Naïve Bayesian	0.137	0.143	0.153
Decision tree	0.105	0.111	0.119
SVM	0.105	0.111	0.119
CF	0.062	0.069	0.076
BehvPred	**0.200**	**0.242**	**0.276**

In the clustering step of our method, the number of groups k can be set different values. The smaller k is, users in one group will be less similar. If k is too big, the dissimilarity between different groups becomes less and hence the classification accuracy in the next step will also be affected. In our experiments, we set $k = 8$ to

make sure users in one group are similar and the classification accuracy is acceptable in the next step.

As shown in Table 5, among three classification methods, *Naïve Bayesian* gets the highest *NDCG*. And *CF* (profile base collaborative filtering method) doesn't work well, because few peoples share same candidate users and have similar profile information at the same time. Our proposed method performs best among these methods. The result indicates that our method to deal with cold-start problem and date sparseness problem is effective when predicting people's preference towards mates.

5 Related Works

As we discussed in the paper, predicting user behavior can be regarded as candidate user recommendation problem. Therefore, traditional recommendation methods are related to our work. Content-based recommendation and collaborative filtering recommendation methods [5] are the most commonly used recommendation methods. Basic content-based methods recommend similar items according to the features of items which the target user liked before. However, new user problem limit its use. Classification method such as Naïve bayes classification [1] and decision tree are often used in content-based recommendation methods.

Collaborative filtering (*CF*) method makes recommendation by finding similar users or similar items, which is often called user-to-user model and item-to-item model. User-to-user method recommends items liked by the users who are similar to the target user.

Works about recommendation in online dating systems are directly related to our work. Krzywicki et. al [2] developed and evaluated several collaborative filtering methods in online dating system. Only user interaction information is used in this paper. Kazienko and Musial [3] claim that both the profile information and the interaction among users should be considered. Work in [2] implies its algorithm can be used as a basis of a recommender system for online dating. However, new user problem is still unresolved in this paper. Chen et. al [4] proposed a method combining clustering with *SimRank* to recommend matching user pairs. The author use *SimRank* as a similarity measure to find similar users of the target user. However, this method doesn't work when dealing with a new user because of lack of the interaction information of the target user.

6 Conclusions

In this paper we study the problem of predicting new user behaviors in online dating systems. Given a new user (or target user) and a list of candidate users meeting basic match requirement, we can recommend the most suitable candidates to the new user if we can predict what action the new user will take towards the candidates. But due to the cold start problem and data sparseness problem, traditional methods such as recommendation methods don't work well on this setting. Therefore, we propose a new method called crowd wisdom based behavior prediction, which could solve these

two problems at the same time. Based on this method, we design and implement an algorithm, *BehvPred*. Experiments conducted on real online dating dataset show that our proposed method is effective to deal with these problems and the algorithm has higher prediction accuracy than existing methods.

To increase the matching success rate, we still have lots to do. For example, if we know the reaction after a user sends message to a candidate, we may have more chance to improve the recommendation accuracy.

Acknowledgement. The work was supported in part by the National Nature Science Foundation of China under Grant No. 70871068 and 70890083 and HGJ project 2010ZX01042-002-002-03.

References

1. Mooney, R.J., Bennett, P.N., Roy, L.: Book Recommending Using Text Categorization with Extracted Information. In: Proc. Recommender Systems Papers from 1998 Workshop, Technical Repot WS-98-08 (1998)
2. Krzywicki, A., Wobcke, W., Cai, X., Mahidadia, A., Bain, M., Compton, P., Kim, Y.S.: Interaction-Based Collaborative Filtering Methods for Recommendation in Online Dating
3. Kazienko, P., Musial, K.: Recommendation FrameWork for Online Social Networks. In: The 4th Atlantic Web Intelligence Conference (AWIC 2006), pp. 110–120. Springer, Washington D.C (2006)
4. Chen, L., Nayak, R., Xu, Y.: Improving Matching Process in Social Network. In: 2010 IEEE International Conference on Data Mining Workshop (ICDMW), pp. 305–311 (2010)
5. Burke, R.: Hybrid Recommender Systems: Survey and Experiments. User Modeling and User-Adapted Interaction 12, 331–370 (2002)
6. Karypis, G., Kumar, V.: Multilevel k-way partitioning scheme for irregular graphs. Journal of Parallel and Distributed Computing 48(1), 96–129 (1998)
7. Karypis, G., Kumar, V.: METIS: Unstructured Graph Partitioning and Sparse Matrix Ordering System. Technical Report, Department of Computer Science, University of Minnesota (1995)
8. Nayak, R.: Utilizing Past Relations and User Similarities in a Social Matching System. In: Huang, J.Z., Cao, L., Srivastava, J. (eds.) PAKDD 2011, Part II. LNCS, vol. 6635, pp. 99–110. Springer, Heidelberg (2011)
9. Nayak, R., Zhang, M., Chen, L.: A Social Matching System for an Online Dating Network: A Preliminary Study. In: IEEE International Conference on Data Mining Workshops, ICDMW 2010, pp. 352–357 (201)
10. Adomavicius, G., Tuzhilin, A.: Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Transactions on Knowledge and Data Engineering 17(6), 734–749 (2005), doi:10.1109/TKDE.2005.99
11. Pazzani, M.J.: A Framework for Collaborative, Content-Based and Demographic Filtering. Artificial Intelligence Review 13(5-6), 393–408 (1999), doi:10.1023/A:1006544522159
12. de Gemmis, M., Iaquinta, L., Lops, P., Musto, C., Narducci, F., Semeraro, G.: Preference Learning in Recommender Systems. In: ECML/PKDD 2009 Workshop on Preference Learning, PL 2009 (2009)

Sequential Pattern Mining from Stream Data

Adam Koper and Hung Son Nguyen

Institute of Mathematics, The University of Warsaw
ul. Banacha 2, 02-097 Warsaw, Poland
son@mimuw.edu.pl

Abstract. Sequential Pattern Mining, briefly SPM, is an interesting issue in Data Mining that can be applied for temporal or time series data. This paper is related to SPM algorithms that can work with stream data. We present three new stream SPM methods, called SS-BE2, SS-LC and SS-LC2, which are the extensions of SS-BE. The proposed methods, similarly to SS-BE, are dealing with fixed-sized batches using PrefixSpan algorithm, and the critical problem in each step is how to store the huge amount of candidate patterns, and how to select the frequent patterns properly. The main idea of based on improving the tree pruning method of the original SS-BE to guarantee the high completeness and correctness of the result. In all experiments performed on benchmark data, the proposed solutions outperform the original SS-BE algorithm. Moreover, the proposed algorithms seems to be scalable, as the usage of memory is linearly depended on the number of patterns, and the size of the buffer.

Keywords: Stream Sequential Patterns Mining, SS-BE, PrefixSpan, support measure.

1 Introduction

The SPM problem has been well recognized for the last 15 years, and many methods for static data have been developed. AprioriLike is the one of the first effective algorithm class, and GSP [2] method is a good representative example of this idea. Disadvantages of this approach have led to developing many new efficient pattern growth algorithms using tree structure in exploring. As an example one can mention the very effective algorithm PrefixSpan [3], or "PLWAP-Tree" [4]. In this case, exploration involves storing a source in the special tree structure, which facilitates exploration. The classification of the SPM algorithms may also be determined by the types of the data source. For the static data sources, it is assumed that data set has a fixed size and is available in complete form before the mining process. In other hand incremental methods are dealing with data that are changed in a fixed time intervals. Sequences in already existing set can be extended, but completely new sequences may also appear. As these changes are usually not significant, there is no need to process all data. Incremental algorithms consecutively modify the actual results basing just on the changed part of source data. IncSpan described in [5] is the best example of this approach.

Currently, stream data analysis is one of the most interesting challenges in data mining. This new type of data, there is no data size restriction, and there

J. Tang et al. (Eds.): ADMA 2011, Part II, LNAI 7121, pp. 278–291, 2011.

is no possibility to keep all data for further analysis due to the limited resources. Data are assumed to be available during stream reading only. Thus, a new class of SPM algorithms has to be developed to deal with this type of data. Among the algorithms for sequential patterns mining from stream data, the attention should be paid to SS-BE (*Stream Sequence miner using Bounded Error* [9]), which is one of the most efficient stream SPM algorithms in recent years. SS-BE possesses two special approved features. Firstly, all true patterns are output. Secondly, all false patterns that are returned by the algorithm, should be above some pre-defined threshold. Experiments on a synthetic data, revealed that the percentage of false patterns in the result is usually very high.

In this paper we propose three new methods SS-BE2, SS-LC, SS-LC2. All of them are based on SS-BE. SS-BE2 has different final criterion, which considerably increase the accuracy of the original SS-BE. The later algorithms SS-LC and SS-LC2 are using different setting for the PrefixSpan algorithm, the stoping criterion, and the tree pruning criterion. The efficiency of the proposed changes have been tested and proved to reveal better results in the accuracy, the performance, and the memory usage. Scalability of the proposed algorithms, just like SS-BE, is linear due to number of sequence and due to number of patterns. Memory usage is linearly dependent on number of patterns, and the buffer size. Experiments have confirmed that replacement of the SS-BE with the SS-LC method will be a good choice in all variety of conditions.

2 Basic Notions

Let $I = \{i_1, \ldots i_n\}$ be a set of items. Any ordered set of items $S = \langle s_1, \ldots s_m \rangle$, where $s_1, \ldots s_m \in I$, is called the sequence. We say that $S = \langle s_1, s_2, \ldots s_k \rangle$ is the *subsequence* of the sequence $S' = \langle s_1', s_2', \ldots s_l' \rangle$ (or S' *contains* S) and denote by $S \Subset S'$, if $k \leq l$ and there exists $1 \leq i_1 < i_2 < .. < i_k \leq l$, such that $s_1 = s_{i_1}', s_2 = s_{i_2}', \ldots, s_k = s_{i_k}'$. The sequence database is defined as a set of the fixed number of sequences, while the sequence data stream is understood as a unlimited sequence series. For the given data stream $\mathcal{S} = \{S_1, S_2, \ldots\}$ the number of sequences already obtained from $\mathcal{S}$, is denoted as *sequenceCount*.

For any sequence S, we denote by $count(S)$ the number of sequences already obtained from the stream that contain S. The support of sequence S (denoted as $supp(S)$) is defined by $\frac{count(S)}{sequenceCount}$. We are looking for the sequences, whose supports are at least as a predefined threshold (denoted as $minSupp$). If sequence S satisfies the inequality $supp(S) \geq minSupp$, we call it *sequential pattern*. Our goal is to find as small as possible set of candidate sequences that contains all the sequential patterns. We do not require to find a set containing just the true sequential patterns, because we want to allow using efficient methods of exploration. This will give us less memory consumption.

Example. Lets take a data stream containing 4 sequences $D = \{S_1 = \langle 3, 1, 2 \rangle$, $S_2 = \langle 1, 2, 3, 1 \rangle$, $S_3 = \langle 4, 1, 1, 2 \rangle$, $S_4 = \langle 2, 3, 3, 4, 1 \rangle\}$ (see Fig. 1). If we assume that $minSupp = 0.5$, then the sequential patterns are as follows: $\langle 1 \rangle : 4$, $\langle 2 \rangle : 4$, $\langle 3 \rangle : 3$, $\langle 4 \rangle : 2$, $\langle 1, 2 \rangle : 3$, $\langle 2, 1 \rangle : 2$, $\langle 3, 1 \rangle : 3$, $\langle 2, 3 \rangle : 2$, $\langle 4, 1 \rangle : 2$, $\langle 1, 1 \rangle : 2$, $\langle 2, 3, 1 \rangle : 2$.

2.1 Algorithm *PrefixSpan*

PrefixSpan [3] is one of the pattern-growth class algorithms, is one of the best methods for solving SPM problem in static database. It uses the database projection technique to make the mining process more efficient. Fig. 1 presents a sample database, and the projected database. In this approach, the computing of $count(S)$ is performed only in that part of sequence database that contain the prefix of the candidate sequence.

Let P be a sequence of length $l \geq 0$. The sequence S is called the P-sequence if P is a prefix of S (or S is the postfix extension of P). We denote by $\mathcal{E}(P)$ the set of all P-sequences. Let $\{P_1 \dots P_m\}$ be the set of all P-sequential patterns of length $l+1$. It is obvious that $\{\mathcal{E}(P_1), \dots, \mathcal{E}(P_m)\}$ is the partition of $\mathcal{E}(P) - \{P\}$.

Let S be a sequence and let P be a subsequence of S. A P-sequence S' is called the *projection of S with respect to prefix P* if and only if S' is the maximal subsequence of S that has P as a prefix. If $S' = \{s_1, \dots, s_n\}$ is a projection of S with respect to prefix $P = \{s_1, \dots, s_m\}$ then the $S'' = \{s_{m+1} \dots s_n\}$ is called *the postfix of sequence* S with respect to prefix P, denoted as $S'' = S|_P$. If S' is not a subsequence of S then the projection and the postfix are empty. For example, let $S = \langle 1, 4, 4, 3, 2, 1\rangle$ be a given sequence, then

- sequences $\langle 1\rangle$, $\langle 1, 4\rangle$, $\langle 1, 4, 4\rangle$ are the prefixes of S, and sequences $\langle 1, 4, 3\rangle$, $\langle 4, 4\rangle$ are not the prefixes of S;
- for $P = \langle 1, 4, 3\rangle$, the projection is $\langle 1, 4, 3, 2, 1\rangle$, and $S|_P = \langle 2, 1\rangle$;
- for $P = \langle 4, 4\rangle$, the projection is $\langle 4, 4, 3, 2, 1\rangle$, and $S|_P = \langle 3, 2, 1\rangle$;
- for $P = \langle 1, 4\rangle$, the projection is $\langle 1, 4, 4, 3, 2, 1\rangle$, and $S|_P = \langle 4, 3, 2, 1\rangle$;

Let P be a sequence and let $\mathcal{B}$ be a sequence database. The *P-projected database*, denoted by $\mathcal{B}|_P$, is defined as the set of postfixes of all sequences in $\mathcal{B}$ with respect to the prefix P, i.e. $\mathcal{B}|_P = \{S|_P : S \in \mathcal{B}\}$ (see Fig. 1). If S be a P-sequence, then the support count of S in P-projected database $\mathcal{B}|_P$, denoted as $supp_{B|_P}(S)$, is the number of sequences C in $\mathcal{B}|_P$ such that $S|_P \Subset C$.

Algorithm 1 presents the outlines of PrefixSpan. Its correctness follows from the next properties (see [3]): for any two sequential patterns P_1 and P_2 in sequence database $\mathcal{B}$, if P_1 is a prefix of P_2, then

1. $\mathcal{B}|_{P_1} = (\mathcal{B}|_{P_1})|_{P_2}$
2. for any P_1-sequence S, $supp_B(S) = supp_{\mathcal{B}|_{P_1}}(S)$;
3. The size of P_1-projected database $\mathcal{B}|_{P_1}$ cannot exceed the size of $\mathcal{B}$.

SequenceID	Sequence
1	<3,1,2>
2	<1,2,3,1>
3	<4,1,1,2>
4	<2,3,3,4,1>

Prefix	Projected (postfix) database	Sequential patterns
<1>	<2>,<2,3,1>,<1,2>	<1>,<1,2>,<1,1>
<2>	<3,1>,<3,3,4,1>	<2>,<2,1>,<2,3>,<2,3,1>
<3>	<1,2>,<1>,<3,4,1>	<3>,<3,1>
<4>	<1,1,2>,<1>	<4>,<4,1>

Fig. 1. Example of sequence database, projected database and sequential patterns

Algorithm 1. $PrefixSpan(P, l, B|_P)$

Input: P = a sequential pattern (prefix); l = length of P; $B|_P$ = P-projected database
Output: The set of all sequential patterns

1: Find all the frequent items b in $B|_P$ such that $(count(b) \geq minSupp)$.
2: **for all** frequent item b **do**
3: $P^{'} := P \circ b$; (*append b to the last element of P*)
4: construct $P^{'}$-projected database $B|_{P'}$;
5: call $PrefixSpan(P^{'}, l+1, B|_{P'})$;
6: **end for**

2.2 SS-BE: Stream Sequence Miner Using Bounded Error

SS-BE is one of the most efficient SPM algorithms for stream data. It was developed in 2008 and was described in [9]. The advantage of this algorithm is that it outputs all true frequent sequences. Another feature of this algorithm is that all potential false frequent sequences have support below the predefined threshold. It is the first stream algorithm that guarantees this both features.

We applied a basic data structure called FSP-Tree (Frequent Sequential Patterns Tree) to store the input sequences from data stream and to calculate the occurrences of candidate sequences. Tree nodes represent items, and paths from a root to nodes represent sequences. Thus, every node n can be associated with a path P_n from a root to node n and a corresponding sequence S_n. Each tree node has 4 attributes:

- *item* - the item that is represented by this node
- *count* - the count of the sequence corresponding to the path from the root to this node
- TID - the number of data batches that introduced this node to the tree;
- *batchCount* - number of batches that contribute to *count* since TID batch

Assume that $D = \{S_1, S_2, ...\}$ is a data stream. The algorithm divides stream into packages by measuring first and the rest of packages during data flow. The main four steps that are performed in SS-BE are presented in Algorithm 2.

Batches created in this way includes L sequences and they are denoted as $B_1, ...B_k$. Each consecutive batch is explored using $PrefixSpan$ with α (batchSupp) based Criterion. Frequent sequences $S_1, S_2, ..$ with support $c_1, c_2, ..$ obtained from batch are inserted into T_0. If path corresponding to sequence does not exist, then it have to be inserted. New node parameters are set as follows: $batchCount = 0$, $count = 0$, $TID = k$. Now as node exists attributes must be modified as follows: increase *count* by c_i and increase *batchCount* by 1. Every time a batch number is a multiple of δ, pruning of tree T_0 is performed. Originally, the SS-BE algorithm used the following criterion:

Definition 1 (Final Criterion SS-BE). *Sequences corresponding to nodes from T_0, that satisfy the inequality count $\geq (\sigma - \epsilon)N$, are returned to output.*

Despite many advantages, the SS-BE also has some drawbacks. Fig. 4(a) shows the situations, when SS-BE can produce more false frequent patterns than the

Algorithm 2. Algorithm SS-BE:

Input: L = batch length (number of sequences in the batch); δ = tree pruning period;
$minSupp = \sigma$ (minimum support threshold);
$sigSupp = \epsilon$ (significance support threshold: $0 \leq \epsilon < \sigma$);
$batchSupp = \alpha$ (batch support threshold: $0 \leq \alpha \leq \epsilon$);

Output: The set of all sequential patterns;

1: Collect all data from the given batch
2: Any batch of data is explored using $PrefixSpan$ with the support threshold $batchSupp \leq minSupp$
3: Frequent patterns obtained in previous step are inserted into the tree. If the number of actual batch is a multiple of pruning period δ, move to (4), otherwise go to (1)
4: The tree T_0 is pruned by using pruning criterion ($sigSupp$)

true ones. This fact motivates us to investigate on the more efficient pruning criterions. The improved methods are described in the next sections.

3 Algorithm SS-BE2

In SS-BE2, we propose the improvement for pruning criterion. For each node the value B is defined as count of packages processed till last pruning phase before node insertion ($B = batchID - \delta\lfloor\frac{TID-1}{\delta}\rfloor$). Let $B^{'} = B - bachCount$. Pruning Criterion, can be defined as follows:

Pruning Criterion SS-BE2. *The entire subtree rooted at a given node should be remove if following inequality holds:*

$$count + B^{'}(\lceil \alpha L \rceil - 1) \leq \epsilon BL,$$

Correctness of tree pruning follows from Apriori principle. Above steps are repeated until user demand for result. Now, algorithm computes the number of sequences obtained from the data stream till this moment $N = kL$. All sequences corresponding to nodes that satisfy Final Criterion are output.

Definition 2 (Final Criterion SS-BE2). *Sequences corresponding to nodes from T_0, that satisfy inequality provided below are output ($b = batchCount$).*

$$count \geq (N\sigma) - (TID - 1)\epsilon L - (N - (b + TID - 1)L)\alpha.$$

This way of data processing and Final Criterion guarantee that all true sequences will be output and the support of all false positive sequential patterns is at least $(\sigma - \epsilon)N$. Below we present the precise formulation and proof of this fact.

Lemma 1. *For each node n from the tree T_0, the upper bound of true count of the sequence S corresponding to this node is evaluated by:*

$$U = count + B^{'}(\lceil \alpha L \rceil - 1)$$

Proof: For each batch B_i of last B batches, the value of U is increased by $\max\{count_{B_i}(S), \lceil \alpha L \rceil - 1\}$, where $count_{B_i}(S)$ is true count of S in batch B_i. □

Theorem 1. *SS-BE2 algorithm output all true sequential patterns. True support of false sequential patterns that are output is at least* $(\sigma - \epsilon)N$*, where* N *is the length of data stream (number of sequences received from data stream).*

Proof: (1) Suppose, the sequence S is pruned n times after processing batches $B_{f_1}, B_{f_2}, ...B_{f_n}$. Suppose last pruning phases before insertions was just after batches $B_{i_1}, B_{i_2}, ...B_{i_n}$. Let $d_k = (B_{f_k} - B_{i_k})$ for $k = 1, ..., n$ and let $p = d_1 + ... + d_n$. According to the definition of pruning phase and to Lemma 1, the true count of sequence S in this p packages is most equal to $\sum_{i=1}^{n}(\epsilon d_i L) = \epsilon pL$.
(2) Let $b = batchCount$. The Final Criterion is defined as $count \geq (N\sigma) - (TID-1)\epsilon L - (N-(b+TID-1)L)\alpha$. Second and last source of undercounting in the remaining $((N/L)-p)$ batches is when a sequence is not a frequent pattern in the batch. Thus, count of a sequence is less then $\alpha L \leq \epsilon L$. Count of these packages is $((N/L)-p-b)$, where $b = 0$, if node does not exists in tree. If node does not exist in tree then $p \leq TID-1$, where $TID = (N/L)+1$. Then, sum of all sources of undercounting is $p\epsilon L + (N/L - p - b)\alpha L \leq p\epsilon L + (N/L - (b+TID-1))\alpha L + (-p+TID-1)\alpha L \leq p\epsilon L + (-p+TID-1)\epsilon L + (N-(b+TID-1)L) = (TID-1)\epsilon L + (N-(b+TID-1)L)\alpha$.
(3) Moreover, any node corresponding to output sequence pattern have *count* at least $(N\sigma) - (TID-1)\epsilon L - (N-(b+TID-1)L)\alpha \geq (\sigma-\epsilon)N$, because algorithm increases *count* by true count in packages (or not at all if sequence count is too small). □

Example. Let $L = 4$, ($minSupp$) $\sigma = 0,75$, ($sigSupp$) $\epsilon = 0,5$, ($batchSupp$) $\alpha = .4$, (pruning period) $\delta = 2$. Let batch $B_1 = \langle 1,2,3\rangle, \langle 1,3\rangle, \langle 1,2\rangle, \langle 2,3\rangle$ and $B_2 = \langle 1,2,3,4\rangle, \langle 3,1,2\rangle, \langle 4,1,2\rangle, \langle 1,5,2\rangle$. It means that the stream length $N = 8$, so we will process two packages, after that we will execute a pruning phase and return result. We process first batch using $PrefixSpan$ with support threshold $\alpha = .4$ Frequent patterns from the first batch are: $\langle 1\rangle$: 3,$\langle 2\rangle$: 3,$\langle 3\rangle$: 3, $\langle 1,2\rangle$: 2,$\langle 1,3\rangle$: 2,$\langle 2,3\rangle$: 2. This frequent patterns are inserted into T_0 (Fig. 2). Afterwards second batch is processed, and the resulting tree is shown in Fig. 3 (left). The result of $PrefixSpan$ consists of: $\langle 1\rangle$: 4,$\langle 2\rangle$: 4,$\langle 3\rangle$: 2,$\langle 4\rangle$: 2,$\langle 1,2\rangle$: 4.

Tree pruning period is 2, thus pruning tree T_0 starts. For each node, the B is defined as the count of batches processed since the last pruning phase before last insertion into the tree. It is the first pruning phase, so it can be assumed that the last pruning phase was at time 0. Then, $B = 2$ for all the

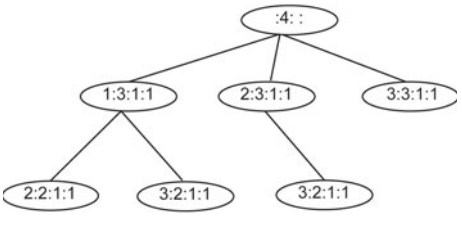


Fig. 2. Tree T_0 after processing first batch ($item$: $count$: TID: $batchCount$)

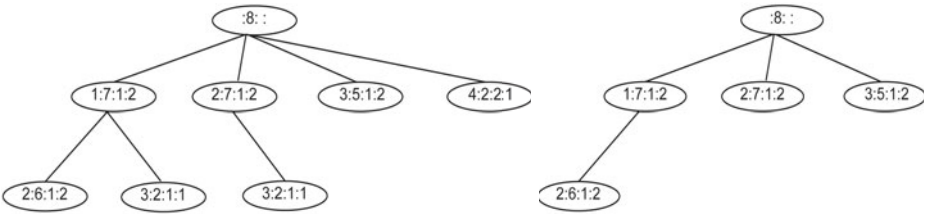


Fig. 3. Tree T_0 after second batch (left) and pruning (right)

nodes. For each node we compute $B^{'} = B - batchCount$. $B^{'} = 1$ for nodes corresponding to the following sequences $\langle 1,3\rangle, \langle 2,3\rangle, \langle 4\rangle$. For all remaining nodes $B^{'} = 0$. The next step is to apply final criterion. We have to prune nodes satisfying $count + B^{'}(\lceil \alpha L \rceil - 1) \leq B\epsilon L$. In this case, final criterion is satisfied if $count + B^{'} \leq 4$. Tree after pruning phase looks like in Fig. 3.
For all nodes in the tree $TID = 1$, $batchCount = 2$. The result set consists of sequences corresponding to nodes satisfying following inequality:

$$count \geq (N\sigma) - (TID - 1)\epsilon L - (N - (batchCount + TID - 1)L)\alpha$$

For the setting of this example, we have the condition: $count \geq 6$. Thus we obtain the following frequent sequential patterns: $\{\langle 1\rangle\colon 7, \langle 2\rangle\colon 7, \langle 1,2\rangle\colon 6\}$. All true patterns are in the result set of sequences and there are no false patterns. If we use the final criterion SS-BE, the final inequality to satisfy is $count \geq N(\sigma - \epsilon) = 8 \cdot (0,75 - 0,5) = 2$. In the result one additional false frequent pattern sequence $\langle 3\rangle\colon 5$ would appear.

4 Further Extensions: SS-LC and SSLC2

SS-BE algorithm uses 3 support thresholds. *minSupp* is the main support threshold, that is used to identify frequent sequential patterns. *sigSupp* is the significance threshold, that is used to identify almost frequent sequences. Sequence that at the beginning of exploration has a little support may appear at the end as a frequent sequential pattern. Support of almost frequent sequences written in the tree contributes to the improvement of count accuracy. *batchSupp* is a support threshold used in batch processing in order to improve even more count accuracy. Size of the tree is increased, because of this additional sequences, but during pruning phase all nodes with low support are deleted. This mechanism causes that between pruning phases the size of the tree is increasing.

In SSLC (Stream Sequence miner with Lossy Counting), sequences that have support between *batchSupp* and *sigSupp* are continuously inserted into and pruned from the tree. This goal was achieved thanks to new $PrefixSpan$ criterion using T_0 tree. The $PrefixSpan$ criterion looks as follows:

The criterion $PrefixSpanSSLC$**:** *The item is considered as frequent if:*

$$count \geq psminSupp \vee item \in T_0,$$

where psminSupp = sigSupp. The fact that path corresponding to the sequence composed of prefix and item exists in the tree is denoted as item $\in T_0$.

The pruning criterion SS-LC: *The entire subtree rooted at a given node should be deleted if following inequality holds:*

$$count \leq sigSupp(batchID - TID + 1)L.$$

The final criterion SS-LC: *Sequences corresponding to nodes from* T_0*, that satisfy inequality provided below are output.*

$$count > (N\sigma) - (TID - 1)\epsilon L.$$

It is important that despite of changes an attribute of guaranteeing outputs of all sequential patterns has been maintained.

Theorem 2. *SS-LC algorithm output all true sequential patterns.*

Proof: The only source of undercounting are packages of data that were processed before TID batch. If node corresponding to sequence does not exists in the tree, then can be assumed that $TID = N/L + 1$. Suppose that sequence S is pruned n times just after processing batches $B_{f_1}, B_{f_2}, ...B_{f_n}$. At every time that node is pruned, the undercounting increases. The size of undercounting according to the pruning criterion is less or equal to $(B_{f_i} - B_{f_{i-1}})\epsilon L$. The sum of undercounting caused by pruning is less or equal to $\sum_{i=1}^{n}(B_{f_i} - B_{f_{i-1}})\epsilon L = (TID - 1)\epsilon L$, where $B_{f_0} = 0$. □

This schema is very close to the idea of *LossyCounting* that was an inspiration of SS-BE. That why the name of this algorithm is SS-LC (Stream Sequence miner Lossy Counting). The new schema of exploration has following features:

- The usage of increased support threshold in packages processing results in:
  - reduces cost of computation and decreases size of tree T_0,
  - reduces memory usage,
  - reduces count accuracy of sequences which have little support and do not exist in the tree.
- The usage of tree in the criterion $PrefixSpan$ results in:
  - increases count accuracy of sequences that exist in the tree and have little support in some packages.
  - increases the cost of computation, because the algorithm counts the exact number of sequence occurrences which exist in the tree but have little support in the batch.
- increased memory usage and increased complication of the algorithm is the effect of using tree in the $PrefixSpan$ criterion
- elimination of *batchCount* results in the decreasing of memory usage
- simpler pruning criterion contributes to the higher performance

Experiments performed on synthetic data have proven the increased performance and less memory usage of new algorithm. The initial accuracy of SS-BE was not only maintained but even increased. The real data that is explored usually has

approximately the normal distribution. This is the reason why there is a lot of sequences having support between *batchSupp* and *sigSupp*. Exclusion of these sequences in exploration contributes to the increase of performance and to the less memory usage, which in fact does not have influence on accuracy.

Some test results for SS-LC, which are showing the more accurate results against SS-BE2, and also willingness to loss of performance, suggest that there is no need for the precise count of sequence occurrence which are in a tree. SS-LC2 algorithm is an attempt of looking for some further improvements.

The criterion $PrefixSpanSSLC2$**:** *The item satisfies the criterion if:*

$$count \geq psminSupp1 \vee (count \geq psminSupp2 \wedge item \in T_0),$$

where $psminSupp1 = sigSupp$*, and* $psminSupp2 = batchSupp$*.*

The pruning criterion SS-LC2: *The entire subtree rooted at a given node should be deleted if following inequality holds:*

$$count + (N/L - (TID + b - 1))(\lceil L\alpha \rceil - 1) \leq \epsilon(N/L - TID + 1)L,$$

where $b = batchCount$

The final criterion SS-LC2: *In the result set we output sequences corresponding to the nodes from a* T_0 *that satisfy this inequality* ($b = batchCount$)*:*

$$count > (N\sigma) - (N - (b + TID - 1)L)\alpha(TID - 1)\epsilon L.$$

The changes presented above had led to the decrease of the count accuracy. SS-LC2 is definitely less precise in this aspect than SS-BE2, but what is important is that it still guarantees to output all true sequential patterns.

Lemma 2. *Counted value for every node* n *during pruning phase*

$$U = count + (N/L - (TID + batchCount - 1))(\lceil \alpha L \rceil - 1)$$

is an upper bound for true number of occurrences of sequence S *in* $(N/L - TID + 1)$ *data batches that precedes pruning phase.*

Proof: The number of batch that introduced a node into the tree is stored in TID parameter. $(N/L - TID + 1)$ is the number of batches processed by the algorithm from the moment of sequence insertion until the last processed batch. Lets consider batches elapsed since last sequence insertion to the tree. Hence *batchCount* is the number of batches that contribute to the count of the sequence. The *count* parameter is used to store the sum of counted occurrences. The number of batches where occurrences were not counted equals $N/L - (TID + batchCount - 1))$. Occurrences in these batches have not been counted, due to the fact that sequence appeared less then αL times. Thus, the number of occurrences in this batches is limited by $(\lceil L\alpha \rceil - 1)$. Thus, $U = count + (N/L - (TID + batchCount - 1))(\lceil L\alpha \rceil - 1)$ is the upper bound of sequence count in $(N/L - TID + 1)$ batches preceding pruning phase. □

Theorem 3. *SS-LC2 algorithm output all true sequential patterns.*

Proof: Lets consider sequence S. Suppose that it was pruned from the tree just after the packages with the numbers $B_{f_1}, B_{f_2}, ...B_{f_n}$. Let $TID_{i_1}, TID_{i_2}, ...TID_{i_n}$ be the numbers of the corresponding packages that inserted sequence into the tree. Let $a_1 = (TID_{i_1} - 1), a_2 = (TID_{i_2} - B_{f_1} - 1), ..., a_n = (TID_{i_n} - B_{f_{n-1}} - 1), a_{n+1} = (TID - B_{f_n} - 1)$. Let $b_1 = (B_{f_1} - TID_{i_1} - 1), ..., b_n = (B_{f_n} - TID_{i_n} - 1)$. According to Lemma 2, the number of occurrences of sequence S in packages TID_{i_j} to B_{f_j} $(j \in 1, ...n)$ is less or equal to $b_j \epsilon L$. According to the $PrefixSpan$ criterion, the count of sequence S in packages with numbers from 1 to $(TID_{i_1} - 1)$, and from $(B_{f_1} + 1)$ to $(TID_{i_2} - 1)$ .... and from $(B_{f_{n-1}} + 1)$ to $(TID_{i_n} - 1)$, and from $(B_{f_n} + 1)$ to $(TID - 1)$ inclusively is less or equal respectively $a_1 \epsilon L, ..., a_{n+1} \epsilon L$. The sum of undercounting from first $(TID - 1)$ batches is less or equal to $\left(\sum_{i=1}^{n+1} a_i + \sum_{j=1}^{n} b_n\right)\epsilon L = ((TID_{i_1} - 1) + (B_{f_1} - (TID_{i_1} - 1)) + ((TID_{i_2} - 1) - B_{f_1}) + ... + (B_{f_n} - (TID_{i_n} - 1)) + ((TID - 1) - B_{f_n}))\epsilon L = (TID - 1)\epsilon L$. The packages from last $(N/L - TID + 1)$, where count of a sequence S is less or equal to αL, are the second and the last source of undercounting. The exact number of these packages is $(N/L - TID + 1 - batchCount)$. Thus, the sum of undercounting is less or equal to $(N/L - TID + 1 - b)L\alpha = (N - (TID + b - 1)L)\alpha$, where $(b = batchCount)$, what ends our proof. □

5 Experimental Results

Algorithms: SS-BE, SS-BE2, SS-LC, SS-LC2, PrefixSpan were implemented in the architecture in accordance with [10] in JAVA language. Experiments were conducted on the syntactic data received from the use of IBM generator [1]. Tests were carried out on Intel Core 2 Duo $1,8GHz$ $2GB$ RAM, where 512MB RAM were allocated for JVM. The goal of this experiment was to show advantage of SS-LC, SS-LC2, and SS-BE2 over SS-BE algorithm. SS-BE2 algorithm differs from the prototype in terms of changed final criterion. Three following methods of measuring the accuracy have been used:

- percentage of true patterns in the result set ($TruePatternsAccuracy$)
- percentage of patterns with exact count ($PerfectCountAccuracy$)
- average count accuracy of undercounted patterns ($CountAccuracy$)

Data for tests were generated by means of IBM data generator. We select the data sets with parameters "$I01\_L15\_P10$", i.e. 100 different items, approximately 15 items in each sequence and average maximal length of patterns is equal to 10. We also distinguish data that had such number of sequences: "$T200$", "$T400$", "$T600$", "$T800$". Firstly, the threshold of minimal support has been selected, which was arbitrarily set to 8%, thus $minSupp = 0,08$. After that, we check the influence of data batch size ($bufferSize$) to performance and memory usage. Small size of batch apart from the decrease of memory usage resulted in

[1] IBM Quest Market-Basket Synthetic Data Generator. http://www.almaden.ibm.com/cs/quest

the loss of performance and accuracy. This parameter has been determined for 4000 sequences in a batch ($bufferSize = 4000$). Interval of pruning phase was arbitrarily determined for 4 ($pruning\_period = 4$). The remaining parameters were being changed during tests in order to show characteristics of algorithms.

5.1 Experiment 1 (Variable Support Threshold)

Accuracy. The tests have been conducted in order to check how accuracy of algorithms behaves in a relationship with the usage of *batchSupp* and *sigSupp* parameters. The experiments were conducted in many data sets but we present the result for "$I01\_L20\_P10$" data as an example. Second test has been conducted only for SS-BE2 and SS-LC algorithms. Fig. 4(a) presents a graph showing accuracy $TruePatternsAccuracy$ from the first test.

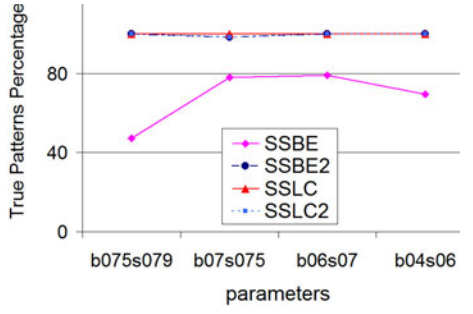
The graph demonstrate the weakness of SS-BE algorithm, which has significant problem with false sequential patterns in result rest (accuracy $40 - 80\%$)). The improvement of final criterion in SS-BE2 has shown its effectiveness by achieving almost perfect results. SS-LC2 achieved similar results as SS-BE. SS-LC obtained ideal result - 100 percent of accuracy in all tests. It can be observed that the accuracy of SS-LC is almost ideal, and, moreover, stable even during the changes of support threshold. SS-BE2 is weak in the situation where *sigSupp* and *batchSupp* thresholds are close to *minSupp*. The increase of a number of patterns negatively influences accuracy of SS-BE, SS-BE2, SS-LC2 algorithms. The comparison of accuracy $PerfectCountAccuracy$ has been shown in Fig. 4(b). If *batchSupp* and *sigSupp* are set close to the *minSupp* it creates huge difficulties for SS-BE, SS-BE2 and SS-LC2 algorithms. For parameters: $sigSupp = 0,079$ and $batchSupp = 0,075$, result of $PerfectCountAccuracy$ equals about $68,5\%$, and is about 9% lower than for SS-LC.

Performance. Figure 4(d) presents the graphs of performance. It can be seen that the increasing distance of other thresholds to *minSupp* has an influence on the performance drop of all algorithms. Performance of SS-LC and SS-LC2 decreases substantially slower than performance of SS-BE and SS-BE2 (see Fig. 4(c)). Advantage of SS-LC over SS-BE increases together with the number of patterns.

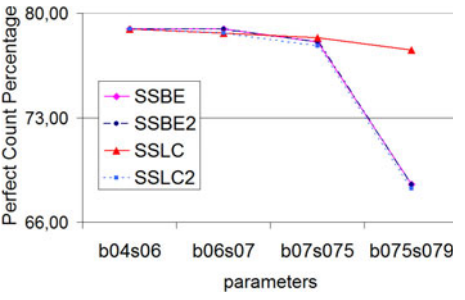
Memory usage. Graphs presenting memory usage are shown in Fig. 4(f). SS-LC and SS-LC2 show less memory usage than SS-BE and SS-BE2. If there is a small distance of other thresholds to *minSupp*, memory usage in SS-BE and SS-BE2 increases unexpectedly. The occurrence of varying tree size that was significantly reduced in SS-LC and SS-LC2 might be the reason for obtaining such results.

5.2 Experiment 2 (Variable Number of Patterns)

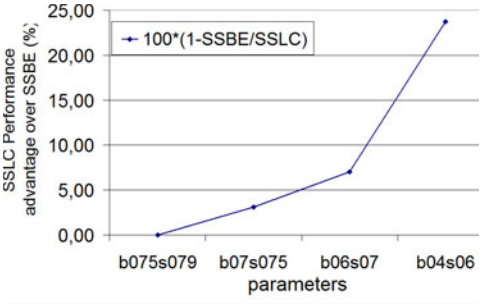
In the second experiment it was checked how algorithms behave if the average length of sequence and the corresponding number of patterns changes. Tests was performed on the following files "$T200\_I01\_L10\_P5$", "$T200\_I01\_L15\_P10$",



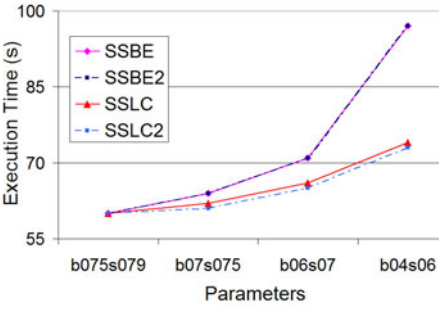
(a) TruePatternsAccuracy



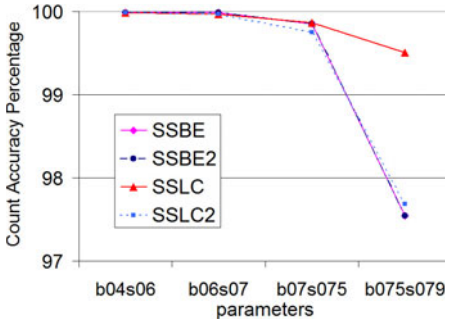
(b) PerfectCountAccuracy



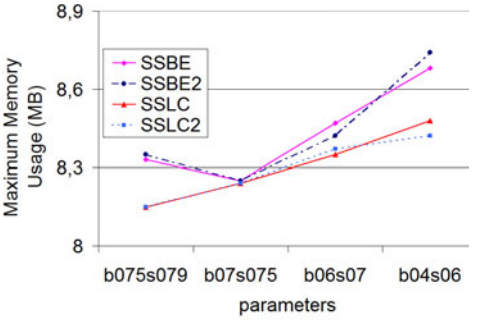
(c) SS-LC vs SS-BE - 359 patterns



(d) Performance



(e) CountAccuracy

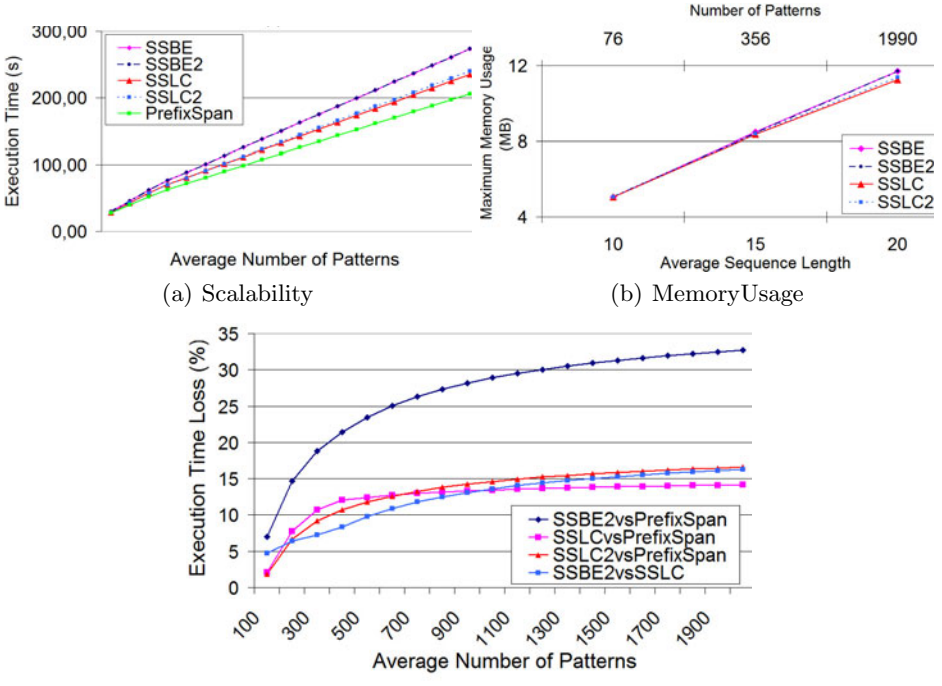


(f) Memory usage

Fig. 4. Results of Experiment 1

"$T200\_I01\_L20\_P15$". Thresholds was set as follows $batchSupp = 0,06$, $sigSupp = 0,07$, $minSupp = 0,08$. If we set the value of minimum support threshold then average length increase is related to the geometrically increasing number of patterns. This dependence looks as follows:

- $I01\_L20\_P15$ around $2K$ of patterns for $(minSupp = 0,08)$
- $I01\_L15\_P10$ around 350 of patterns for $(minSupp = 0,08)$
- $I01\_L10\_P5$ around 75 of patterns for $(minSupp = 0,08)$



(a) Scalability (b) MemoryUsage

(c) Performance

Fig. 5. Experiment 2: T200I01, $\delta = 0.08, \epsilon = 0.07, \alpha = 0.06$

Performance. It is noticeable that the performance is dramatically decreasing when average length of sequences is increasing. It is connected with the geometrically increasing number of patterns in the data source. Scalability of performance with respect to the number of patterns is linear, what is presented in the figure 5(a). The advantage of SS-LC and SS-LC2 over SS-BE and SS-BE2 in this test is presented in the figure 5(c). The advantage of SS-LC over SS-BE2 is increasing upon to increasing number of patterns.

Memory usage. Memory usage of stream algorithms is much less than memory usage of $PrefixSpan$. The comparison of memory usage of stream algorithms is illustrated in the figure 5(b). SS-BE and SS-BE2 are slightly worse than two others and loss is increasing upon to increasing number of patterns.

Acknowledgement. This work is partially supported by the National Centre for Research and Development (NCBiR) under Grant No. SP/I/1/77065/10 by the Strategic scientific research and experimental development program: "Interdisciplinary System for Interactive Scientific and Scientific-Technical Information" and grants from Ministry of Science and Higher Education of the Republic of Poland N N516 077837.

6 Conclusions

Experiments confirmed the efficiency of the modifications of SS-BE. The improvement of final criterion used in the SS-BE2 algorithm has achieved the most spectacular results. It seems that the final criterion in the original version was overestimated. Moreover, the results of significant accuracy (Fig. 4(a)) are more then satisfactory. The next improvement in SS-LC increased the performance and accuracy o SS-BE2. Results are not so spectacular like for the final criterion but in many cases the improvement should be noticeable. It is also very important that in comparison to the SS-BE2 algorithm, the stability of efficiency and accuracy has increased. The sensitivity of the SS-LC to the changes of mining parameters and data characteristic has significantly decreased in comparison to SS-BE and SS-BE2. In comparison to SS-LC, the changes introduced in SS-LC2 did not result in a significant improvement. Tests results are showing that SS-BE2 and SS-LC algorithms are natural and appropriate modification to the idea of streaming data mining of sequential patterns used in the SS-BE.

References

1. Zhao, Q., Bhowmick, S.S.: Sequential pattern mining: A survey. ITechnical Report CAIS Nayang Technological University Singapore, pp. 1-26 (2003)
2. Srikant, R., Agrawal, R.: Mining Sequential Patterns: Generalizations and Performance Improvements. In: Apers, P.M.G., Bouzeghoub, M., Gardarin, G. (eds.) EDBT 1996. LNCS, vol. 1057, pp. 1–17. Springer, Heidelberg (1996)
3. Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., Hsu, M.: Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach. IEEE Trans. Knowl. Data Eng., 1424–1440 (2004)
4. Ezeife, C.I., Lu, Y., Liu, Y.: PLWAP Sequential Mining: Open Source Code. In: Proc. of the 1st Int. Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations (OSDM 2005), pp. 26–35. ACM, New York (2010)
5. Cheng, H., Yan, X., Han, J.: IncSpan: incremental mining of sequential patterns in large database. In: Proc. of the 10th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD 2004), pp. 527–532. ACM, New York (2004)
6. Giannella, C., Han, J., Pei, J., Yan, X., Yu, P.S.: Mining Frequent Patterns in Data Streams at Multiple Time Granularities. In: Kargupta, H., Joshi, A., Sivakumar, K., Yesha, Y. (eds.) Next Generation Data Mining, pp. 191–212. AAAI/MIT (2003)
7. Marascu, A., Masseglia, F.: Mining Sequential Patterns from Temporal Streaming Data. In: Proc. of the 1st ECML/PKDD Workshop on Mining Spatio-Temporal Data (MSTD 2005), pp. 1–13 (2005)
8. Ezeife, C.I., Monwar, M.: SSM: A Frequent Sequential Data Stream Patterns Miner. In: IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2007, pp. 120–126 (2007)
9. Mendes, L.F., Ding, B., Han, J.: Stream Sequential Pattern Mining with Precise Error Bounds. In: Proceedings of ICDM 2008, pp. 941–946 (2008)
10. Wojnarski, M.: Debellor: A Data Mining Platform with Stream Architecture. In: Peters, J.F., Skowron, A., Rybiński, H. (eds.) Transactions on Rough Sets IX. LNCS, vol. 5390, pp. 405–427. Springer, Heidelberg (2008)
11. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proc. of ICDE 1995, pp. 3–14 (1995)

Social Influence Modeling on Smartphone Usage

Masaji Katagiri[1,2] and Minoru Etoh[1,3]

[1] R&D Center, NTT DOCOMO, Inc.
Yokosuka, Kanagawa, 239-8536 Japan
katagirim@nttdocomo.co.jp, etoh@ieee.org
[2] Graduate School of Information Science and Technologies, Osaka University
Suita, Osaka, 565-0871 Japan
[3] Cybermedia Center, Osaka University
Toyonaka, Osaka, 560-0043 Japan

Abstract. This paper presents a probabilistic influence model for smartphone usage; it applies a latent group model to social influence. The probabilistic model is built on the assumption that a time series of students' application downloads and activations can be represented by individual inter-personal influence factors which consist of latent groups. To verify that model with its assumption, about 160 university students voluntarily participated in a mobile application usage monitoring experiment. Analysis could identify latent user groups by observing predictive performance against reduced dimensions of factor matrices with NMF. Proper dimension reduction is shown to significantly improve predictive performance, which implies a reduction in the over-fitting phenomenon. With this improvement, the model outperforms conventional collaborative filtering models and popularity models in perplexity evaluation. The results validate the model and its assumption as well as its usefulness.

Keywords: user influence, mobile application, recommendation, behavior prediction, latent structure analysis, matrix factorization, NMF.

1 Introduction

Smartphones are becoming popular devices that can replace traditional cellular phones globally. One of their features is that various applications are available through the public network. In the case of Android phones, there is the "Android Market" site where people can find and download applications. Since the sheer number of applications makes it difficult for users to find the appropriate software, it is crucial to provide an appropriate search function and/or recommendation system for applications. The challenge here is to recommend niche but appropriate applications to each users from 100s of thousand candidate applications.

Many studies have investigated various collaborative filtering (CF) methods for recommending items potentially interesting items to users in different systems [1] [15]. However, the filtering used is based on just the symmetric similarity

J. Tang et al. (Eds.): ADMA 2011, Part II, LNAI 7121, pp. 292–303, 2011.

among users or products commonly used or bought and ignores temporal order information.

A recent study shows asymmetric inter-personal influence may play an important role in understanding user adoption patterns and recommending items to users with significant potential in improving user satisfaction [11]. Unfortunately, it is difficult to obtain individual inter-personal influence factors.

With regard to influence factors, there are several models that simulate information diffusion through a network. A widely-used model is the independent cascade (IC), a fundamental probabilistic model of information diffusion [12] [7]. The IC model assumes that a social network structure is given and also requires diffusion probabilities associated with links to be specified as parameters. Several methods have been proposed to estimate diffusion probabilities by addressing the maximization problem [9] [14]. Those sophisticated models apply machine learning techniques and so are not scalable for realistic problem settings due to their computational complexity. Other than the IC model, Song proposed the information flow network to predict future adoption; it can be created from adoption history [20]. Kawamae proposed personal innovator degree, which assumes that influence on the future adoption decreases exponentially with time[11]. All influence-related methods published to date need to acquire inter-personal influence factors individually. This implies the cold-start problem for new users as well as computational scalability issues.

A noteworthy contribution of this paper is its proposal of a latent structure model on inter-personal influence; another is its confirmation that the latent structure can improve predictive performance significantly. Intuitively, we naturally form communities in various ways among people, and are often surprised by finding similar people to ourselves. Thus the authors expect inter-personal influence can be represented by some latent group structure. To validate the model and assumption, the authors conduct experiments with 160 students of Osaka University. Product purchase history and usage (i.e., studentfs activities on application download and activation) are modeled by a latent group model derived from implicit individual influence factors.

In order to acquire latent groups on inter-personal influence, the authors propose to apply non-negative matrix factorization (NMF) to the influence matrix derived from observations. NMF significantly reduces the over-fitting effect, and the results suggest that $N \times N$ relationships can be represented by $N \times r \times N$ group-wise relationships, where N represents the number of users and r represents the number of internally influential groups. In a comparison with conventional collaborative filtering models, the model offers the best predictive performance with regard to perplexity.

The following sections will detail model formulation, determining influence factors from observations, latent group acquisition, and the experiments conducted toconfirm the proposed method's usability in a real environment.

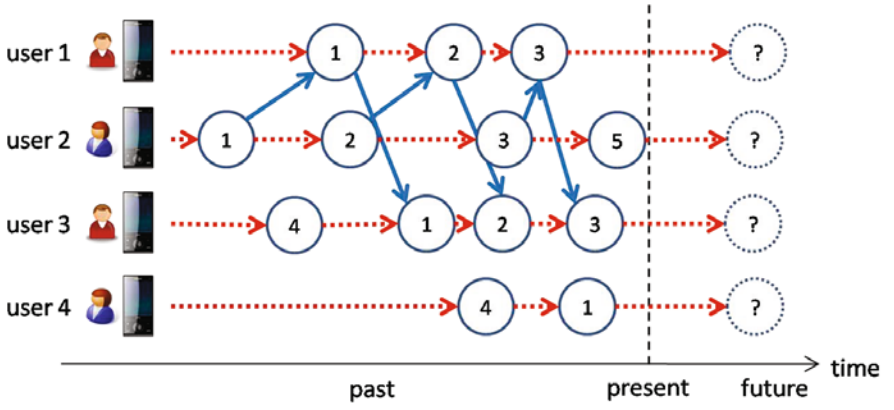


Fig. 1. Cascades in Application Usage

2 Modeling Inter-personal Influence

2.1 Asymmetric Relationship

Fig. 1 shows a simple example of the application adoption logs of 4 users (users 1, 2, 3, & 4) given a pool of 5 applications. Here, the horizontal axis depicts time. A circle with number denotes that the user adopted the corresponding application at that time. Obviously there are cascades on user-application relationships, which is a sequence of users who adopted an application (blue arrows). The goal is to predict future application adoption for each user (i.e. personalized recommendations). The conventional collaborative filtering technique suggests that user 1 is most similar to user 2 and user 3. Thus application 4 and 5 will be recommended equally by CF. This result is denied if we pay attention to the order of adoption; user 2 is a forerunner of user 1 and user 3 is a follower of user 1. Thus application 5 is more likely to be adopted by user 1. This clearly illustrates the importance of incorporating inter-personal influence in prediction.

2.2 Deriving Influence Factors from Observations

First, we define the influence factor model, which is similar to the model described in [8] as static model. However, we do not assume that the social network is given, since in reality it cannot be observed. Therefore we need to predict influence factors for all of users from observed application usage data.

Frequency-Based Model. Under this model, user u is assumed to have static influence probability $P_r(u \rightarrow v)$ for each user v. If user u invoked application d, anytime during a certain time period τ after invocation, he/she tries to influence his/her inactive (not yet invoked d) neighbor v. Here, parameter τ is assumed to be constant. Each attempt can be viewed as a Bernoulli trial. Thus the Maximum Likelihood Estimator of successful probability is the ratio of number of successful

attempts over the total number of trials. Hence, influence probability of u on v is estimated as;

$$P_r(u \to v) = \frac{\|A_{u \to v}\|}{\|A_u\|}. \tag{1}$$

where $\|A_u\|$ denotes the number of applications which user u invoked in the training set. $\|A_{u \to v}\|$ denotes the number of applications which user u invoked then v invoked in the training set.

Let $P_{inf}(d|u)$ be the joint influence probability for application d on user u from surrounding neighbors. In this paper, we assume that the probabilities of various friends influencing u are independent of each other. Hence, the joint probability $P_{inf}(d|u)$ can be defined as follows;

$$P_{inf}(d|u) = \left[1 - \prod_{u'} \{1 - P_r(u' \to u) Y(u', d)\}\right], \quad \text{and} \tag{2}$$

$$Y(u', d) = \begin{cases} 1 \text{ if user } u' \text{ has invoked application } d \text{ within } \tau \\ 0 \text{ otherwise} \end{cases}. \tag{3}$$

Entropy-Based Model. Taking niche items as the target to recommend, we introduce an entropy-based influence factor. Entropy can be considered as an indicator for rarity. In the model, we obtain the application entropy from its number of unique users. Hence, the proposed entropy-based influence probability $\widehat{P_r}(u \to v)$ is defined as follows;

$$\widehat{P_r}(u \to v) = \frac{\sum_{aA_{u \to v}} (-log\left(\frac{u_a}{U_{glob}}\right))}{\sum_{aA_u} (-log\left(\frac{u_a}{U_{glob}}\right))}. \tag{4}$$

where U_{glob} denotes the possible total number of users for applications (Android applications) and u_a denotes the number of unique users of application a. It is difficult to know the true values of u_a and U_{glob}. Instead of the true value of u_a, the authors employ the approximate number of downloads from the "Android Market". For U_{glob}, the value of 2,000,000 was tentatively applied in the experiments. $P_{inf}(d|u)$ has no need to be modified if $\widehat{P_r}(u' \to u)$ is applied instead $P_r(u' \to u)$.

The authors have performed pre-evaluation experiments comparing the frequency-based model to the entropy-based model. The results show that the latter has slightly better performance, and so the rest of this paper uses the entropy-based model.

The effects on the parameter τ were also examined in preliminary experiments. The results showed that, within the limits of the period examined in the experiments, increasing parameter τ yields better performance. Therefore, all experiments in this paper are carried out under the condition of $\tau = \infty$.

Let user influence matrix R be a $U_{exp} \times U_{exp}$ matrix which has the value of $P_r(u \to v)$ as the intersection of row u and column v, where U_{exp} denotes the number of contributing subjects in the experiment.

2.3 Latent Group Model

Acquired user influence matrix R is assumed to contain the latent structure of inter-personal relationship. Assuming k latent groups, influence matrix R is transformed into approximated matrix $\widehat{R}$ with reduced dimension k. The prediction will be performed using $\widehat{R}$ instead of R.

Motivated by latent semantic analysis (LSA) [4], originally developed for text processing, extensive analysis was conducted on the latent structure of those relationships using non-negative matrix factorization (NMF) technique [16] as an equivalent to probabilistic LSA [10]. NMF decomposes one non-negative matrix X ($n \times m$ matrix) into two non-negative matrices A ($n \times k$ matrix) and B ($k \times m$ matrix) with reduced dimension k ($k \ll n, m$) as a result of factorization, such that $X \simeq AB$. Here, let us call matrix $\widehat{X} = AB$ ($n \times m$ matrix) the approximated matrix of X with reduced dimension k. NMF has a unique feature, it keeps values non-negative through decomposition as a constraint, which contributes to achieving realistic solutions in certain applications. Although negative user influence is possible such as recommending his/her friends not to use some applications, the authors believe that they are far less major cases and negligible. Thus, constraining the influence matrix to non-negative values is considered promising than other methods such as SVD. A later section compares NMF to SVD as a decomposer.

By observing the prediction performances of approximated influence matrices with different reduced dimensions, we could find the most likely dimension of the best dimensionality of the latent structure assuming the existence of performance degradation due to over-fitting. However, there are several issues we must be careful of. NMF algorithms solve problem iteratively and thus produce local optimum results rather than the global optimum, i.e., different initial values may lead to different solutions. There are several algorithms on NMF that address optimization, which may produce different solutions. Although NMF provides capabilities to customize optimization direction to suit the application's characteristics, difficulty still occurs since it is unknown which characteristic on the decomposed matrices (e.g. sparseness, smoothness, etc.) leads to better predictive performance. To cope with above issues, stochastic approach was taken. Approximate influence matrices with reduced dimensions were calculated using multiple NMF algorithms with different parameter settings if applicable. An evaluation of item adoption prediction was conducted by measuring the perplexity of the produced matrices. The results are shown in the later section. Table 1 shows the NMF algorithms used in the experiments. All NMF computations were performed using the package NMF [6] in R [19].

3 Data and Experiments

3.1 Collected Data

The authors conducted experiments to monitor usage of smartphones in collaboration with Osaka University. 160 university students voluntarily joined the

Table 1. NMF Algorithms

#	Reference	Description
1	[3]	Standard NMF, based on Kullbach-Leibler divergence
2	[17]	Standard NMF, based on Euclidean distance
3	[18]	Non-smooth NMF based on Kullbach-Leibler divergence
4	[2]	Modified version of [17] based on Euclidean distance
5	[21]	Pattern-Expression NMF based on Euclidean distance
6	[13]	Alternating Least Square (ALS) approach

monitoring trial. Each student was given a smartphone, an Xperia®, onto which, with informed consent on data collection and its use for research purposes, data collection software was installed.

Each application invocation was captured and recorded with timestamp by monitoring software. The log data were anonymized and collected on a server through 3G networks. Each log record contained timestamp, anonymized user-id, and the component name of the application invoked.

The first invocation record for each application on each phone was extracted. We split the data set into training set and test set by timestamp. The training set consisted of records made in a 89 day period (February – April 2011) and the test set consisted of records made inthe 31 day period (May 2011) right after the training set. We excluded records on applications that had less than 3 unique users in the training set. This yielded 3383 records (155 users and 291 applications) in the training set and 249 records (98 users and 116 applications) in the test set.

3.2 Evaluation Process

Model parameters were estimated from the training set. Application invocations during the test period were predicted using the estimated model parameters and the usage history (the training set itself). The prediction quality was measured by the observed test set data. Here, performance was evaluated by test set perplexity [5], which is a standard measure for language model evaluation and corresponds to MAE/RMSE in the probabilistic domain. Conventional user-based and SVD-based collaborative filtering, and simple prediction based on popularity were tested as baseline algorithms.

3.3 Latent Structure by Matrix Factorization

Fig. 2 plots the best perplexity predictive performance results from approximated influence matrices $\widehat{R}$ with reduced dimensionality by NMF. We can observe a slight decrease in perplexity (i.e., a slight performance improvement) at the remarkably small dimension of 6. This implies that the inter-personal relationship

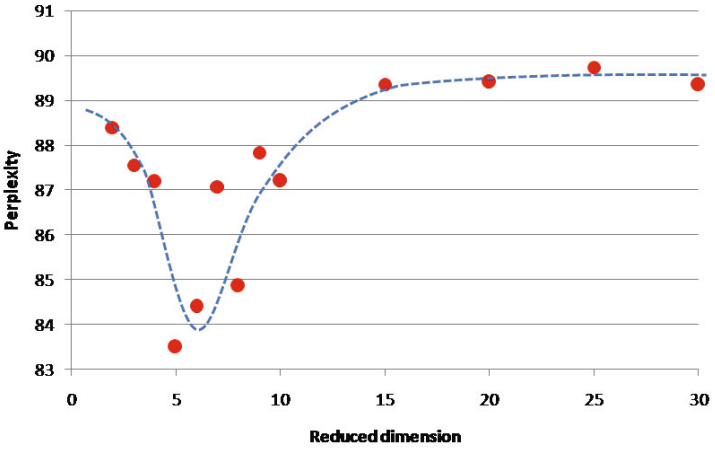


Fig. 2. Perplexity on reduced dimensionality influence matrix yielded by NMF

has a structure containing 6 latent components (or groups) and each userfs characteristic can be represented by the appropriately weighted coefficients of these components.

Fig. 3 plots the perplexity predictive performance results from the approximated influence matrix yielded by SVD derived from the same data set used for Fig. 2. Regardless of the dimension, NMF demonstrates better perplexity than SVD. In the case of SVD, we observe worse perplexity over a wide range of reduced dimensionality than original influence matrix (which has dimension of 98.2635). It is notable that $\widehat{R}$ by NMF offers better performance than the original R, even if there is no guarantee that the approximation leads to an improvement in predictive performance, supposedly as a result of the non-negativity constraint.

Fig. 4 provides heatmaps to visualize the decomposed influence matrices at the lowest perplexity (dimension $k = 5$). The values of the matrix elements are normalized for each user where the sum of elements is taken as 1, and are expressed by color (red color means high value, light yellow shows small value). Fig. 4(a) shows coefficient matrix for the influence patterns and Fig. 4(b) shows the basis matrix representing influence patterns. Vertical (horizontal) axis represents individual users in the coefficient (basis) matrix. The vector of each user on the coefficient matrix denotes his/her influence pattern,i.e. who are influenced by him/her. On the other hand, the vector of each pattern (1 to 5 in this case) on the basis matrix shows the destination of influence. For example, pattern 1 means a strong influence on the 26 students in the left most columns of the basis matrix. Please note that those matrices are inherently bi-directional. The matrix of Fig. 4(b) can be considered as a coefficient matrix for being-influenced patterns, in such a case, Fig. 4(a) can also be considered as a basis matrix.

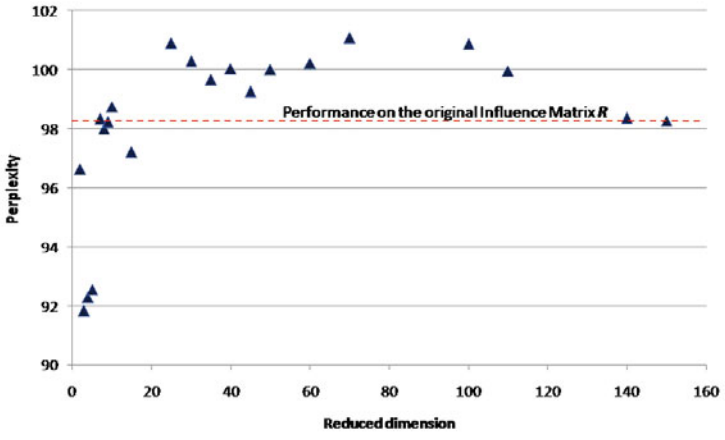
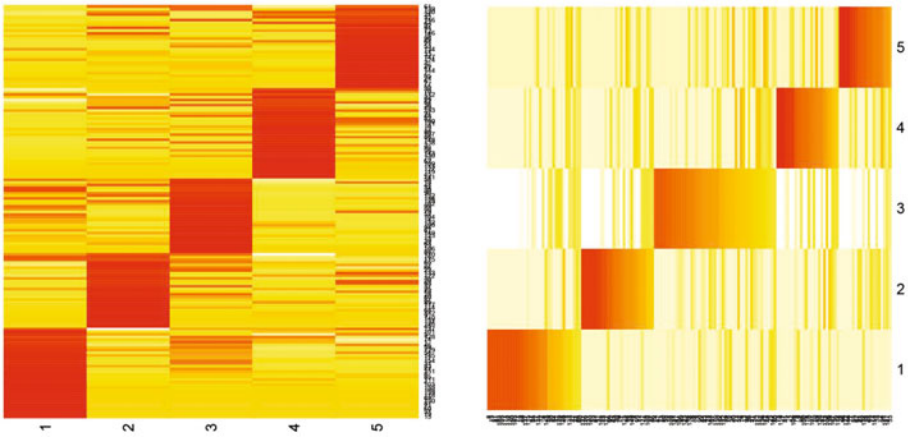


Fig. 3. Perplexity on reduced dimensionality influence matrix yielded by SVD



(a) Coefficient matrix (b) Basis matrix

Fig. 4. Heatmaps of decomposed influence matrices ($k = 5$)

3.4 Predictive Performance Comparison

Fig. 5 shows a comparison of perplexity performance.

In Fig. 5, Latent Str. denotes the latent group models discussed in the previous section. CF(cos) and CF(cos:k-nn) denote conventional user-based CF on cosine similarity with and without k-nearest-neighbors filtering. CF(svd) denotes SVD-based CF. Popularity(long) and Popularity(short) denote simple predictions based on popularity degrees, which are measured by the numbers of unique users during sampling periods. As sampling periods, "long" means the whole period of the training set. "short" means the last 1 week of the training set. Both

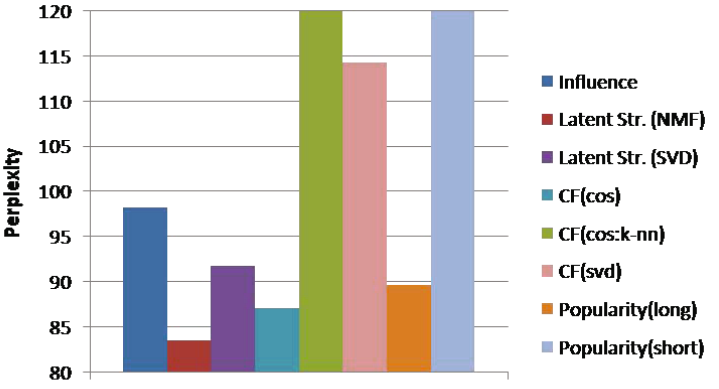


Fig. 5. Performance on Perplexity

are just before the test set begins. For the latent group models, the best performances among the different dimensions are shown.

As we can see, the proposed latent group model yielded by NMF demonstrates the best performance. This result validates the proposed model and its usefulness for recommendations.

4 Discussion

4.1 Influence / Influencee Factor on Individuals

Identifying degree of influence on each individual user is fruitful for viral marketing. The degree of individual influence can be defined as follows;

$$Influencer_u = \frac{\sum_v (\widehat{P_r}(u \to v)}{U_{exp} - 1}, \tag{5}$$

$$Influencee_u = \frac{\sum_v (\widehat{P_r}(v \to u)}{U_{exp} - 1}. \tag{6}$$

Fig. 6 shows a scatter plot of influencer versus influencee factors calculated from the approximated influence matrix $\widehat{R}$ with the best predictive performance. Each point corresponds to an individual user. Dashed lines show median values of the distributions. Arrows are drawn for user pairs whose influence factor exceeds a given threshold. It is natural that most arrows going the lower right to upper left direction. Users in the lower right segment are considered to be early adopters and/or opinion leaders. Similarly, users in the upper left segment are considered to be followers and/or laggards.

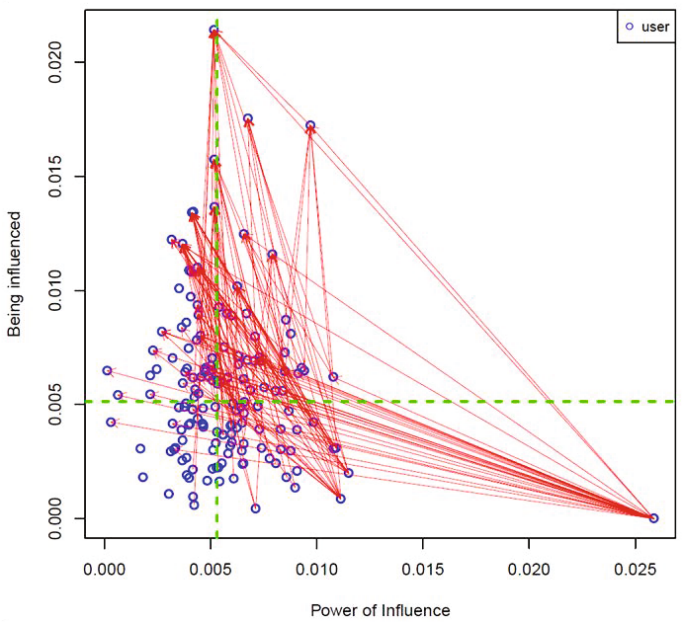


Fig. 6. Influencer / Influencee Degree on Individuals

5 Conclusion

This paper described a probabilistic model, based on latent groups, for extracting social influence among users in the adoption of smartphone applications. The model assumes that a time series of students' application downloads and activations can be represented by individual inter-personal influence factors, which consist of latent groups. Experiments verified the model and its assumption. Latent user groups were experimentally identified by observing predictive performance while reducing the dimensionality of the influence matrix. Moreover, the model successfully predicted students' application downloads from their past activities; it showed better perplexity than conventional collaborative filtering models and popularity models.

Further studies will explore the employment of more sophisticated diffusion models with diffusion rate to achieve better performance. Scalability and dynamic performance are indispensable issues to be explored.

References

1. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering 17, 734–749 (2005)

2. Badea, L.: Extracting gene expression profiles common to colon and pancreatic adenocarcinoma using simultaneous nonnegative matrix factorization. In: Altman, R.B., Dunker, A.K., Hunter, L., Murray, T., Klein, T.E. (eds.) Pacific Symposium on Biocomputing, pp. 267–278. World Scientific (2008)
3. Brunet, J.P., Tamayo, P., Golub, T.R., Mesirov, J.P.: Metagenes and molecular pattern discovery using matrix factorization. PNAS 101(12), 4164–4169 (2004)
4. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. Journal of the American Society for Information Science 41(6), 391–407 (1990)
5. Furui, S.: Speech and speaker recognition evaluation. In: Dybkjœr, L., Hemsen, H., Minker, W., Ide, N. (eds.) Evaluation of Text and Speech Systems, Text, Speech and Language Technology, vol. 37, pp. 1–27. Springer, Netherlands (2007)
6. Gaujoux, R., Seoighe, C.: A flexible R package for nonnegative matrix factorization. BMC bioinformatics 11(1), 367+ (2010)
7. Goldenberg, J., Libai, B., Muller, E.: Talk of the Network: A Complex Systems Look at the Underlying Process of Word-of-Mouth. In: Marketing Letters, pp. 211–223 (August 2001)
8. Goyal, A., Bonchi, F., Lakshmanan, L.V.: Learning influence probabilities in social networks. In: Proceedings of the Third ACM International Conference on Web Search and Data Mining, WSDM 2010, pp. 241–250. ACM, New York (2010)
9. Gruhl, D., Guha, R., Liben-Nowell, D., Tomkins, A.: Information diffusion through blogspace. In: Proceedings of the 13th International Conference on World Wide Web, WWW 2004, pp. 491–501. ACM, New York (2004)
10. Hofmann, T.: Probabilistic latent semantic indexing. In: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 1999, pp. 50–57. ACM, New York (1999)
11. Kawamae, N., Sakano, H., Yamada, T.: Personalized recommendation based on the personal innovator degree. In: Proceedings of the Third ACM Conference on Recommender Systems, RecSys 2009, pp. 329–332. ACM, New York (2009)
12. Kempe, D., Kleinberg, J., Tardos, E.: Maximizing the spread of influence through a social network. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2003, pp. 137–146. ACM, New York (2003)
13. Kim, H., Park, H.: Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. Bioinformatics 23(12), 1495–1502 (2007)
14. Kimura, M., Saito, K., Nakano, R.: Extracting influential nodes for information diffusion on a social network. In: Proceedings of the 22nd National Conference on Artificial Intelligence, AAAI 2007, vol. 2, pp. 1371–1376. AAAI Press, Vancouver (2007)
15. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer 42(9), 30–37 (2009)
16. Lee, D.D., Seung, H.S.: Learning the parts of objects by non-negative matrix factorization. Nature 401(6755), 788–791 (1999)
17. Lee, D.D., Seung, H.S.: Algorithms for Non-negative Matrix Factorization. In: Leen, T.K., Dietterich, T.G., Tresp, V. (eds.) Advances in Neural Information Processing Systems, vol. 13, pp. 556–562. MIT Press, Cambridge (2001)
18. Pascual-Montano, A., Carazo, J., Kochi, K., Lehmann, D., Pascual-Marqui, R.D.: Nonsmooth nonnegative matrix factorization (nsnmf). IEEE Transactions on Pattern Analysis and Machine Intelligence 28, 403–415 (2006)

19. R Development Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2008)
20. Song, X., Tseng, B.L., Lin, C.Y., Sun, M.T.: Personalized recommendation driven by information flow. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2006, pp. 509–516. ACM, New York (2006)
21. Zhang, J., Wei, L., Feng, X., Ma, Z., Wang, Y.: Pattern expression nonnegative matrix factorization: Algorithm and applications to blind source separation. Computational Intelligence and Neuroscience 2008 (2008)

Social Network Inference of Smartphone Users Based on Information Diffusion Models

Tomonobu Ozaki[1] and Minoru Etoh[1,2]

[1] Cybermedia Center, Osaka University,
1-32 Machikaneyama Toyonaka, Osaka 560-0043, Japan
tozaki@dcm.cmc.osaka-u.ac.jp
[2] NTT DOCOMO R&D Center,
3-6 Hikarino-oka Yokosuka, Kanagawa 239-8536, Japan
etoh@ieee.org

Abstract. In this paper we propose models for inferring a social network of smartphone users. By applying the concept of information diffusion models to the log of application executions in smartphones, strength of relationships among users will be estimated as an optimization problem. Functions on time difference and application significance are employed to capture user behavior precisely. In addition, affiliation information of users is effectively utilized as an exogenous factor. Experimental results using 157 of smartphone users indicate that the proposed model outperforms naive methods and infers a social network appropriately. Especially, the model succeeds in capturing the important relations in user communities accurately.

Keywords: social network inference, log analysis, information diffusion model, smartphone.

1 Introduction

In this paper, we discuss the problem of inferring a social network of smartphone users by using log data of application executions. Social network discovery with a subsequent network analysis expects to provide key information for wide variety of applications such as advertisement, recommendation and viral marketing.

In order to infer a social network accurately, we focus on the process of *information diffusion* on mobile applications. Many sophisticated models for information diffusion have been proposed recently in the area of social network analysis [10,7,9,20]. In diffusion models, the behavior of a person or user is assumed to be affected by her neighbors in a social network. For example in Fig. 1(left), a user u decides whether or not she adopts items a_1 and a_2 based on the strength of influence $w_{u_i,u}$ from neighboring four users $u_i\,(i = 1, \cdots, 4)$. The structure of network is effectively utilized to model the diffusion process or user behavior precisely. While an accurate social network drives good performance of the models, we cannot always expect to prepare accurate and precise networks. Although the information diffusion models are not directly applicable to estimate

J. Tang et al. (Eds.): ADMA 2011, Part II, LNAI 7121, pp. 304–317, 2011.

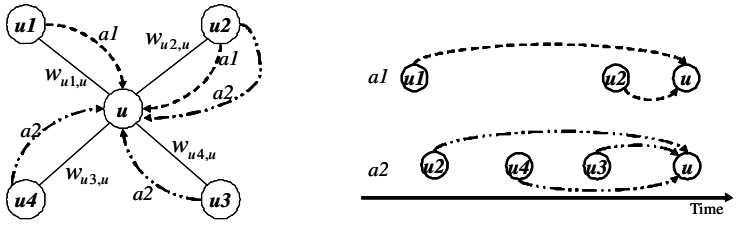


Fig. 1. Information propagation on a network (left) and information cascades (right)

the network structures, we employ their concepts and develop a new method for inferring social networks.

In this paper, we consider the information cascades of mobile application executions. As similar to the information diffusion models, we regards that the behavior of a user is affected by early adopters of each mobile application. For example in Fig. 1(right), u's behaviors on item a_2 is determined by early adopters u_i $(i = 2, \cdots, 4)$. However, in the actual situation, every early adopter does not make equal impacts on the behavior of other users. To alleviate the negative effects of this inconsistency, we introduce two factors for adjusting the effects of early adopters. One is the time difference of application execution and the other is the significance of application itself. We assume that an application has different degree of influential effects in different situations, and the influential effects decrease with time. By adjusting the influential effects with two factors, we build a model of user behavior on mobile application executions, and use it to infer a social network. Beside local influences of users, an exogenous factor such as affiliation of users is also incorporated into the model.

The rest of this paper is organized as follows. In section 2, we propose models of user behavior based on information diffusion to infer a social network. Some extensions to the basic model are also discussed. After describing related work in section 3, experimental results are shown in section 4. Finally, we conclude the paper and describe future work in section 5.

2 Inference of a Social Network Based on Information Diffusion

2.1 Preliminaries

In this research, we use a dataset collected by android-based smartphones from February to April 2011 with 157 participants, who are undergraduate students in Osaka University. We install custom made software in each smartphone to capture the logs of application executions and phone calls. Since our focus is to infer a social network based on information diffusion of applications, we exclude logs of applications which are used by less than five participants during the experiment period. Besides the application and call logs, we employ the department to which she belongs as well as the year when she entered the university as

basic information on participants. In addition, six communities of participants were identified by a questionnaire. Each community has 50, 50, 26, 13, 10, and 8 members, respectively. We employ the community information as a ground truth to evaluate our proposal.

For the formal discussion, basic notations and definitions used throughout the paper are given. An application log L is a series of triplets $I = (u, a, t)$ which indicates that a user u execute an application a at time t. Two sets $U_L = \{u \,|\, (u, a, t) \in L\}$ and $A_L = \{a \,|\, (u, a, t) \in L\}$ represent sets of all users and applications in L, respectively. Our dataset consists of about 866 thousands of triplets ($|L| \approx 866,000$) having 211 of applications ($|A_L| = 211$) and 157 of users ($|U_L| = 157$).

For a user $u \in U_L$ and an application $a \in A_L$, an indicator function $x_u^a : U_L \times A_L \to \{0, 1\}$ becomes 1 if u uses a, *i.e.*, there exists a triplet (u, a, t) in L. On the contrary, x_u^a becomes 0 if $(u, a, t) \notin L$ holds. A notation t_u^a represents the first time when u uses a, *i.e.*,

$$t_u^a = \begin{cases} \min(\{t \,|\, (u, a, t) \in L\}) & (\, x_u^a = 1\,) \\ \infty & (\text{otherwise}) \end{cases}.$$

We prepare three indicator functions, $call : U_L \times U_L \to \{0, 1\}$, $dep : U_L \times U_L \to \{0, 1\}$ and $com : U_L \times U_L \to \{0, 1\}$, on pairs of users:

$$call(i, j) = \begin{cases} 1 & (i \text{ made a call to } j \text{ or } j \text{ made a call to } i) \\ 0 & (\text{otherwise}) \end{cases}$$

$$dep(i, j) = \begin{cases} 1 & (\text{department and entry year of } i \text{ and } j \text{ are the same}) \\ 0 & (\text{otherwise}) \end{cases}$$

$$com(i, j) = \begin{cases} 1 & (i \text{ and } j \text{ belong to the same community}) \\ 0 & (\text{otherwise}) \end{cases}.$$

A social network $S = (V, E)$ of users consists of a set of vertices $V = U_L$ and a set of weighted directed edges $E = \{e_{j,i} \,|\, j \in U_L, i \in U_L, j \neq i\}$. Each edge $e_{j,i}$ has its own weight $A_{j,i}$ $(0 \leq A_{j,i} \leq 1)$ representing the strength of relationship from j to i. The purpose of this paper is to infer the weight $A_{j,i}$ for every edge in E by using an application log L.

2.2 Modeling User Behavior on Application Adoption

Information diffusion models in social networks [10,7,9,20] are powerful tools to estimate the potential or probability that a user adopts an item by utilizing the effects of neighbors in the network. While several extensions are proposed, the potential that a user i adopts an item a in the linear threshold model[10,17] can be represented as $(1 - \exp(-\sum_{j \in N(i), x_j^a = 1} w_{j,i}))$ where $N(i)$ denotes a set of i's neighbors and $w_{j,i}$ $(0 \leq w_{j,i} \leq 1)$ denotes the effect from j to i, respectively. As similar, the potential is defined as $(1 - \prod_{j \in N(i), x_j^a = 1}(1 - w_{j,i}))$ in the independent cascade model[7,15]. While information diffusion models are not applicable

directly to our problem since they assume a social network (with edge weights $w_{j,i}$), we utilize their concepts to infer a social network.

We propose two models p^a_{IC} and p^a_{LT} which represent the potentials that a user i adopts an item a at time t below:

$$p^a_{\mathrm{IC}}(i,t) = 1 - \prod_{j\,:\,t^a_j < t} (1 - wt^a_j(t) \cdot wc^a_j(t) \cdot A_{j,i}) \tag{1}$$

$$p^a_{\mathrm{LT}}(i,t) = 1 - \exp(- \sum_{j\,:\,t^a_j < t} wt^a_j(t) \cdot wc^a_j(t) \cdot A_{j,i}) \tag{2}$$

where $wt^a_j(t)\,(0 \leq wt^a_j(t) \leq 1)$ and $wc^a_j(t)\,(0 \leq wc^a_j(t) \leq 1)$ are functions representing the effects of time difference and significance of a at t, respectively. Formal discussion on $wt^a_j(t)$ and $wc^a_j(t)$ will be given later. While p^a_{IC} can be regarded as an extension of the independent cascade model, p^a_{LT} corresponds to the linear threshold model. In the proposed models, we assume that the network is completely linked and the effects of neighbors $w_{j,i}$ is composed as a combination of $wt^a_j(t)$, $wc^a_j(t)$ and $A_{j,i}$. In other words, by adjusting the effects of neighbors through the consideration of time difference and application significance, we try to reduce the negative effects on the complete network assumption and to build a precise model of information diffusion to infer $A_{j,i}$ appropriately.

Given a user j and an application a, various instantiations of $wt^a_j(t)$ can be considered as shown in [15,8]. In this paper, we propose two functions wt_F and wt_L on time difference based on the exponential model:

$$wt\_F^a_j(t) = \begin{cases} \exp(\,-(t - t^a_j)\,/\,\alpha) & (t^a_j < t) \\ 0 & (\text{otherwise}) \end{cases} \tag{3}$$

$$wt\_L^a_j(t) = \begin{cases} \exp(\,-(t - tmax^a_j(t))\,/\,\alpha) & (t^a_j < t) \\ 0 & (\text{otherwise}) \end{cases} \tag{4}$$

where $\alpha\,(> 0)$ is a user defined parameter, and $tmax^a_j(t) = \max(\{t_j \mid (j,a,t_j) \in L, t_j < t\})$ denotes the maximal or most recent time when j uses a before t. A function $wt\_F^a_j(t)$ employs the longest time difference from t in considering the effects of time difference, while $wt\_L^a_j(t)$ uses the shortest one. We regard that $wt\_F^a_j(t)$ represents a situation under which j's potential on transmission a to other users decreases monotonically with time. On the other hand, $wt\_L^a_j(t)$ captures the situation in which the potential is initialized whenever j uses a (see Fig. 2).

In addition to $wt\_F^a_j(t)$ and $wt\_L^a_j(t)$, a function $wt\_C^a_j(t)$ which ignores the effect of time difference is prepared for the comparison purpose. The formal definition is shown below.

$$wt\_C^a_j(t) = \begin{cases} 1 & (t^a_j < t) \\ 0 & (\text{otherwise}) \end{cases} \tag{5}$$

We discuss instantiations of function $wc^a_j(t)$ for representing the significance of applications. First, we introduce two measures on application usages which

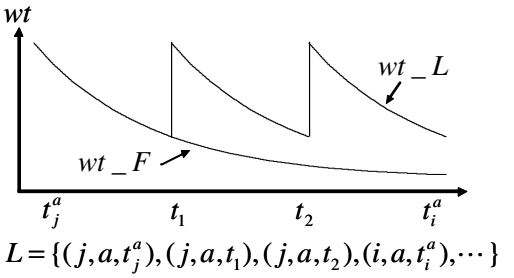


Fig. 2. The effects of Time Difference

correspond to the term frequency and the document frequency in information retrieval, respectively. For a user j and an application a,

$$tf_j^a(t) = |\{t_x \,|\, (j,a,t_x) \in L, t_x \leq t\}| \,/ \sum_{c \in A_L} |\,\{t_y \,|\, (j,c,t_y) \in L, t_y \leq t\}\,|$$

denotes the j's frequency of using a by time t. As similar,

$$df^a(t) = |\,\{u \,|\, (u,a,t_x) \in L, t_x \leq t\}\,| \,/\, |U_L|$$

represents the ratio of users who adopt an application a by time t. By using these two measures, two functions $wc\_I_j^a(t)$ and $wc\_T_j^a(t)$ are proposed which are based on the exponential model and which represent the significance of application a at time t by a user j:

$$wc\_I_j^a(t) = \begin{cases} 1 - \exp(-\log(1 \,/\, df^a(t)) \,/\, \beta) & (t_j^a < t) \\ 0 & (\text{otherwise}) \end{cases} \tag{6}$$

$$wc\_T_j^a(t) = \begin{cases} 1 - \exp(-tf_j^a(t) \cdot \log(1 \,/\, df^a(t)) \,/\, \beta) & (t_j^a < t) \\ 0 & (\text{otherwise}) \end{cases} \tag{7}$$

where $\beta\,(>0)$ is a user defined parameter.

As you can see, $wc\_I_j^a(t)$ represents an weighting function based on the inverse document frequency. Ordinary users will adopt major applications without direct influence from their neighbors. Thus, by giving small weights to popular applications by using $wc\_I_j^a(t)$, we can expect to capture more precise processes of information diffusion for application adoptions. While $wc\_I_j^a(t)$ gives the same weights for any user j such that $t_j^a \leq t$, $wc\_T_j^a(t)$ can give different weights for different users since it inherits the concept of tf-idf(term frequency-inverse document frequency). The weight of application is determined by considering not only the popularity but also individual usages of applications.

Similar to the case of time difference, a constant function

$$wc\_C_j^a(t) = \begin{cases} 1 & (t_j^a < t) \\ 0 & (\text{otherwise}) \end{cases} \tag{8}$$

is employed for ignoring the effects of application significance.

2.3 Estimation of User Influence

As similar to the related work[15,17], we formalize the problem of inferring the edge weight $A_{j,i}$ $(0 \leq A_{j,i} \leq 1)$ as an optimization problem of maximizing a log likelihood function.

We infer a set of edge weights $A(i) = \{A_{j,i} \,|\, j \in U_L, j \neq i\}$ for each user $i \in U_L$. Given an application log L and a user $i \in U_L$, by employing the concept of independent cascade model, the likelihood function of i's behaviors in L is defined as follows:

$$f_{\mathrm{IC}}(A(i), L) = \prod_{a \in A_L : x_i^a = 1} [\, p_{\mathrm{IC}}^a(i, t_i^a) \,] \cdot \prod_{a \in A_L : x_i^a = 0} \left[\prod_{j : x_j^a = 1} \left(1 - \max_{t' : (j,a,t') \in L} wc_j^a(t') \cdot A_{j,i} \right) \right] \tag{9}$$

The first term in $f_{\mathrm{IC}}(A(i), L)$ corresponds to the mobile applications used by a user i. Thus, the adoption time t_i^a for each application a is used in considering the weight for time difference and that for application significance in p_{IC}^a. On the other hand, the second term in $f_{\mathrm{IC}}(A(i), L)$ corresponds to the applications not used by i. We regard that the adoption time $t_i^a = \infty$ is nonsense for capturing the weights for time difference and that for application significance. To obtain meaningful weights for accurate estimation, we consider to use the maximal influence. To obtain a rough approximation of the maximal influence $\max_{t'} p_{\mathrm{IC}}^a(i, t')$, the effects of time difference is eliminated from p_{IC}^a by assuming $wt_j^a(t') = 1$ and the maximal application significance $\max_{t' : (j,a,t') \in L} wc_j^a(t')$ is employed.

By applying the same discussion, we obtain the likelihood function of i's behaviors based on the linear threshold model as follows:

$$f_{\mathrm{LT}}(A(i), L) = \prod_{a \in A_L : x_i^a = 1} [\, p_{\mathrm{LT}}^a(i, t_i^a) \,] \cdot \prod_{a \in A_L : x_i^a = 0} \left[\exp \left(- \sum_{j : x_j^a = 1} \max_{t' : (j,a,t') \in L} wc_j^a(t') \cdot A_{j,i} \right) \right] \tag{10}$$

We use a general nonlinear optimization routine provided in R-language[18] to obtain $A(i)$ which maximizes log likelihood functions $\log(\, f_{\mathrm{IC}}(A(i), L)\,)$ and $\log(\, f_{\mathrm{LT}}(A(i), L)\,)$ under the constraint $0 \leq A_{j,i} \leq 1$.

2.4 Extension

To infer a social network more accurately, we introduce two extensions to the basic model proposed in the previous subsections.

Introduction of Penalty Terms: As proposed in a previous work[15], we introduce penalty terms in maximal likelihood estimation. Instead of maximizing the log likelihood functions, we obtain $A_{j,i}$ which maximizes log likelihood function with penalty term $\log(\, f_{\mathrm{IC}}(A(i), L)\,) - \gamma\mathrm{PT}$ and $\log(\, f_{\mathrm{LT}}(A(i), L)\,) - \gamma\mathrm{PT}$ where $\gamma\,(> 0)$ is a user defined parameter and PT denotes a penalty term.

Three penalty terms are prepared:

$$p_{\mathrm{sn}} = \sum_{j \in U_L} A_{j,i}, \quad p_{\mathrm{call}} = -\sum_{j:call(j,i)=1} A_{j,i}, \quad \text{and} \quad p_{\mathrm{dep}} = -\sum_{j:dep(j,i)=1} A_{j,i}.$$

While the first one is domain independent, the others are specialized for the dataset in this paper. The first penalty term p_{sn} prefers a sparse network having small amount of weights. The second penalty p_{call} gives certain bonus to a network if it contains edges with heavy weight between two users j and i who made a telephone call. The implicit assumption behind this penalty term is that two users j and i are friends and thus the weight from j to i must be high. Based on the similar considerations, the third penalty p_{dep} employs the information of users departments in the university.

Restriction of Time Period for Computing Application Significance: To reflect recent situations, we consider to restrict the triplet in L for computing $tf_j^a(t)$ and $df^a(t)$. The restricted versions of document frequency and term frequency are defined as follows:

$$tf_j^a(t,\delta) = \frac{|\{t_x \,|\, (j,a,t_x) \in L, (t-\delta) < t_x \leq t\}|}{\sum_{c \in A_L} |\,\{t_y \,|\, (j,c,t_y) \in L, (t-\delta) < t_y \leq t\}\,|}$$
$$df^a(t,\delta) = |\,\{u \,|\, (u,a,t_x) \in L, (t-\delta) < t_x \leq t\}\,| \,/\, |U_L|$$

where $\delta\,(> 0)$ is a user defined parameter for restricting the time period. Elimination of old triplets can expect to capture more recent situation accurately.

Corresponding functions on application significance, denoted as $wc\_I_j^a(t,\delta)$ and $wc\_T_j^a(t,\delta)$, are obtained by replacing $tf_j^a(t)$ and $df^a(t)$ with those restrictions $tf_j^a(t,\delta)$ and $df^a(t,\delta)$ in equation (6) and (7), respectively.

3 Related Work

Recently, modeling the process of information diffusion in a social network attracts much attention and many models as well as inference methods are proposed. While [9] proposes a method for learning influence probabilities in the framework of linear threshold model, [20] and [19] propose learning methods in (extended) independent cascade model. In addition, [11] proposes a diffusion model for taking into account of the effects of multiple contagion based on SIS model[23]. These models must be useful for estimating the strength of relationship among users in a network. Although they are not applicable directly to our problem, we can adopt the concept of diffusion models to infer a social network.

We mention recent progress of learning the strength of user relation from log data. An algorithm named NetInf[8] discovers a directed and unweighted network from a log data. While NetInf makes use of the effects of time difference on information adoption to obtain a final network, it does not consider the effects of significance of contents.

ConNie[15] is also an algorithm for inferring a social network from a log data. It employs the concept of extended independent cascade model for handling time differences of information adoption. A weighted directed network can be inferred as an optimization problem with a penalty term. While the overall framework of our proposal on the independent cascade model is very similar to that in ConNie, it does not consider the effects of popularity of contents nor multiple adoptions of an item by a user. In addition, no method based on the linear threshold model is mentioned.

A probabilistic model for user adoption behaviors is proposed which can extract a directed weighted network from a log data [3]. It incorporates the effects of popularity as well as recency of contents by introducing virtual users. In addition, the effect of time difference is considered by restricting the data used for estimating the model. This restriction is similar to our extension for restricting time period for computing application significance.

Related to the network estimation from mobile phone data, [5] estimates the networks from observational data from mobile phones, and then compares them to a network obtained by self-report survey data. While [5] uses logs on phone call and bluetooth proximity, it does not consider the diffusion of applications. [17] proposes a model for mobile application adoptions based on the linear threshold model by taking plural networks. While it applies the information diffusion model to the data obtained from smartphones, the purpose is not network inference but prediction of application installations.

4 Experiments

4.1 Baseline Methods and Evaluation Measures

To assess the effectiveness of the proposal, we infer networks by combining different diffusion models(f_{IC} and f_{LT}), functions on application significance (wc_I, wc_T and wc_C) with $\beta = 1$ and functions on time difference (wt_F, wt_L and wt_C) with $\alpha \in \{75, 150, 300\}$. One hour is taken as time unit in the experiments. $\alpha = 150$ cuts down the effects to about 0.3 in a week. The combinations of two extensions, (1)introduction of penalty terms with $\gamma \in \{1, 2\}$ and (2)restriction of the time period with $\delta \in \{7, 14, 28\}$, are also examined.

Cosine similarity of tf-idf (cos) and jaccard coefficient (jac) are employed as baseline methods:

$$\cos : A_{j,i} = \sum_{a \in A_L} \mathrm{T}(j,a)\mathrm{T}(i,a) \,/\, \sqrt{\sum_{a \in A_L} \mathrm{T}(j,a)^2} \sqrt{\sum_{a \in A_L} \mathrm{T}(i,a)^2}$$
$$\mathrm{jac} : A_{j,i} = |\,\{a \in A_L \,|\, x_j^a = x_j^a = 1\}\,| \,/\, |\,\{a \in A_L \,|\, x_j^a = 1 \vee x_j^a = 1\}\,|$$

where $\mathrm{T}(u,a) = tf_u^a(\infty) \cdot \log(1 \,/\, df^a(\infty))$ denotes u's value of tf-idf on an application a during all the experiment period. As an additional baseline method, a

network "imp" is prepared, which is the best results obtained by the method in [3] with several combinations of parameters.

Three evaluation measures are prepared from different aspects, all of which employ the user community, *i.e.* an indicator function $com : U_L \times U_L \to \{0, 1\}$, as a ground truth.

To evaluate the results from the aspect of users, we focus on the communities to which users having strong relationship belong. For a user i and positive integer k, let $g(i, k)$ be a set of users j who is in the top-k place in descending order of $A_{i,j}$. In other words, $g(i, k)$ is a set of users to whom a user i gives the best-k strongest effects. By assuming that users receiving strong effects from i must belong to the same community as i, we define an evaluation measure based on precision@k as follows:

$$p_g(k) = \textstyle\sum_{i \in U_L} |\{j \in g(i, k) \mid com(j, i) = 1\}| \,/\, \sum_{i \in U_L} |g(i, k)|.$$

In the experiments, we assess the results of $k = 3$ and $k = 5$.

The second evaluation measure f_e is f-measure of edge prediction:

$$f_e = \max_{0 \le \theta \le 2} 2 \cdot r_e(\theta) \cdot p_e(\theta) \,/\, (r_e(\theta) + p_e(\theta))$$

where

$$r_e(\theta) = \frac{|\,\{\langle j,i\rangle \mid i,j \in U_L, i \neq j, com(j,i)=1, A^+_{j,i} \ge \theta\}\,|}{|\,\{\langle j,i\rangle \mid i,j \in U_L, i \neq j, com(j,i)=1\}\,|},$$

$$p_e(\theta) = \frac{|\,\{\langle j,i\rangle \mid i,j \in U_L, i \neq j, com(j,i)=1, A^+_{j,i} \ge \theta\}\,|}{|\,\{\langle j,i\rangle \mid i,j \in U_L, i \neq j, A^+_{j,i} \ge \theta\}\,|}$$

and $A^+_{j,i} = A_{j,i} + A_{i,j}$. In this measure, the direction of edges is ignored by employing $A^+_{j,i}$.

We also evaluate the networks through the community discovery [22,6]. By using $A^+_{i,j}$, a community structure having maximal modularity [16] is discovered by using the igraph library [4]. An indicator function $com^m(j, i)$ becomes 1 if two users j and i are judged as belonging to the same community by the community discovery algorithm. Otherwise, $com^m(j, i) = 0$. The results are evaluated by f-measure

$$f_c = 2 \cdot r_c \cdot p_c \,/\, (r_c + p_c)$$

where

$$r_c = \frac{|\,\{\langle j,i\rangle \mid i,j \in U_L, i \neq j, com(j,i)=1, com^m(j,i)=1\}\,|}{|\,\{\langle j,i\rangle \mid i,j \in U_L, i \neq j, com(j,i)=1\}\,|} \quad \text{and}$$

$$p_c = \frac{|\,\{\langle j,i\rangle \mid i,j \in U_L, i \neq j, com(j,i)=1, com^m(j,i)=1\}\,|}{|\,\{\langle j,i\rangle \mid i,j \in U_L, i \neq j, com^m(j,i)=1\}\,|}.$$

4.2 Results

The best 5 results of each evaluation measure are shown in Table 1. The results of baseline methods are also shown. In the table, the combinations of functions and extensions are represented in the form of

$$[\ \{IC, LT\},\ wt/\alpha,\ wc,\ \text{penalty}/\gamma,\ \text{restriction of time period}\ ]$$

Table 1. Best-5 models for each evaluation measure

$p_g(3)$	model				
0.530	LT	$wt\_F/300$	wc_T	$p_{\mathrm{dep}}/2$	28
0.521	LT	wt_C	wc_T	$p_{\mathrm{dep}}/2$	28
0.517	LT	$wt\_L/300$	wc_T	$p_{\mathrm{dep}}/2$	28
0.509	LT	$wt\_F/300$	wc_T	$p_{\mathrm{dep}}/2$	14
0.509	LT	$wt\_L/150$	wc_T	$p_{\mathrm{dep}}/2$	28
0.421	IC	$wt\_F/300$	wc_T	$p_{\mathrm{dep}}/1$	–
0.404	IC	$wt\_F/150$	wc_T	$p_{\mathrm{call}}/1$	–
0.404	IC	$wt\_F/300$	wc_T	$p_{\mathrm{call}}/2$	–
0.402	IC	$wt\_F/300$	wc_T	$p_{\mathrm{call}}/1$	–
0.402	IC	$wt\_F/300$	wc_T	$p_{\mathrm{dep}}/1$	28
0.408	cos				
0.333	jac				
0.344	imp				

$p_g(5)$	model				
0.509	LT	$wt\_F/300$	wc_T	$p_{\mathrm{dep}}/2$	28
0.501	LT	$wt\_L/300$	wc_T	$p_{\mathrm{dep}}/2$	28
0.494	LT	$wt\_L/75$	wc_T	$p_{\mathrm{dep}}/2$	28
0.490	LT	wt_C	wc_T	$p_{\mathrm{dep}}/2$	28
0.487	LT	$wt\_L/150$	wc_T	$p_{\mathrm{dep}}/2$	28
0.396	IC	$wt\_F/300$	wc_T	$p_{\mathrm{dep}}/1$	–
0.377	IC	$wt\_L/75$	wc_T	$p_{\mathrm{dep}}/2$	28
0.373	IC	$wt\_F/150$	wc_T	–	–
0.373	IC	$wt\_L/300$	wc_T	$p_{\mathrm{call}}/2$	–
0.371	IC	$wt\_F/150$	wc_T	$p_{\mathrm{dep}}/1$	–
0.392	cos				
0.329	jac				
0.326	imp				

f_e	model				
0.393	LT	$wt\_L/75$	wc_T	$p_{\mathrm{dep}}/2$	–
0.391	LT	$wt\_F/300$	wc_T	$p_{\mathrm{dep}}/2$	–
0.390	LT	wt_C	wc_T	$p_{\mathrm{dep}}/2$	–
0.390	LT	$wt\_L/150$	wc_T	$p_{\mathrm{dep}}/2$	–
0.390	LT	$wt\_F/300$	wc_T	$p_{\mathrm{dep}}/2$	28
0.362	IC	$wt\_F/75.0$	wc_T	–	–
0.360	IC	$wt\_F/150$	wc_T	–	–
0.358	IC	$wt\_F/75.0$	wc_T	$p_{\mathrm{call}}/2$	–
0.357	IC	$wt\_F/150$	wc_T	$p_{\mathrm{dep}}/1$	–
0.357	IC	$wt\_L/75$	wc_T	$p_{\mathrm{dep}}/1$	–
0.384	cos				
0.384	jac				
0.386	imp				

f_c	model				
0.517	LT	$wt\_F/300$	wc_T	$p_{\mathrm{dep}}/2$	–
0.474	LT	$wt\_L/150$	wc_T	$p_{\mathrm{dep}}/2$	28
0.468	LT	$wt\_L/75$	wc_T	$p_{\mathrm{dep}}/2$	–
0.466	LT	wt_C	wc_T	$p_{\mathrm{dep}}/2$	–
0.462	LT	$wt\_F/150$	wc_T	$p_{\mathrm{dep}}/1$	–
0.357	IC	$wt\_L/150$	wc_T	$p_{\mathrm{call}}/2$	–
0.353	IC	$wt\_F/75$	wc_T	$p_{\mathrm{dep}}/2$	–
0.351	IC	$wt\_F/150$	wc_T	$p_{\mathrm{dep}}/2$	28
0.347	IC	$wt\_L/150$	wc_T	$p_{\mathrm{call}}/2$	–
0.345	IC	$wt\_F/75$	wc_T	–	28
0.329	cos				
0.325	jac				
0.386	imp				

where IC and LT correspond to the likelihood functions (9) and (10), respectively. The symbol "–" means that the corresponding extension is not applied.

We can observe in the results that the proposed method based on the linear threshold model clearly outperforms the baseline methods, while no significant improvements are recognized for the independent cascade model. The notable performance gains of LT in $p_g(3)$, $p_g(5)$ and f_c indicate that the proposed models succeed in capturing the primary relations in a social network.

As previously mentioned, the proposed models employ the concept of information diffusion models. By assuming a complete network, the effects of neighbors is re-defined as the combination of $wt_j^a(t)$, $wc_j^a(t)$ and $A_{j,i}$. As a second experiment, we compare the proposed models to the basic information diffusion models to confirm each effect of time difference and application significance. The networks of the basic information diffusion models are obtained by using functions wt_C on time difference and wc_C on application significance, respectively. The

Table 2. Comparisons with the basic information diffusion models

	$p_g(3)$		$p_g(5)$		f_e		f_c	
	LT	IC	LT	IC	LT	IC	LT	IC
wt_C, wc_C	0.167	0.226	0.180	0.148	0.016	0.066	0.070	0.187
$wt\_F/75$	0.109	0.248	0.084	0.247	0.031	0.238	0.102	0.213
$wt\_F/150$	0.162	0.212	0.130	0.249	0.027	0.162	0.081	0.207
$wt\_F/300$	0.058	0.215	0.057	0.197	0.018	0.106	0.085	0.176
$wt\_L/75$	0.120	0.216	0.088	0.213	0.025	0.188	0.098	0.213
$wt\_L/150$	0.107	0.282	0.100	0.259	0.019	0.147	0.090	0.220
$wt\_L/300$	0.086	0.229	0.081	0.178	0.018	0.088	0.076	0.185
wc_I	0.083	0.222	0.100	0.145	0.017	0.032	0.085	0.095
wc_T	0.372	0.378	0.365	0.358	0.344	0.341	0.326	0.320
$p_{\mathrm{sn}}/1$	0.222	0.198	0.133	0.145	0.001	0.057	0.001	0.129
$p_{\mathrm{sn}}/2$	0.222	0.216	0.133	0.154	0.001	0.063	0.001	0.154
$p_{\mathrm{call}}/1$	0.148	0.198	0.178	0.140	0.015	0.063	0.073	0.156
$p_{\mathrm{call}}/2$	0.111	0.221	0.156	0.158	0.015	0.067	0.072	0.169
$p_{\mathrm{dep}}/1$	0.267	0.195	0.200	0.138	0.025	0.064	0.084	0.186
$p_{\mathrm{dep}}/2$	0.281	0.199	0.200	0.134	0.028	0.056	0.096	0.128

results of diffusion models are compared those results obtained by single modification of the basic models. More specifically, the combination of 'wt_F/α and wc_C' or 'wt_L/α and wc_C' is used to assess the effects of the time difference. As similar, the combinations 'wt_C and wc_T' and 'wt_C and wc_I' are also employed for examining the effect of the application significance. In addition, the effects of penalty terms with wt_C and wc_C are compared. The results are shown in Table 2.

In the results, performance improvements from the basic information diffusion models are observed in only half cases. In addition, these results are not effective, some times quite poor, compared with the baseline methods in the first experiments. On application significance, the function wc_T contributes to the performance gain in all cases. Introduction of time difference has negative effects on the LT models in $p_g(3)$ and $p_g(5)$, while it works well in f_e and f_c. Although p_{dep}s are promising for the LT models, their effects are not observed in the IC models. From these results, we simply conclude that the separate and independent incorporation of time difference, application significance and penalty term does not always have positive effects for improving the basic information diffusion models. On the other hand, by taking together with the first experiments, we can expect large synergistic effects for significant performance improvements by employing appropriate combinations such as wc_T and $p_{\mathrm{dep}}/2$. The results also show that the appropriate combinations heavily depend on evaluation measures. Thus, we have to carefully select the combinations by considering the purpose of subsequent applications of obtained results.

As a third experiment, we compare different instantiations of functions on time difference and application significance. In order to determine an instantiation on time difference which derives the best performance stably, we compare

the models having the same parameters other than $wt_j^a(t)$ with α. The same comparisons are conducted for the significance of applications ($wc_j^a(t)$), introduction of penalty term, and restriction of time period. We summarize the rates of taking the best performance within the models having the same instantiations except one comparing target in Table 3.

Table 3. Winning rates within the models having the same instantiations except one comparing target

	$p_g(3)$		$p_g(5)$		f_e		f_c	
model	LT	IC	LT	IC	LT	IC	LT	IC
$wt\_F/75$	0.167	0.333	0.153	0.488	0.821	0.929	0.488	0.238
$wt\_F/150$	0.131	0.327	0.143	0.113	0.024	0.024	0.095	0.238
$wt\_F/300$	0.125	0.077	0.071	0.054	0.012	0.012	0.048	0.131
$wt\_L/75$	0.095	0.095	0.048	0.250	0.131	0.036	0.202	0.131
$wt\_L/150$	0.060	0.060	0.042	0.071	0.012	0.000	0.083	0.155
$wt\_L/300$	0.012	0.071	0.028	0.024	0.000	0.000	0.036	0.071
wt_C	0.411	0.036	0.516	0.000	0.000	0.000	0.048	0.036
wc_I	0.010	0.061	0.000	0.051	0.000	0.041	0.000	0.026
wc_T	0.985	0.893	1.000	0.878	1.000	0.913	1.000	0.816
wc_C	0.005	0.046	0.000	0.071	0.000	0.046	0.000	0.158
$p_{\mathrm{sn}}/1$	0.000	0.012	0.000	0.012	0.000	0.000	0.000	0.012
$p_{\mathrm{sn}}/2$	0.006	0.060	0.000	0.048	0.000	0.000	0.000	0.024
$p_{\mathrm{call}}/1$	0.000	0.190	0.012	0.143	0.000	0.167	0.012	0.214
$p_{\mathrm{call}}/2$	0.060	0.190	0.071	0.268	0.000	0.226	0.107	0.190
$p_{\mathrm{dep}}/1$	0.202	0.232	0.179	0.149	0.012	0.190	0.179	0.214
$p_{\mathrm{dep}}/2$	0.732	0.155	0.726	0.214	0.976	0.143	0.690	0.179
–	0.000	0.161	0.012	0.167	0.012	0.274	0.012	0.167
$\delta = 7$	0.138	0.226	0.151	0.226	0.138	0.124	0.145	0.240
$\delta = 14$	0.185	0.199	0.158	0.219	0.124	0.226	0.131	0.185
$\delta = 28$	0.366	0.223	0.270	0.172	0.145	0.247	0.247	0.219
–	0.311	0.352	0.420	0.383	0.594	0.403	0.478	0.355

If we ignore the synergistic effects, the time difference function $wt\_F/75$ contributes to improve f_e in both IC and LT models. It also gets the best performance in f_c of LT and $p_g(5)$ of IC. On the other hand, ignoring time difference by using wc_C derives better results for $p_g(3)$ and $p_g(5)$ of LT models. A function wc_T on application significance clearly outperforms other instantiations in every evaluation measures. As similar, the penalty term $p_{\mathrm{dep}}/2$ contributes to obtain the significantly results in LT models. These two observations are consistent with the results of the first experiments. Extensions do not always contribute to gain the performance. The penalty terms $p_{\mathrm{sn}}/1$ and $p_{\mathrm{sn}}/2$ seem not to have positive effects in this dataset. Non-restriction of time period for computing application significance takes the first place in all cases except of $p_g(3)$ of LT. These results suggest again that selecting appropriate combinations is a key for successful network inference.

5 Conclusion

In this paper, we discuss the problem of inferring a social network of smartphone users from their application usages. To capture the user behavior more accurately, we adopt two basic information diffusion models, linear threshold model and independent cascade model, and extend them by incorporating the effects of time difference and application significance. Experimental study shows that our models can improve the performance of social network inference compared with naive methods.

For future work, we plan to employ more sophisticated diffusion models, *e.g.* incorporation of the effects of topics [21,13]. In addition, we believe that consideration of the homophily effects [14,12,2,1] in modeling user behavior is one of promising future directions for the accurate social network discovery.

References

1. Anagnostopoulos, A., Kumar, R., Mahdian, M.: Influence and correlation in social networks. In: Proc. of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 7–15 (2008)
2. Aral, S., Muchnik, L., Sundararajan, A.: Distinguishing influence-based contagion from homophily-driven diffusion in dynamic networks. Proc. of National Academy of Sciences of the United States of America 106(51), 21544–21549 (2009)
3. Au Yeung, C., Iwata, T.: Capturing implicit user influence in online social sharing. In: Proc. of the 21st ACM Conference on Hypertext and Hypermedia, pp. 245–254 (2010)
4. Csardi, G., Nepusz, T.: The igraph software package for complex network research. Inter. Journal, Complex Systems, 1695 (2006)
5. Eagle, N., Pentland, A.S., Lazer, D.: Inferring friendship network structure by using mobile phone data. Proceedings of the National Academy of Sciences 106(36), 15274–15278 (2009)
6. Fortunato, S.: Community detection in graphs. Physics Reports 486(3–5), 75–174 (2010)
7. Goldenberg, J., Libai, B., Muller, E.: Talk of the Network: A Complex Systems Look at the Underlying Process of Word-of-Mouth. Marketing Letters 12(3), 211–223 (2001)
8. Gomez Rodriguez, M., Leskovec, J., Krause, A.: Inferring networks of diffusion and influence. In: Proc. of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1019–1028 (2010)
9. Goyal, A., Bonchi, F., Lakshmanan, L.V.: Learning influence probabilities in social networks. In: Proc. of the 3rd ACM International Conference on Web Search and Data Mining, pp. 241–250 (2010)
10. Granovetter, M.: Threshold Models of Collective Behavior. The American Journal of Sociology 83(6), 1420–1443 (1978)
11. Kimura, M., Saito, K., Motoda, H.: Efficient estimation of influence functions for sis model on social networks. In: Proc. of the 21st International Joint Conference on Artificial Intelligence, pp. 2046–2051 (2009)
12. La Fond, T., Neville, J.: Randomization tests for distinguishing social influence and homophily effects. In: Proc. of the 19th International Conference on World Wide Web, pp. 601–610 (2010)

13. Liu, L., Tang, J., Han, J., Jiang, M., Yang, S.: Mining topic-level influence in heterogeneous networks. In: Proc. of the 19th ACM International Conference on Information and Knowledge Management, pp. 199–208 (2010)
14. McPherson, M., Lovin, L.S., Cook, J.M.: Birds of a Feather: Homophily in Social Networks. Annual Review of Sociology 27(1), 415–444 (2001)
15. Myers, S., Leskovec, J.: On the convexity of latent social network inference. Advances in Neural Information Processing Systems 23, 1741–1749 (2010)
16. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Physical Review E 69(2), 26113 (2004)
17. Pan, W., Aharony, N., Pentland, A.: Composite social network for predicting mobile apps installation. In: Proc. of the 25th Conference on Artificial Intelligence (2011)
18. R Development Core Team. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing (2011)
19. Saito, K., Kimura, M., Ohara, K., Motoda, H.: Learning Continuous-Time Information Diffusion Model for Social Behavioral Data Analysis. In: Zhou, Z.-H., Washio, T. (eds.) ACML 2009. LNCS, vol. 5828, pp. 322–337. Springer, Heidelberg (2009)
20. Saito, K., Nakano, R., Kimura, M.: Prediction of Information Diffusion Probabilities for Independent Cascade Model. In: Lovrek, I., Howlett, R.J., Jain, L.C. (eds.) KES 2008, Part III. LNCS (LNAI), vol. 5179, pp. 67–75. Springer, Heidelberg (2008)
21. Tang, J., Sun, J., Wang, C., Yang, Z.: Social influence analysis in large-scale networks. In: Proc. of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 807–816 (2009)
22. Tang, L., Liu, H.: Community Detection and Mining in Social Media. Morgan & Claypool Publishers (2010)
23. Wierman, J.C., Marchette, D.J.: Modeling computer virus prevalence with a susceptible-infected-susceptible model with reintroduction. Computational Statistics & Data Analysis 45(1), 3–23 (2004)

Support Vector Regression with A Priori Knowledge Used in Order Execution Strategies Based on VWAP

Marcin Orchel

AGH University of Science and Technology, Mickiewicza Av. 30,
30-059 Kraków, Poland
marcin@orchel.pl

Abstract. In this article, we propose a novel application for Support Vector Regression (SVR) for order execution strategies on stock exchanges. We use SVR for predicting volume participation function in execution strategies which try to achieve Volume Weighted Average Price (VWAP) measure of quality of the execution. Moreover, we use SVR with a priori knowledge about stock prices in order to further improve the cost of order execution. The main result is that SVR outperforms tested null hypotheses such as prediction based on average from historical data. SVR with additional knowledge about prices improve the prediction performance and the final execution error. The tests were performed on real stock data from NASDAQ exchange.

Keywords: Support Vector Machines, order execution strategies, vwap.

1 Introduction

Big orders cannot be executed on exchanges at once because of limited number of offers on the opposite side. They must be split into smaller orders and execute in a longer time period. There are various possible measures of the quality of order execution. The most popular are market volume weighted average price (market VWAP), pre-trade price, and post-trade price compared to VWAP for the order. In this article, we investigate the first one. The model of the strategy achieving market VWAP was presented recently in [1] [2]. In [1], authors found that improving quality of the volume prediction leads to better execution performance, although they had found contradicting results in [4].

The goal of the work was to extend the theoretical results for the execution strategy achieving VWAP and to show on which factors the final execution error depends on. Furthermore, we wanted to implement a part of the strategy by using a general purpose machine learning method such as Support Vector Regression (SVR).

One of the main learning problems is a regression estimation. Vapnik [10] proposed a new regression method, which is called ε-insensitive Support Vector Regression (ε-SVR). It belongs to a group of methods called Support Vector Machines (SVM). For estimating indicator functions Support Vector Classification

J. Tang et al. (Eds.): ADMA 2011, Part II, LNAI 7121, pp. 318–331, 2011.

(SVC) method was developed [10]. SVM were invented on a basis of statistical learning theory. They are efficient learning methods partly for the reason of having the following important properties: they realize Structural Risk Minimization principle, they lead to convex optimization problems, they generate sparse solutions, kernel functions can be used for generating nonlinear solutions.

Recently, an alternative regression method was proposed [8,5], which is called δ-SVR. The idea of the new method is to duplicate and shift data in order to use SVC to solve regression problems. The new method posses the same important advantages as ε-SVR: it leads to convex optimization problems, it generates sparse solutions, kernel functions can be used for generating nonlinear solutions. It was shown experimentally, that δ-SVR can achieve comparable or better generalization performance compared to ε-SVR [8]. It was also reported in [8] that some type of a priori knowledge already incorporated to SVC can be directly used for regression problems. We will use δ-SVR with incorporated a priori knowledge about prices.

In [1], authors predict a volume function by decomposing volume into two parts and using the average method and autoregressive models. In [2], authors predict a volume participation function instead of volume function by using autoregressive models. In this article, we use SVR for predicting volume participation function based on historical data. We compare SVR with some proposed null hypotheses such as predicting volume participation while assuming constant volume profile, prediction based on average from historical data for the same time slice and prediction from the previous time slice.

The final execution performance depends not only on volume but also on stock prices during order execution. One of the way of improving the strategy is to incorporate information about prices to the model. The presented strategy splits the order to smaller chunks based on volume participation function. The possible way of incorporating information about prices to the model is to adjust volume participation function. We propose modeling the final solution by incorporating a priori knowledge about prices by using detractors a priori knowledge recently proposed for SVC [6], for δ-SVR [7] and for ε-SVR [9]. It was used for manipulating a decision curve for classification problems, and manipulating a regression function for regression ones.

A test scenario which is investigated in this article is to split execution of the order during a one exchange session. Note that the size of the order has a direct influence on the possibility of achieving VWAP. It is easier to achieve VWAP for bigger orders relative to the daily volume, because the order is also a part of the market VWAP. In the extreme situation where the order is the only one executed during the session we achieve VWAP (neglecting transaction costs of executing the order).

The outline of the article is as follows, first an introduction to ε-SVR, δ-SVR and detractors is given. Then, we present an introduction to Volume Participation Strategy. Next, we present incorporating a priori knowledge about prices, and finally experiments are described.

1.1 Introduction to ε-SVR and SVC

In a regression estimation, we consider a set of training vectors $\boldsymbol{x_i}$ for $i = 1..l$, where $\boldsymbol{x_i} = \left(x_i^1, \ldots, x_i^m\right)$. The i-th training vector is mapped to $y_{\mathrm{r}}^i \in \mathbb{R}$. The m is a dimension of the problem. The ε-SVR soft case optimization problem is

OP 1. *Minimization of*

$$f\left(\boldsymbol{w}_{\mathbf{r}}, b_{\mathrm{r}}, \boldsymbol{\xi}_{\mathbf{r}}, \boldsymbol{\xi}_{\mathbf{r}}^*\right) = \left\|\boldsymbol{w}_{\mathbf{r}}\right\|^2 + C_{\mathrm{r}} \sum_{i=1}^{l} \left(\xi_{\mathrm{r}}^i + \xi_{\mathrm{r}}^{*i}\right) \tag{1}$$

with constraints $y_{\mathrm{r}}^i - g\left(\boldsymbol{x_i}\right) \leq \varepsilon + \xi_{\mathrm{r}}^i, \quad g\left(\boldsymbol{x_i}\right) - y_{\mathrm{r}}^i \leq \varepsilon + \xi_{\mathrm{r}}^{i*}, \quad \boldsymbol{\xi}_{\mathbf{r}} \geq 0, \quad \boldsymbol{\xi}_{\mathbf{r}}^* \geq 0$ *for* $i \in \{1..l\}$, *where* $g\left(\boldsymbol{x_i}\right) = \boldsymbol{w}_{\mathbf{r}} \cdot \boldsymbol{x_i} + b_{\mathrm{r}}$.

The $g^*\left(\boldsymbol{x}\right) = \boldsymbol{w}_{\mathbf{r}}^* \cdot \boldsymbol{x} + b_{\mathrm{r}}^*$ is a regression function. Optimization problem 1 is transformed to an equivalent dual optimization problem. The regression function becomes

$$g^*\left(\boldsymbol{x}\right) = \sum_{i=1}^{l} \left(\alpha_i^* - \beta_i^*\right) K\left(\boldsymbol{x_i}, \boldsymbol{x}\right) + b_{\mathrm{r}}^* \ , \tag{2}$$

where α_i, β_i are Lagrange multipliers of the dual problem, $K\left(\cdot,\cdot\right)$ is a kernel function which is incorporated to a dual problem. The most popular kernel functions are linear, polynomial, radial basis function (RBF) and sigmoid. A kernel function which is a dot product of its variables we call *a simple linear kernel*. The i-th training example is a support vector, when $\alpha_i^* - \beta_i^* \neq 0$. It can be proved that a set of support vectors contains all training examples which fall outside the ε tube, and most of the examples which lie on the ε tube.

For an indicator function estimation, we consider a set of training vectors $\boldsymbol{x_i}$ for $i = 1..l$, where $\boldsymbol{x_i} = \left(x_i^1, \ldots, x_i^m\right)$. The i-th training vector is mapped to $y_{\mathrm{c}}^i \in \{0, 1\}$. The m is a dimension of the problem. The SVC 1-norm soft margin case optimization problem is

OP 2. *Minimization of*

$$f\left(\boldsymbol{w}_{\mathbf{c}}, b_{\mathrm{c}}, \boldsymbol{\xi}_{\mathbf{c}}\right) = \left\|\boldsymbol{w}_{\mathbf{c}}\right\|^2 + C_{\mathrm{c}} \sum_{i=1}^{l} \xi_{\mathrm{c}}^i \tag{3}$$

with constraints $y_{\mathrm{c}}^i h\left(\boldsymbol{x_i}\right) \geq 1 - \xi_{\mathrm{c}}^i, \quad \boldsymbol{\xi}_{\mathbf{c}} \geq 0$ *for* $i \in \{1..l\}$, *where* $h\left(\boldsymbol{x_i}\right) = \boldsymbol{w}_{\mathbf{c}} \cdot \boldsymbol{x_i} + b_{\mathrm{c}}$.

The $h^*\left(\boldsymbol{x}\right) = \boldsymbol{w}_{\mathbf{c}}^* \cdot \boldsymbol{x} + b_{\mathrm{c}}^* = 0$ is a decision curve of the classification problem. Optimization problem 2 is transformed to an equivalent dual optimization problem. The decision curve becomes

$$h^*\left(\boldsymbol{x}\right) = \sum_{i=1}^{l} y_{\mathrm{c}}^i \alpha_i^* K\left(\boldsymbol{x_i}, \boldsymbol{x}\right) + b_{\mathrm{c}}^* = 0 \ , \tag{4}$$

where α_i are Lagrange multipliers of the dual problem, $K\left(\cdot,\cdot\right)$ is a kernel function, which is incorporated to a dual problem. *Margin boundaries* are defined as

the two hyperplanes $h(\boldsymbol{x}) = -1$ and $h(\boldsymbol{x}) = 1$. *Optimal margin boundaries* are defined as the two hyperplanes $h^*(\boldsymbol{x}) = -1$ and $h^*(\boldsymbol{x}) = 1$. The i-th training example is a support vector, when $\alpha_i^* \neq 0$. It can be proved that a set of support vectors contains all training examples which fall below optimal margin boundaries ($y_i h^*(\boldsymbol{x_i}) < 1$), and most of the examples which lie on the optimal margin boundaries ($y_i h^*(\boldsymbol{x_i}) = 1$).

1.2 Introduction to δ-SVR

We consider a set of training vectors $\boldsymbol{x_i}$ for $i = 1..l$, where $\boldsymbol{x_i} = \left(x_i^1, \ldots, x_i^m\right)$. The i-th training vector is mapped to $y_{\mathrm{r}}^i \in \mathbb{R}$. The δ-SVR method is based on the following scheme of finding a regression function:

1. Every training example $\boldsymbol{x_i}$ is duplicated, an output value y_{r}^i is translated by a value of a parameter $\delta \geq 0$ for an original training example, and translated by $-\delta$ for a duplicated training example.
2. Every training example $\boldsymbol{x_i}$ is converted to a classification example by incorporating the output as an additional feature and setting class 1 for original training examples, and class -1 for duplicated training examples.
3. SVC is run with the classification mappings.
4. The solution of SVC is converted to a regression form.

The above procedure is repeated for different values of δ. The best value of δ is found experimentally by comparing mean squared error (MSE) on regression examples.

The result of the first step is a set of training mappings for $i \in \{1, \ldots, 2l\}$

$$\begin{cases} \boldsymbol{b_i} = \left(x_i^1, \ldots, x_i^m\right) \rightarrow y_{\mathrm{r}}^i + \delta & \text{for } \; i \in \{1, \ldots, l\} \\ \boldsymbol{b_i} = \left(x_{i-l}^1, \ldots, x_{i-l}^m\right) \rightarrow y_{\mathrm{r}}^{i-l} - \delta & \text{for } \; i \in \{l+1, \ldots, 2l\} \end{cases}$$

for $\delta \geq 0$. The δ is called *the translation parameter*. The result of the second step is a set of training mappings for $i \in \{1, \ldots, 2l\}$

$$\begin{cases} \boldsymbol{c_i} = \left(b_i^1, \ldots, b_i^m, y_{\mathrm{r}}^i + \delta\right) \rightarrow 1 & \text{for } \; i \in \{1, \ldots, l\} \\ \boldsymbol{c_i} = \left(b_i^1, \ldots, b_i^m, y_{\mathrm{r}}^{i-l} - \delta\right) \rightarrow -1 & \text{for } \; i \in \{l+1, \ldots, 2l\} \end{cases}$$

for $\delta \geq 0$. The dimension of the $\boldsymbol{c_i}$ vectors is equal to $m + 1$. The set of $\boldsymbol{x_i}$ mappings is called *a regression data setting*, the set of $\boldsymbol{c_i}$ ones is called *a classification data setting*. In the third step, OP 2 is solved with $\boldsymbol{c_i}$ examples. Note that $h^*(\boldsymbol{x})$ is in the implicit form of the last coordinate of $\boldsymbol{x}$. In the fourth step, an explicit form of the last coordinate needs to be find. The explicit form is needed for example for testing new examples. The $\boldsymbol{w_c}$ variable of the primal problem for a simple linear kernel is found based on the solution of the dual problem in the following way

$$\boldsymbol{w_c} = \sum_{i=1}^{2l} y_{\mathrm{c}}^i \alpha_i \boldsymbol{c_i} \ .$$

where $y_{\mathrm{c}}^i = 1$ for $i \in \{1, \ldots, l\}$ and $y_{\mathrm{c}}^i = -1$ for $i \in \{l+1, \ldots, 2l\}$. For a simple linear kernel the explicit form of (4) is

$$x^{m+1} = \frac{-\sum_{j=1}^{m} w_{\mathrm{c}}^j x^j - b_{\mathrm{c}}}{w_{\mathrm{c}}^{m+1}} \ .$$

The regression solution is $g^*(\boldsymbol{x}) = \boldsymbol{w_{\mathrm{r}}} \cdot \boldsymbol{x} + b_{\mathrm{r}}$, where $w_{\mathrm{r}}^i = -w_{\mathrm{c}}^i / w_{\mathrm{c}}^{m+1}$, $b_{\mathrm{r}} = -b_{\mathrm{c}} / w_{\mathrm{c}}^{m+1}$ for $i = 1..m$. For nonlinear kernels, a conversion to the explicit form has some limitations. First, a decision curve could have more than one value of the last coordinate for specific values of remaining coordinates of $\boldsymbol{x}$ and therefore it cannot be converted unambiguously to the function (e.g. a polynomial kernel with a dimension equals to 2). Second, even when the conversion to the function is possible, there is no explicit analytical formula (e.g. a polynomial kernel with a dimension greater than 4), or it is not easy to find it and hence a special method for finding an explicit formula of the coordinate should be used, e.g. a bisection method. The disadvantage of this solution is a longer time of testing new examples. To overcome these problems, a new kernel type in which the last coordinate is placed only inside a linear term was proposed [8]. A new kernel is constructed from an original kernel by removing the last coordinate, and adding the linear term with the last coordinate. For the most popular kernels polynomial, RBF and sigmoid, the conversions are respectively

$$(\boldsymbol{x} \cdot \boldsymbol{y})^d \ \rightarrow \ \left(\sum_{i=1}^{m} x_i y_i \right)^d + x_{m+1} y_{m+1} \ , \tag{5}$$

$$\exp - \frac{\|\boldsymbol{x} - \boldsymbol{y}\|^2}{2\sigma^2} \ \rightarrow \ \exp - \frac{\sum_{i=1}^{m} (x_i - y_i)^2}{2\sigma^2} + x_{m+1} y_{m+1} \ , \tag{6}$$

$$\tanh \boldsymbol{x}\boldsymbol{y} \ \rightarrow \ \tanh \sum_{i=1}^{m} x_i y_i + x^{m+1} y^{m+1} \ , \tag{7}$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are $m+1$ dimensional vectors. For the new kernel type, the explicit form of (4) for δ-SVR is

$$x^{m+1} = \frac{-\sum_{i=1}^{2l} y_{\mathrm{c}}^i \alpha_i K_{\mathrm{r}} (\boldsymbol{b_i}, \boldsymbol{x_r}) - b_c}{\sum_{i=1}^{2l} y_{\mathrm{c}}^i \alpha_i c_i^{m+1}} \ , \tag{8}$$

where $\boldsymbol{x_r} = \left(x^1, \ldots, x^m\right)$.

1.3 Introduction to Detractors for SVC, ε-SVR and δ-SVR

Detractors for SVC were introduced in [6]. *A detractor example* is defined as a point with the additional weight d and defined class (1 or -1). The SVC optimization problem with additional weights $\boldsymbol{\varphi}$ where $\boldsymbol{d} = 1 + \boldsymbol{\varphi}$ is

OP 3. *Minimization of*

$$f(\boldsymbol{w_c}, b_c, \boldsymbol{\xi_c}) = \frac{1}{2} \|\boldsymbol{w_c}\|^2 + \boldsymbol{C_c} \cdot \boldsymbol{\xi_c}$$

with constraints $y_c^i h(\boldsymbol{a_i}) \geq 1 - \xi_c^i + \varphi_i,\ \boldsymbol{\xi_c} \geq 0$ *for* $i \in \{1..l\}$, *where* $\boldsymbol{C_c} \gg 0$, $\varphi_i \in \mathbb{R}$, $h(\boldsymbol{a_i}) = \boldsymbol{w_c} \cdot \boldsymbol{a_i} + b_c$.

The new weights $\boldsymbol{\varphi}$ are only present in constraints. When $\boldsymbol{\varphi} = 0$, the OP 3 is equivalent to the OP 2. A functional margin for a point $\boldsymbol{p}$ is defined as a value $y_{\boldsymbol{p}} h(\boldsymbol{p})$. A value v in functional margin units is equal to $v / \|\boldsymbol{w_c}\|$. We can easily verify that a detractor parameter for $\varphi_i \geq 0$ is a lower bound on a distance from the detractor example to a decision boundary measured in functional margin units.

δ-SVR is based on solving a classification problem, therefore we can use detractors from classification case for δ-SVR [8]. When the class of the detractor is 1, for original examples we set $d_i = d_i + \delta_i$, for duplicated examples δ_i is unchanged; when a class is -1, $d_i = d_i + \delta_i$ for the duplicates, for the original examples d_i is unchanged.

Detractors for $\varphi_i > 0$ have a possibility to pushing away a decision function. After setting a detractor it is possible that the function remains the same, because either $\|\boldsymbol{w_c}\|$ is decreased, or ξ_i is adjusted to fulfill the constraint, e.g. when the new position of the decision curve candidate causes increased sum of slack variables. Therefore they have ability to change the function depending on prediction performance.

2 Introduction to Volume Participation Strategy

In this section, we present the model of the strategy of executing orders preceded by some definitions and theorems regarding VWAP measure which we will use later. First, we introduce some notation: T is the time period for executing the order, in conducted experiments it is a duration of the one session, n is the number of trades during T, $v(i)$ is a volume of the i-th trade, v is a market volume in T, $p(i)$ is a price of the i-th trade. We have

$$v = \sum_{i=1}^{n} v(i) \ .$$

Definition 1. *Market VWAP is*

$$VWAP = \frac{\sum_{i=1}^{n} p(i)\, v(i)}{v} \ .$$

For a volume of the order in T, v_0, we have

$$v_0 = \sum_{i=1}^{n} v_0(i) \ .$$

where $v_0(i)$ is a part of the order volume belongs to the i-th trade.

Definition 2. *Order VWAP is*

$$VWAP_0 = \frac{\sum_{i=1}^{n} p(i) v_0(i)}{v_0}$$

In the presented strategy we divide T to some time slices. Below we list some propositions regarding time slices.

Proposition 1. *Assuming that the volume is divided to two parts with known VWAP for these parts ($VWAP_1$ and $VWAP_2$) and known volumes (v_1 and v_2 respectively), overall VWAP is*

$$VWAP = \frac{VWAP_1 v_1 + VWAP_2 v_2}{v_1 + v_2}$$

We can generalize this proposition to multiple parts, e.g. multiple time slices, we can divide T to m parts, aggregated volume from all trades in the i-th part is noted as $v(T_i)$, $VWAP$ for all trades in the i-th part is noted as $VWAP(T_i)$, aggregated volume of the order in the i-th part is noted as $v_0(T_i)$. Then a market volume in T is

$$v = \sum_{i=1}^{m} v(T_i) \ .$$

market VWAP in T is

$$VWAP = \frac{\sum_{i=1}^{m} VWAP(T_i) v(T_i)}{v} \ .$$

A volume of the order in T is

$$v_0 = \sum_{i=1}^{m} v_0(T_i) \tag{9}$$

and order VWAP in T is

$$VWAP_0 = \frac{\sum_{i=1}^{m} VWAP_0(T_i) v_0(T_i)}{v_0} \ . \tag{10}$$

In this article, we investigate a problem of developing a strategy which optimizes the ratio of order VWAP to market VWAP for future trades

$$\frac{VWAP_0}{VWAP} = \frac{\sum_{i=1}^{n} p(i) v_0(i)}{v_0} \frac{v}{\sum_{i=1}^{n} p(i) v(i)} \ . \tag{11}$$

We can reformulate (11) by substituting

$$v_0 = V_1 v \tag{12}$$

and we get

$$\frac{VWAP_0}{VWAP} = \frac{\sum_{i=1}^{n} p(i) v_0(i)}{V_1 \sum_{i=1}^{n} p(i) v(i)} \tag{13}$$

where V_1 is defined as a ratio of order volume to market volume

$$V_1 = \frac{v_0}{v} \ . \tag{14}$$

For m time slices we get

$$\frac{VWAP_0}{VWAP} = \frac{\sum_{i=1}^{m} VWAP\left(T_i\right) v_0\left(T_i\right)}{V_1 \sum_{i=1}^{m} VWAP\left(T_i\right) v\left(T_i\right)} \ . \tag{15}$$

For buy orders we would like to minimize this ratio, for sell orders maximize. Particularly, the goal is to achieve the ratio equal or less than 1 for buy orders and equal or greater than 1 for sell orders. Note that a challenge in optimizing this ratio is that future volume and/or future prices have to be predicted. First, we will present a strategy which achieves the ratio equal to 1 by predicting volume participation. Second, we will present an extension of this strategy which allows to incorporate information about prices. Such separation is desirable, because we can compute the error for prediction based on volume, and the error for price prediction.

2.1 Volume Participation Strategy

Here, we describe a model of the strategy which achieves VWAP ratio equal to 1 without assuming any price information. The strategy is to trade with a predicted volume. It means that for every time slice T_i we have

$$v_0\left(T_i\right) = V_1 v\left(T_i\right) = \frac{v_0}{v} v\left(T_i\right) \ . \tag{16}$$

We can see that the strategy fulfills (9). We can reformulate it

$$v_0\left(T_i\right) = \frac{v\left(T_i\right)}{v} v_0 = r\left(T_i\right) v_0 \tag{17}$$

where

$$r\left(T_i\right) = \frac{v\left(T_i\right)}{v} \ .$$

The r is called *volume participation*, Fig. 1. We can easily check that for this strategy (12) is fulfilled. After substituting (16) to (15) we get the ratio equals to 1.

In order to use this strategy in practice we have to predict volume participation $r\left(T_i\right)$ (17) for every time slice and try to trade at $VWAP\left(T_i\right)$ inside every time slice. Note that it would be possible to use (16) instead of (17), but then we would need to predict volume v.

2.2 Errors for Volume Participation Strategy

There are two possible sources of errors in realizing this strategy. The first error ε_1 is related to trading with $VWAP\left(T_i\right)$, the second error ε_2 is related to

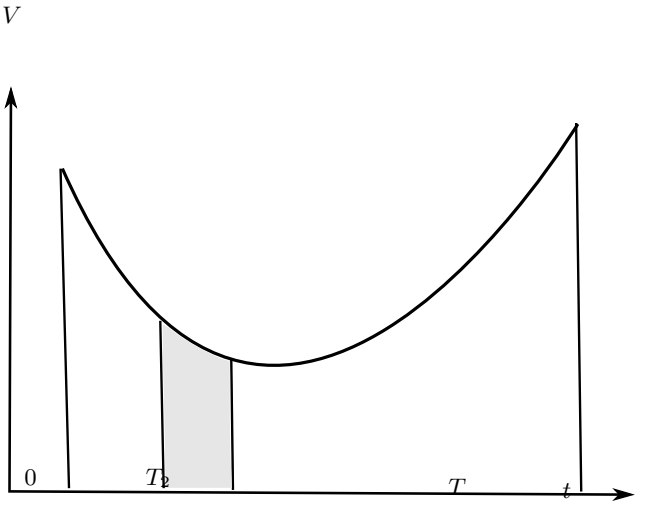


Fig. 1. Visualization of volume participation. Volume participation for T_2 is interpreted as a ratio of gray area to the whole area below volume from 0 to T.

predicting volume participation in T_i, after substituting (17) to (10) and considering the errors

$$VWAP_0 = \sum_{i=1}^{m} \left(VWAP\left(T_i\right) + \varepsilon_1\left(T_i\right)\right)\left(r\left(T_i\right) + \varepsilon_2\left(T_i\right)\right) \ .$$

While comparing $VWAP$ to $VWAP_0$ we get the following error

Theorem 1

$$\varepsilon = \frac{VWAP_0}{VWAP} - 1 = \frac{\sum_{i=1}^{m} \varepsilon_1\left(T_i\right) r\left(T_i\right)}{\sum_{i=1}^{m} VWAP\left(T_i\right) r\left(T_i\right)} + \frac{\sum_{i=1}^{m} \varepsilon_2\left(T_i\right) VWAP\left(T_i\right)}{\sum_{i=1}^{m} VWAP\left(T_i\right) r\left(T_i\right)}$$

$$+ \frac{\sum_{i=1}^{m} \varepsilon_1\left(T_i\right) \varepsilon_2\left(T_i\right)}{\sum_{i=1}^{m} VWAP\left(T_i\right) r\left(T_i\right)}$$

In this article, we are interested mainly in optimizing ε_2. We generate a priori values of E_1 where $\varepsilon_1\left(T_i\right) = E_1 VWAP\left(T_i\right)$. For comparing performance of various strategies which depend on ε_2 it is enough to compare

$$\sum_{i=1}^{m} \varepsilon_2\left(T_i\right) VWAP\left(T_i\right)\left(1 + E_1\right) \ . \tag{18}$$

Instead of $\varepsilon_2\left(T_i\right)$ we will use $\operatorname{sgn}\left(\varepsilon_2\right)\varepsilon_2^2\left(T_i\right)$, because SVR grid search method finds the best values of SVM parameters comparing that error. Lowering ε_1 leads to a lower variance of (18).

Comparison to Time Weighted Average Price (TWAP) strategy. TWAP strategy trades the same quantity in every time slice. TWAP can be interpreted as the volume participation strategy with predicted volume as a constant function. We expect worser prediction of volume participation for TWAP, therefore greater value of ε_1 compared to the VWAP strategy, so we expect greater variance of ε for TWAP method.

2.3 Predicting Volume Participation

In order to use Volume Participation Strategy we need to predict volume participation $r(T_i)$ for all time slices. In this article, we investigated four methods of prediction, the first one arbitrarily assumes that a volume is a constant function, so a volume participation function is also a constant one (it is used in TWAP strategy), the second one predicts $r(T_i)$ as an average value from previous days, it is a kind of local strategy. The third one predicts $r(T_i)$ as $r(T_{i-1})$ from the previous time slice and the last one predicts volume participation $r(T_i)$ from historical data by assuming that $r(\cdot)$ is a continuous function. For the last one we use SVR methods.

For SVR prediction we propose only one feature that is the id of the time slice, so the feature is a discrete one. The output of the method is a volume participation. In the future, we plan to include additional features to potentially increase performance of the prediction such as a day of the week, a day of the month, previous values, etc.

Note that the prediction problem has an additional constraint that the sum of volume participation during the session must be equal to 1

$$\sum_{i=1}^{m} r(T_i) = 1 \ . \tag{19}$$

For the TWAP predictor, it is fulfilled out of hand. For the remaining predictors it might not be fulfilled. Therefore, for average and previous predictors we propose proportional adjustment for every time slice. For SVR predictor we propose adjusting a free term b to fulfill (19). In the future, we plan to add (19) directly to the (12) as an additional constraint.

3 Incorporating A Priori Knowledge about Prices

Volume Participation Strategy achieves the ratio equal to 1 in the presented model. It is possible to achieve better execution performance by taking into account price prediction. The general idea of an improvement is to increase order volume when the predicted price is relatively low during the session for buy orders (relatively high for sell orders).

There are two problems concerning manipulating a participation function based on price prediction. First is in achieving enough price prediction performance for improving the error ε. Second, that increased order volume for

some time slices could change noticeably the prices during the next sessions (it is called *market impact*) and additionally decrease price prediction performance.

Because price prediction is a challenging task, we propose to incorporate simple price prediction rules, such as *in the second part of the session prices will be generally higher than in the first one (or vice versa)*. For this rule we might want to increase participation in the first half of the session, and decrease in the second one (for buy orders). The simple way of incorporating such knowledge is to increase participation by some value e.g. $p = 0.1$ for the first part of the session and decrease by the same value in the second part of the session (assuming an even number of time slices). The problem with this solution is that participation rate is not smooth in the half of the session. The second issue is that participation change by the same value in the first part and the second. We cannot improve participation changes by using price information, because we have just only simple prediction rules. So we propose to set participation changes based on volume participation prediction performance. We want to increase value and chance of changing p for time slices with worser volume participation prediction performance, and decrease value of p for the rest. For this purpose, we use extended SVM predictors with a technique of detractors a priori knowledge introduced for SVC in [6,7], for ε-SVR in [9] and for δ-SVR in [8]. The technique was used for manipulating classification boundaries [6] and regression functions [8]. It posses a desired property of adjusting the output function depending on the prediction performance.

3.1 Detractors in Practice

We divide the period T to 2 periods, first half of the session and the second. We propose setting a detractor parameter $\varphi_i = r$ for all training examples, where r is a configurable parameter. When we expect that prices will be higher in the second part of the session, for every detractor from the first part of the session we set -1 class, and for the second part we set 1 class (in reverse for opposite prediction).

4 Experiments

We divide experiments into three parts: in the first part we compare prediction performance of SVM with null hypotheses. In the second experiment, we compare (18) for SVM and null hypotheses. We compare prediction performance of SVM with the following null hypotheses: prediction based on constant function, prediction based on average participation from historical data for the same time slice and prediction from the previous time slice. In the third experiment, we compare (18) for δ-SVR and δ-SVR with incorporated a priori knowledge in the form of detractors.

For solving ε-SVR and SVC for particular parameters we use LibSVM [3] ported to Java. Data which are used for experiments are tick data for securities from NASDAQ-100 index for about a half year period (from 01.01.2011 to

20.05.2011) which were compressed to a desired size of time slices. Data includes trades from opening and closing crosses. For all data sets, every feature is scaled linearly to $[0, 1]$. The results are averaged for all tested instruments. For variable parameters like the C, σ for the RBF kernel, δ for δ-SVR, and ε for ε-SVR, we use a double grid search method for finding the best values. We use modified double cross validation with shifting data. Inner cross validation is used for finding the best values of the variable parameters. Instead of standard outer cross validation, we shift data. Hence, the validation set is always after the training set. We use a fixed size for the training set, that is 2 weeks, and for the validation set 1 week.

4.1 Prediction Performance and (18) Comparison

We compare δ-SVR and ε-SVR with null hypotheses. Results are presented in Table 1. Because ε_1 is related to trading performance in every time slice, we generate normally distributed E_1 as $E_1 = 0.02N(0, 1)$. We performed tests for half hour slices.

We achieve better generalization performance for ε-SVR than for all null hypotheses. The average null hypothesis is the most competitive comparing to SVR, we achieve slightly better generalization performance for SVR but without significant difference based on t-test. Comparing a variance of (18) we can

Table 1. Description of test cases with results for prediction performance comparison of δ-SVR with null hypotheses. Column descriptions: *id* – an id of a test, *a name* – a name of the test, δ-SVR compared with hypotheses 1 or 2 or 3, *ts* – a size of time slice (in minutes), *simT* – a number of shifts, results are averaged, *ker* – a kernel (*pol* – a polynomial kernel), *kerP* – a kernel parameter (for a polynomial kernel it is a dimension, for the RBF kernel it is σ), *trs* – a training set size for every stock, *all* – a number of all data for every stock, *dm* – a dimension of the problem, *tr12M* – a percent average difference in MSE for training data, if greater than 0 than SVM is better, *te12M* – the same as tr12M, but for testing data, *teT* – t value for the t-test for comparing testing error, *e12M* – comparison of a variance of (18). The value 'var' means that we search for the best value.

id	name	ts	simT	ker	kerP	trs	all	dm	tr12M	te12M	teT	e12M
1	δ-SVRvsH1	30m	5	lin	-	130	1075	1	1.9%	−0.2%	−0.08	41%
2	ε-SVRvsH1	30m	5	lin	-	130	1075	1	3.2%	1.6%	0.65	24%
5	δ-SVRvsH1	30m	5	rbf	0.1	130	1075	1	70%	68%	35.3	92%
6	ε-SVRvsH1	30m	5	rbf	0.1	130	1075	1	69.7%	68.5%	35	91%
11	δ-SVRvsH2	30m	5	rbf	0.1	130	1075	1	−8.8%	3.5%	1.0	−60%
12	ε-SVRvsH2	30m	5	rbf	0.1	130	1075	1	−7.7%	4.5%	1.33	−22%
13	δ-SVRvsH3	30m	5	lin	-	130	1075	1	19.4%	20.4%	7.9	−90%
14	ε-SVRvsH3	30m	5	lin	-	130	1075	1	20.7%	22.1%	8.5	−30%
17	δ-SVRvsH3	30m	5	rbf	0.1	130	1075	1	75%	75%	32.5	80%
18	ε-SVRvsH3	30m	5	rbf	0.1	130	1075	1	75%	75%	32.5	77%

see that not always better generalization performance leads to better variance, for the first and some of third hypotheses SVR is better, for average hypothesis SVR is worse.

4.2 Final Execution Error Comparison for A Priori Knowledge

We compare (18) for δ-SVR with incorporated a priori knowledge about prices, and without. The scope of this article does not include testing the effectiveness of the price prediction. Therefore we propose the following procedure for generating a priori knowledge about prices, we check in advance on historical data whether market VWAP will be higher in the first part of the session, or in the second one. We define detractor points with class -1 and values r for δ-SVR for the data points from the first part of the session or the second accordingly, r value is chosen arbitrarily to 0.5. Results are presented in Table 2.

The results show that volume participation prediction performance could be worser after manipulating the function, but we can see significant improvement in (18) for the modified solution. SVR with additional a priori knowledge about prices achieves better execution performance than without a priori knowledge.

Table 2. Description of test cases with results for comparing δ-SVR with δ-SVR with a priori knowledge. Column descriptions: *id* – an id of a test, *ts* – a size of time slice (in hours), *simT* – a number of shifts, results are averaged, *ker* – a kernel (*pol* – a polynomial kernel), *kerP* – a kernel parameter (for a polynomial kernel it is a dimension, for the RBF kernel it is σ), *trs* – a training set size for every stock, *all* – a number of all data for every stock, *dm* – a dimension of the problem, r – a detractor value, *tr12M* – a percent average difference in MSE for training data, if greater than 0 than SVM is better, *te12M* – the same as tr12M, but for testing data, *teT* – t value for the t-test for comparing testing error, *e12M* – comparison of (18), *eT* – t-value for comparing (18). The value 'var' means that we search for the best value.

id	ts	simT	ker	kerP	trs	all	dm	r	tr12M	te12M	teT	e12M	eT
22	30m	5	rbf	0.1	130	1075	1	1	−5%	−6%	−1.7	19%	2.4

5 Conclusions

We proposed an efficient application for SVR for constructing order execution strategies. Additionally, we showed how to incorporate a priori knowledge about prices to the model. The future work will include analysis of the possibility of achieving VWAP price in every time slice (optimizing ε_1) by taking into account microstructure of the market.

Acknowledgments. I would like to express my sincere gratitude to Professor Witold Dzwinel (AGH University of Science and Technology, Department of Computer Science) and Josef Holzer (Merkursoft Sp. z o.o.) for contributing ideas, discussion and useful suggestions.

A Proof of Thm. 1

Proof

$$VWAP_0 = \frac{\sum_{i=1}^{m} \left(VWAP\left(\Delta t_i\right) + \varepsilon_1\left(\Delta t_i\right)\right)\left(v_0\left(r\left(\Delta t_i\right) + \varepsilon_2\left(\Delta t_i\right)\right)\right)}{v_0}$$

$$VWAP_0 = \sum_{i=1}^{m} \left(VWAP\left(\Delta t_i\right) + \varepsilon_1\left(\Delta t_i\right)\right)\left(r\left(\Delta t_i\right) + \varepsilon_2\left(\Delta t_i\right)\right)$$

$$\frac{VWAP_0}{VWAP} - 1 = \frac{v\sum_{i=1}^{m} \left(VWAP\left(\Delta t_i\right) + \varepsilon_1\left(\Delta t_i\right)\right)\left(r\left(\Delta t_i\right) + \varepsilon_2\left(\Delta t_i\right)\right)}{\sum_{i=1}^{m} VWAP\left(\Delta t_i\right) v\left(\Delta t_i\right)} - 1 =$$

$$= \frac{\sum_{i=1}^{m} \left(VWAP\left(\Delta t_i\right) + \varepsilon_1\left(\Delta t_i\right)\right)\left(r\left(\Delta t_i\right) + \varepsilon_2\left(\Delta t_i\right)\right)}{\sum_{i=1}^{m} VWAP\left(\Delta t_i\right) r\left(\Delta t_i\right)} - 1 =$$

$$= \frac{\sum_{i=1}^{m} \varepsilon_1\left(\Delta t_i\right) r\left(\Delta t_i\right)}{\sum_{i=1}^{m} VWAP\left(\Delta t_i\right) r\left(\Delta t_i\right)} + \frac{\sum_{i=1}^{m} \varepsilon_2\left(\Delta t_i\right) VWAP\left(\Delta t_i\right)}{\sum_{i=1}^{m} VWAP\left(\Delta t_i\right) r\left(\Delta t_i\right)}$$

$$+ \frac{\sum_{i=1}^{m} \varepsilon_1\left(\Delta t_i\right) \varepsilon_2\left(\Delta t_i\right)}{\sum_{i=1}^{m} VWAP\left(\Delta t_i\right) r\left(\Delta t_i\right)}$$

References

1. Bialkowski, J., Darolles, S., Le Fol, G.: Improving vwap strategies: A dynamic volume approach. Journal of Banking & Finance 32(9), 1709–1722 (2008)
2. Brownlees, C.T., Cipollini, F., Gallo, G.M.: Intra-daily volume modeling and prediction for algorithmic trading. Econometrics working papers archive, Universita' degli Studi di Firenze, Dipartimento di Statistica "G. Parenti" (February 2009)
3. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines, software (2001), http://www.csie.ntu.edu.tw/~cjlin/libsvm
4. Hobson, D.: Vwap and volume profiles. Journal of Trading 1(2), 38–42 (2006)
5. Lin, F., Guo, J.: A novel support vector machine algorithm for solving nonlinear regression problems based on symmetrical points. In: Proceedings of the 2010 2nd International Conference on Computer Engineering and Technology (ICCET), pp. 176–180 (2010)
6. Orchel, M.: Incorporating Detractors into SVM Classification. In: Cyran, K.A., Kozielski, S., Peters, J.F., Stańczyk, U., Wakulicz-Deja, A. (eds.) Man-Machine Interactions. AISC, vol. 59, pp. 361–369. Springer, Heidelberg (2009)
7. Orchel, M.: Incorporating a Priori Knowledge from Detractor Points into Support Vector Classification. In: Dobnikar, A., Lotrič, U., Šter, B. (eds.) ICANNGA 2011, Part II. LNCS, vol. 6594, pp. 332–341. Springer, Heidelberg (2011)
8. Orchel, M.: Regression Based on Support Vector Classification. In: Dobnikar, A., Lotrič, U., Šter, B. (eds.) ICANNGA 2011, Part II. LNCS, vol. 6594, pp. 353–362. Springer, Heidelberg (2011)
9. Orchel, M.: Support Vector Regression as a Classification Problem with a Priori Knowledge in the form of Detractors. In: Czachórski, T., Kozielski, S., Stańczyk, U. (eds.) Man-Machine Interactions 2. AISC, vol. 103, pp. 351–358. Springer, Heidelberg (2011)
10. Vapnik, V.N.: Statistical Learning Theory. Wiley-Interscience (September 1998)

Terrorist Organization Behavior Prediction Algorithm Based on Context Subspace

Anrong Xue, Wei Wang, and Mingcai Zhang

School of Computer Science and Telecommunication Engineering,
Jiangsu University, Zhenjiang 212013, China
xuear@ujs.edu.cn, wangwei1237@gmail.com, zhzhmc86@163.com

Abstract. Researchers have developed methods to predict a terrorist organization's probable actions (such as bombings or kidnappings). The terrorist organization's actions can be affected by the context of the organization. Thus, the organization's context variables can be used to improve the accuracy of forecasting the terrorist behavior. Those algorithms based on context similarity suffer a serious drawback that it can result in the algorithm's fluctuation and reduce the prediction accuracy if not all the attributes are detected. A prediction algorithm PBCS (Prediction Based on Context Subspace) based on context subspace is proposed in this paper. The proposed algorithm first extracts the context subspace according to the association between the context attributes and the behavior attributes. Then, it predicts the terrorist behavior based on the extracted context subspace. The proposed algorithm uses the improved spectral clustering method to obtain the context subspace. It concerns the distribution of the data samples, the label information and the local similarity of the data in the process of extracting the subspace. Experimental results on the artificial dataset and the MAROB dataset show that the prediction method proposed in this paper can not only improve the prediction accuracy but also reduce the prediction fluctuation.

Keywords: terrorism behavior prediction, context knowledge, context subspace, spectral analysis.

1 Introduction

The context of the terrorist group often affects the attack behavior taken by the terrorist group. Thus, making full use of the context of the terrorist group can improve the forecasting accuracy, and the policy-makers based on the prediction can take more effective measures.

As cultural modeling (CM) increasingly becomes an emergent research area in social computing, the context knowledge has been risen to predict the terrorist behavior [1-4]. The idea of cultural modeling was first proposed in [4], and Subrahmanian suggested that real-time computational models can use various cultural factors as context knowledge to help policy-makers predict the terrorist behavior [4].

J. Tang et al. (Eds.): ADMA 2011, Part II, LNAI 7121, pp. 332–345, 2011.

Then, the CARA (Cognitive Architecture for Reasoning about Adversaries) model, as the first concept model to forecast the terrorist behavior, was proposed in [5]. To further develop CARA model, the HOP (Head Oriented Processing) and SemiHOP algorithms, mainly based on probabilistic logic programming, were proposed by Khuller and his colleagues in [6]. Nevertheless, there are still some weaknesses in those two algorithms. For example, the improvement in computational efficiency is in sacrifice of the forecasting accuracy [7]. To better predict terrorist group behavior, a new algorithm based on the context-knowledge similarity called CONVEX was proposed in [8]. The CONVEX algorithm views each instance as a pair of two vectors, namely context vector and action vector. The context vector contains the values of the context variables associated with the terrorist group, while the action vector contains the values of action variables. The CONVEX algorithm performs well when all the context knowledge is detected, but not all the context variables can be identified in practice [7]. Those missing context variables should be calculated by certain data preprocess technology. Thus, it does not only increase the computation time but also produce bias. Secondly, as the context variables have different influences on the terrorist behavior, the missing variables may lead to the prediction fluctuation.

This paper firstly divides the whole context variables space into two subspaces according to the correlation between the context variables and the action variables. Then, it extracts the context variables which have stronger correlation as the subspace, and predicts the terrorist's behavior in the subspace to avoid the prediction fluctuation caused by the missing attributes.

As mentioned in section 2, there exits faults in some machine learning algorithms in terms of the context dataset, this paper proposes an improved spectral clustering algorithm, SCBAA (Spectral Clustering Based on Attributes' Association), to extract the subspace. The SCBAA algorithm can not only reduce the computational complexity but also concern the distribution of the data samples, the label information and the local similarity of the data. After extracting the context subspace, it predicts the terrorist behavior in the extracted subspace for the sake of avoiding the prediction fluctuation.

2 Characteristic of the Context Dataset

Generally, the context dataset of the terrorist group includes the historical behavior records of terrorist group. When a new organizational behavior or event happens, the data items capture current time, related context factors such as the related cultural and the current political situation [7-8]. An instance in the context dataset has the following form:

$$\mathbf{R} = (\mathbf{context}, \mathbf{action}) \tag{1}$$

In (1), the context vector contains the group's context knowledge such as the region of the terrorist group, the religion of this region, the region's economic situation, political situation, and so on. The action vector involves the violent behavior taken by the

terrorist group such as kidnap, arm attack, etc and non-violent behavior. The violent behavior variables are the behavior that this paper will predict, and the non-violent behavior variables are not contained in the action vector in this paper. Then, the two vectors have the following schema:

$$\mathbf{context} = (C_1, C_2, ..., C_M) \tag{2}$$

$$\mathbf{action} = (A_1, A_2, ..., A_N) \tag{3}$$

context$\cap$**action** $= \varnothing$, each variable in the two vectors such as C_j and A_i has an associated domain $dom(C_j)$ and $dom(A_i)$ separately. Hence, an instance in the context dataset is a pair comprising a context vector and an action vector, $((c_1,\ldots,c_M),(a_1,\ldots,a_N))$, where $c_j \in dom(C_j)$ and $a_i \in dom(A_i)$. Besides, the vectors satisfy the correlation-ship that the context variables $\{C_1, C_2, \ldots, C_M\}$ affects the existence of center terrorist behavior. Hence, it can extract the context subspace according to the correlation-ship.

The most machine learning algorithms require that the training samples are dense in the sample space, so as to obtain satisfactory performance. For example, for one-dimensional space, it can be considered to be dense if there are 100 samples in the unit interval. For the typical context dataset, MAROB dataset, if each instance in that dataset contains 20 context variables, it requires $100^{20}=10^{40}$ instances to reach the same density in this hypercube.

However, the frequency of the terrorist behavior is low. In most cases, for a given organization, there are only dozens of records each year. Thus, the number of samples is small. For example, the MAROB dataset only has 2000 records more or less. Thus, in the context variables space, it is difficult to linearly represent a point in its neighborhood. Secondly, in such a sparse situation, the data coverage between different neighborhoods is hard to find. Finally, the instances in the context dataset are discrete, so they don't satisfy linear relationship.

For certain terrorist behavior that in the action vector, not all of the context variables have the same effect. And the context variables that have weak correlation could not be detected in practice. Then, it does not only increase the computation time but also lead to the prediction fluctuation. Thus, it should extract a subspace from the whole context variables for different behaviors. And then, it predicts the behavior in the extracted subspace to avoid the prediction fluctuation.

The PCA (principal component analysis) algorithm can extract the principal component of the context variables as the subspace. But the data in the context dataset doesn't satisfy the linear relationship and cannot be embed in a linear similar of local low-dimensional sub-manifold. Thus, the PCA algorithm has difficulty in extracting the subspace of the context dataset [9-10]. The LLE (Locally Linear Embedding) algorithm can achieve the dimensionality reduction by using the data coverage between different neighborhoods, and then extract the subspace. However, it is difficult to find data coverage in such a sparse space. So, the LLE couldn't be used to extract the subspace [9],[11]. The kernel methods can resolve the non-linear feature of the context dataset, but there are no guidelines to choose the kernel function and its parameters [9]. As a non-linear dimensionality reduction algorithm, the spectral clustering algorithm is suitable for any shape of the sample space. In addition, after

dimensionality reduction, the local connection characteristics of the sample space are consistent with the original space. It can also converge to the global optimal solution in any sample space [12-13]. Thus, the spectral clustering algorithm is used as the subspace-extracted algorithm in this paper.

3 Preliminaries of Spectral Clustering

Transferring the clustering problem to the graph partitioning problem is the mathematical theory of the spectral clustering algorithm. Its aim is to divide the data into several sub-graphs based on the theory of the optimal division of graph. The optimal division should have the highest within-class similarity and the lowest inter-class similarity. So, after the graph division by the spectral clustering algorithm, it can extract the sub-graph that has the stronger correlation with certain action as its subspace.

Given a data set $\mathbf{X}$ that has N records $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$, $\mathbf{x}_i \in \Re^d$. Let $G=<V,E>$ expresses a undirected graph, and $V=\{\mathbf{x}_1, \mathbf{x}_2,\ldots,\mathbf{x}_N\}$ is the set of the vertexes, $E=\{e_{ij}=(\mathbf{x}_i,\mathbf{x}_j) | 1\le i, j\le N \wedge i\ne j\}$ is the set of the edges of the graph. If the graph G is a weighted graph, then each e_{ij} has a non-negative weighted a_{ij}. a_{ij} is the similarity between the vertex $\mathbf{x}_i$ and the vertex $\mathbf{x}_j$, and $a_{ij}=0$ if $i=j$.

In order to measure the effect of division, make definition as follows:

$$W(V_1,V_2) = \sum_{\mathbf{x}\in V_1,\mathbf{x}'\in V_2} a(\mathbf{x},\mathbf{x}') \tag{4}$$

In (4), V_1, V_2 are the two sub-graphs after division. Thus, the target function of the optimal division of graph is:

$$Ncut = \sum_{i=1}^{2} \frac{W(V_i,V) - W(V_i,V_i)}{W(V_i,V)} = 2 - \sum_{i=1}^{2} \frac{W(V_i,V_i)}{W(V_i,V)} \tag{5}$$

The optimal division of graph makes the $W(V_i,V_i)$ get maximum and $W(V_i,V)$ obtain minimum, it must satisfy (6).

$$(V_1,V_2)^* = \min Ncut(V) \tag{6}$$

According to the Raleigh-Ritz theory [12], it can handle the problem of (6) through the spectral analysis of the matrix $\mathbf{L}$ as shown in (7).

$$\mathbf{L} = \mathbf{D}^{-1/2}(\mathbf{D}-\mathbf{A})\mathbf{D}^{-1/2} \tag{7}$$

Where, $\mathbf{A} = [a_{ij}]_{i,j=1}^{N}$, and matrix $\mathbf{D}$=diag($d_1,d_2,\ldots,d_N$) is the degree matrix of the graph, $d_i = \sum_{j=1}^{N} a_{ij}$, $i = 1\ldots N$.

4 Prediction Algorithm Based on the Context Subspace

4.1 Extract the Context Subspace

Given a dataset **X** that has *N* records $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$, $\mathbf{x}_i \in \Re^d$. The traditional spectral clustering should form an *N***N* affinity matrix and compute eigenvectors of this matrix, an operation that has a computational complexity of $O(N^3)$. As for the special dataset, in order to reduce the algorithm's computational complexity, this paper proposes an improved spectral clustering algorithm called SCBAA. The SCBAA algorithm doesn't construct the affinity matrix by the similarity among the instances. It constructs the affinity matrix by the correlation between C_j and A_i, where $C_j \in$ **context** vector and $A_i \in$ **action** vector. For the SCBAA algorithm, the order of the affinity matrix will not increase with the data records' increasing. Thus, the algorithm's computational complexity will not increase.

To construct the affinity matrix of the C_j according to A_i ($C_j \in$ **context** vector and $A_i \in$ **action** vector), the SCBAA algorithm firstly calculates the distance (d) between C_j and A_i. Then, it calculates the similarity (a_{ij}) between C_i and C_j. Finally, the SCBAA algorithm obtains the affinity matrix of the context variables by a_{ij}.

By using the label information that the dataset provided, the SCBAA algorithm uses (9) to calculate the distance between C_j and A_i.

$$\rho(C_j, A_i) = (\sum_{k=1}^{N} (c_j \oplus a_i)) / N \tag{8}$$

$$d(C_j) = 1 / \rho(C_j, A_i) \tag{9}$$

$$\oplus(x_1, x_2) = \begin{cases} 1; both\ or\ neither\ x_1\ and\ x_2\ occurce \\ 0; otherwise \end{cases} \tag{10}$$

Where $c_j \in dom(C_j)$, $a_i \in dom(A_i)$. As shown in (8), if C_j and A_i occur simultaneously, the correlation between them is stronger. What is more, the stronger correlation between C_j and A_i is, the shorter distance between C_j and A_i is.

The SCBAA algorithm concerns the local similarity of C_j as (11) shows.

$$a_{ij} = \mathrm{e}^{-\frac{(d(C_i) - d(C_j))^2}{2\sigma^2}} \tag{11}$$

And σ is the standard deviation of the data samples.

As shown in (11), it makes use of the distance between the context variables and a specific behavior to calculate the similarity among the context variables. The greater the $(d(C_i)\text{-}d(C_j))^2$ is, the weaker correlation between C_i and C_j is.

The SCBAA algorithm is described as follows.

SCBAA algorithm:
Input: The dataset $\mathbf{X}=\{\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_N\}$, and $\mathbf{x}_i=(c_1,c_2,\ldots,c_M,\mathbf{action})$, $\mathbf{action}=(a_1,a_2,\ldots,a_N)$.
Output: The extracted subspace S_i for the $A_i \in \mathbf{action}$ vector.
Step:

1) Calculate the correlation $\rho(C_j, A_i)$ between C_j and A_i with formula (8), $C_j \in \mathbf{context}$ vector;
2) Calculate the distance $d(C_j, A_i)$ with formula (9);
3) Construct the undirected graph of $C_j \in \mathbf{context}$ vector, and calculate a_{ij} with formula (11);
4) Construct the affinity matrix $\mathbf{A}$ with a_{ij};
5) Get matrix $\mathbf{L}$ as the Laplacian matrix;
6) Calculate the spectral of matrix $\mathbf{L}$ with formula (7), and then calculate the Fiedler vector: $\mathbf{F}=[f_1,f_2,\ldots,f_M]^T$;
7) Get the division $S=\{C_i : f_i<0\}$, $\overline{S}=\{C_i : f_i>0\}$ according to the Fielder vector;
8) Extract the context variables that are in the S set as the subspace to the A_i;
9) Get the other actions' subspace as the step 1)- 8).

The number of the context variables of the terrorist group may not change. Thus, there are not any other additional context variables which can be added to the dataset. It is the weight in the graph that could be changed if the dataset is enlarged. The vertex of the graph will not change. In this case, the algorithm can obtain the new weight between vertexes by scanning the dataset from the first record to the last. Then it calculates the new weight with the new data record. Thus, the computational complexity of the SCBAA algorithm is $O(N)$. As shown in (11), the SCBAA algorithm concerns the label information of the dataset. In addition, the local similarity of the data is also considered in the process of extracting the subspace.

4.2 Prediction Algorithm Based on the Extracted Subspace

As for the context dataset, not all the context variables have the same correlation with the different terrorist behaviors. For certain behavior, it doesn't detect the context variables that have weak correlation in practice. Hence, it will lead to the prediction fluctuation. Given the detected values of the context variables $\mathbf{q}=(c_1,c_2,\ldots c_M)$, it should extract the subspace according to the different behavior by using SCBAA algorithm, and then predict the occurrence probability of the terrorist behavior.
The missing attributes in the extracted subspace will influence the forecasting result. Thus, this paper uses (12) to perform the prediction.

$$f(\mathbf{x}') = f(\mathbf{x})g(C_i) \tag{12}$$

In (12), $\mathbf{x}'=[\mathbf{x}^T, C_i]^T$, $\mathbf{x}=[C_1,C_2,\ldots,C_j]^T$, $C_i \notin \mathbf{x}$, $f(\mathbf{x})$ is the prediction result by the context attributes $\mathbf{x}$. The (12) shows that the final prediction result is calculated by

updating the fore-prediction result with the newly detected attribute C_i. Assuming that $y=A_i$, we can get (13) and (14) with conditional probability theory.

$$\begin{aligned} P(y \mid \mathbf{x}^{\mathrm{T}}, C_i) &= P(y, \mathbf{x}^{\mathrm{T}}, C_i) / P(\mathbf{x}^{\mathrm{T}}, C_i) \\ &= P(y) P(\mathbf{x}^{\mathrm{T}}, C_i \mid y) / P(\mathbf{x}^{\mathrm{T}}, C_i) \end{aligned} \tag{13}$$

$$P(y \mid \mathbf{x}^{\mathrm{T}}, C_i) = \frac{P(y) P(\mathbf{x}^{\mathrm{T}} \mid y) P(C_i \mid y)}{P(\mathbf{x}^{\mathrm{T}}) P(C_i)} \tag{14}$$

As for formula (14), it can be transformed to the following formula:

$$P(y \mid \mathbf{x}^{\mathrm{T}}, C_i) = P(y \mid \mathbf{x}^{\mathrm{T}}) P(C_i \mid y) / P(C_i) \tag{15}$$

Finally, we obtain (16).

$$f(\mathbf{x}'^{\mathrm{T}}) = f(\mathbf{x}^{\mathrm{T}}) P(C_i \mid y) / P(C_i) \tag{16}$$

In (16), $P(C_i|y)$ is the posterior probability of C_i conditioned on y, $P(C_i)$ is the prior probability of C_i.

The prediction algorithm based on the extracted subspace is described as follows:

PBCS Algorithm:
Input: The dataset $\mathbf{X}=\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, $\mathbf{x}_i=(c_1, \ldots, c_M, a_1, \ldots, a_N)$;
The context variables $\mathbf{q}=(c_1, c_2, \ldots, c_M)$;
Output: the violent behavior that will occur under the context values $\mathbf{q}=(c_1, c_2, \ldots, c_M)$.
Step:

1) list[1...N]←**X**;
2) for(i=1,i<=N;i++){//computing ρ(C_j, a)
3) for(j=1;j<=M;j++){
4) if((list[i][j]>0&&a>0)||(list[i][j]<0 && a<0))
5) ρ[j]++;
6) }
7) }
8) for(j=1;j<=M;j++){
9) ρ[j]=ρ[j]/N;
10) d[j]=1/ρ[j];
11) }
12) for(i=1;i<=M^2;i++)
13) a[i/M][i%M]=exp(-(d(i/M)-d(i%M))^2/(2σ^2));
14) Calculate the matrix **L**;
15) Divide the graph with the spectral of the matrix **L**;
16) Extract the context subspace of the behavior;
17) Calculate the prediction result with the formula (16);
18) Calculate the occurrence probability of all the behavior and get the prediction result which has the maximum probability.

As mentioned above, the computational complexity of PBCS algorithm contains two partitions. The computational complexity of SCBAA processing is $O(N)$. The computational complexity of the prediction processing is linear with the data size. Thus, the total computational complexity of the PBCS algorithm is $O(N)$. Predicting the terrorist behavior in the extracted subspace can avoid the impact caused by the interference attributes. Then, it does not only improve the prediction result, but also reduce the prediction fluctuation.

5 Experiment and Analysis

5.1 Analysis Based on the Artificial Context Dataset

High dimensionality and fewer instances are the characteristics of the context dataset. In order to meet the requirements, it constructs the artificial dataset as Table 1 with the following rules: a) the behavior of BOMB has stronger correlation with C_4 and C_6, b) the behavior of BOMB has smaller correlation with other context variables, c) the instances in the artificial dataset should be not much more, d) The frequency of different attributes should be normal distribution.

In Table 1, each line represents a 7-dimensional record, where $(C_1,\ldots,C_6)$ represents the context vector, $\{C_4,C_6\} = S_i$, BOMB represents A_i. In the below table, $dom(C_j)=\{0, 1\}$, $dom(A_i)=\{0,1\}$, where '1' represents the occurrence of the attributes and '0' represents the un-occurrence of the attributes.

Using the SCBAA algorithm to extract subspace from artificial dataset, the results are shown in Fig. 1 (the weight in figure shows the similarity between vertexes, the context variables in the solid-line circle are the extracted subspace, the vertexes in the dotted-line circle are the context variables that are in $\overline{S}$).

Using different methods to extract the subspace, the results are shown in Fig. 2. And the *Ncut* values of the sub-graphs are shown in Fig. 3.

Comparing Fig. 1 and Fig. 2, the SCBAA algorithm can extract the context subspace more effectively. Although kernel-based algorithm can extract C_4 and C_6, it also extracts the C_5. Secondly, Fig. 3 shows that the *Ncut* value of the SCBAA algorithm is lower than other algorithms. Thus, the SCBAA algorithm can obtain the graph's optimal division. For the special characteristic of the context dataset, SCBAA concerns the distribution of the samples, the local similarity of data, and the information of the sample's label. Thus, it is superior to other algorithms.

The prediction results of the BOMB behavior based on the context subspace are shown in Fig. 4 and Fig. 5. As shown in Fig.4 and Fig. 5, predicting terrorist behavior in the context subspace is better than in the original whole context space. Although the CONVEX algorithm is worse than Bayesian and SVM algorithms in the original whole context space, it is an effective prediction algorithm in the context subspace. Therefore, the quality of the attribute extraction plays an important role in the prediction process.

Table 1. The artificial dataset

ID	C_1	C_2	C_3	C_4*	C_5	C_6*	BOMB
1	1	0	0	1	1	1	1
2	0	1	1	0	1	0	0
3	0	1	1	0	0	0	0
4	1	0	1	1	1	1	1
5	0	1	1	1	1	0	0
6	0	1	0	0	1	1	0
7	0	1	1	0	0	0	0
8	0	1	1	1	1	1	1
9	0	1	1	1	0	1	1
10	0	1	0	0	1	1	0
11	1	0	1	1	0	1	0
12	1	0	1	0	1	1	0
13	1	1	1	1	0	0	0
14	0	1	0	0	1	0	0
15	1	1	1	0	0	0	0
16	1	1	1	0	1	0	0

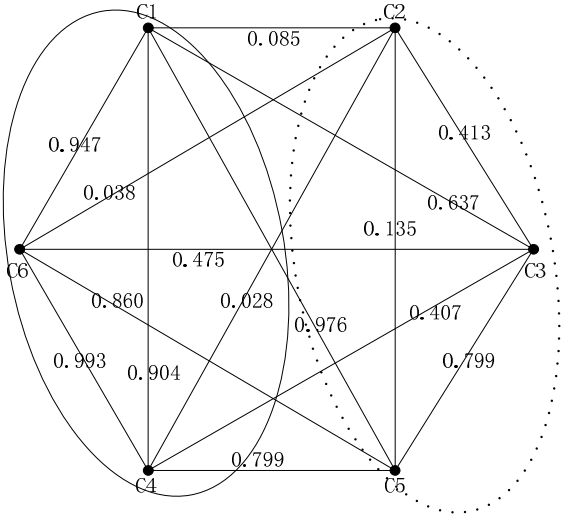


Fig. 1. The result of the SCBAA algorithm

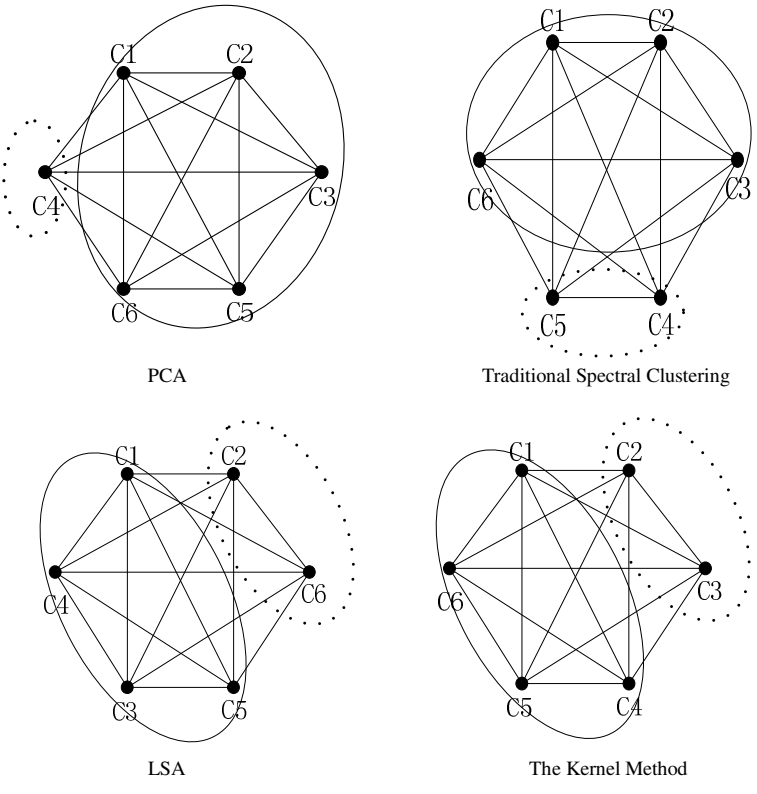


Fig. 2. The results of extracted subspace with the different algorithms

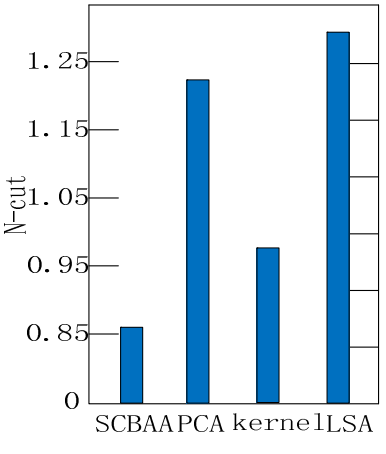


Fig. 3. The Ncut values of the division result

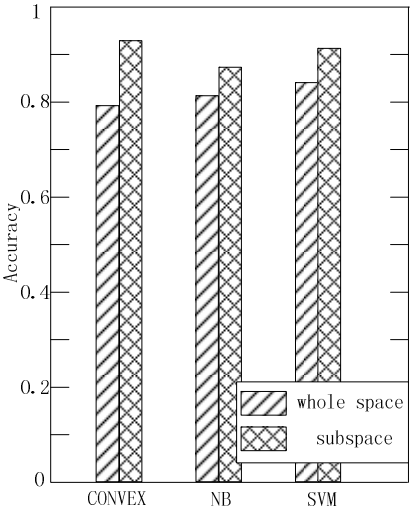


Fig. 4. The accuracy of the prediction

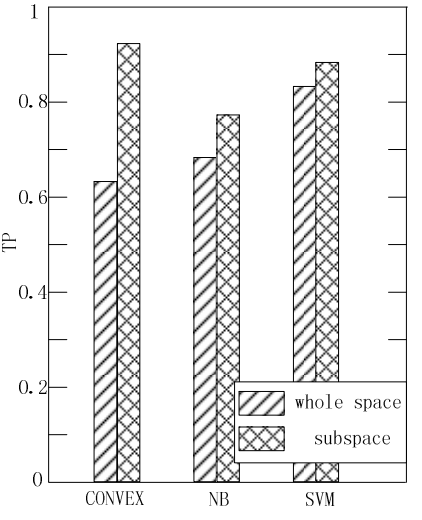


Fig. 5. The TP rate of the prediction

5.2 Analysis Based on the MAROB Dataset

This paper takes MAROB data as the practical context dataset of the terrorist group. The MAROB dataset was published in the 2008. And it can be got from http://www.cidcm.umd.edu/mar/data.asp#marob. The MAROB dataset is the core of the project MAR which records the related information of 118 terrorist organizations in the Middle East from 1980 to 2004.

According to the characteristic of the experimental data, this paper chooses four measures to evaluate the performance of the PBCS algorithm.

Accuracy is commonly used in evaluating the quality of prediction. However, there are some limitations of the accuracy measure in the context dataset. For example, accuracy cannot reflect an algorithm's performance well when data distribution is imbalanced. Thus, in addition to the accuracy, this paper considers other two evaluation measures, TP(True positive rate) and AUC(Area Under roc Curve).

ROC(Receiver Operating Characteristic) curve is a useful visual tool for comparing two prediction models. AUC is the area under ROC curves. It is a statistically consistent and more discriminating measure than accuracy for imbalanced dataset. The algorithm performs better if its AUC is greater than the others.

For the imbalanced dataset, the TP can evaluate a classifier's ability to classify the positive examples correctly. Thus, TP can evaluate the ability to predict the terrorist behavior. This paper also takes the performance time as an evaluation measure.

For the results, the MAX corresponds to the maximum values of the statistics, and the AVG corresponds to the average value of the statistics.

Firstly, the result of the prediction by deleting 20 context attributes randomly in the context vector is shown as Table 2.

If the deleted attributes are not in the extracted subspace, the prediction result is shown as Table 3.

For the situation that the deleted attributes are in the extracted subspace, the prediction result is shown as Table 4.

Table 3 shows that, if the deleted attributes are not in the extracted subspace, PBCS algorithm will reduce the data dimensionality and avoid the distance function failure in the high-dimensional space. Thus, the PBCS algorithm is better than others. As shown in Table 4, when the deleted attributes are in the extracted subspace, it will influence the CONVEX algorithm. However, the PBCS algorithm iteratively calculates the result by updating the fore-prediction with the new information. In this way, it avoids the interference caused by the missing variables. So, the prediction result is better than the traditional context-based similarity algorithm. As shown in Table 2, Table 3, and Table 4, the PBCS algorithm does not only promote the prediction accuracy but also reduce the prediction fluctuation.

Table 2. The result by deleting the context variables randomly

Measure	Statistic	Algorithm			
		PBCS	CONVEX	NB	SVM
Accuracy	MAX	0.941	0.940	0.969	0.929
	AVG	0.910	0.697	0.723	0.710
TP	MAX	0.812	0.962	0.826	0.844
	AVG	0.773	0.721	0.595	0.756
AUC	MAX	0.885	0.885	0.861	0.923
	AVG	0.872	0.702	0.775	0.801
Time(s)	AVG	2.20	1.80	1.27	3.02

Table 3. The deleted variables are not in the extracted subspace

Measure	Algorithm		
	PBCS	NB	SVM
Accuracy	0.928	0.740	0.803
TP	0.838	0.623	0.823
AUC	0.801	0.761	0.803
Time(s)	2.20	1.13	4.99

Table 4. The deleted variables are in the extracted subspace

Measure	Algorithm		
	PBCS	CONVEX	SVM
Accuracy	0.817	0.62	0.775
TP	0.778	0.433	0.728
AUC	0.767	0.627	0.762
Time(s)	3.10	2.30	5.40

6 Conclusion

This paper proposes PBCS prediction algorithm based on the context subspace. The PBCS algorithm firstly uses the improved spectral clustering algorithm to extract the context subspace according to the correlation between the context variables and the behavior variables. Secondly, it predicts the terrorist behavior based on the extracted context subspace. Experiment shows that with the given context knowledge the PBCS algorithm can forecast the most probable terrorist behavior in a few seconds. Although it will cost more 30% of the time consumption in the step of extracting subspace, it avoids the prediction fluctuation. And the performance of the PBCS algorithm is better than the SVM both in the time consumption and the forecast accuracy. It suggests that the extracted subspace plays an important role in the process of the terrorist behavior prediction.

Acknowledgments. This work is sponsored by National Natural Science Foundation of China under Grant No.60773049, supported by Doctoral Fund of Ministry of Education of China No.20093227110005, supported by senior personnel launched Fund of Jiangsu University No.09JDG041, and Innovative fund project of science and technology enterprises in Jiangsu Province No.BC2010172, and SME(Medium and small enterprises) Technology Innovation Fund of China No.09C26213203689.

References

1. Wang, F.Y.: Is Culture Computable. IEEE Intelligent Systems 24(2), 2–3 (2009)
2. Wang, F.Y., Carley, K.M., Zeng, D., Mao, W.: Social Computing: from Social Informatics to Social Intelligence. IEEE Intelligent Systems 22(2), 79–83 (2007)
3. Zeng, D., Wang, F.Y., Carley, K.M.: Social Computing. IEEE Intelligent Systems 22(5), 20–22 (2007)
4. Subrahmanian, V.S.: Culture Modeling in Real Time. Science 317(5844), 1509–1510 (2007)
5. Subrahmanian, V.S., Massimiliano, A., Vanina, M.M., et al.: CARA: A Cultural Reasoning Architecture. IEEE Intelligent Systems 22(2), 12–15 (2007)
6. Samir, K., Martinez, V., Nau, D., et al.: Finding Most Probable Worlds of Probabilistic Logic Programs. In: 1st International Conference on Scalable Uncertainty Management, pp. 45–57. IEEE Press, Washington DC (2007)
7. Xiaochen, L., Wenji, M., Daniel, Z., et al.: Performance Evaluation of Machine Learning Methods in Cultural Modeling. Journal of Computer Science and Technology 24, 1010–1017 (2009)
8. Vanina, M., Simari, G.I., Sliva, A., et al.: CONVEX: Similarity-based Algorithms for Forecasting Group Behavior. IEEE Intelligent Systems 23(4), 51–57 (2008)
9. Daoqiang, Z., Songcan, C.: High-dimensional Data Reduction Methods. Communications of the CCF 5(8), 15–22 (2009) (in Chinese)
10. Jolliffe, I.T.: Principal Component Analysis. Springer, New York (2002)
11. Sam, R.T., Lawrence, S.: Nonlinear Dimensionality Reduction by Local Linear Embedding. Science 290(5500), 2323–2326 (2000)
12. Fan, C.: Spectral Graph Theory. American Mathematical Society, Providence (1997)

13. Xiaoyan, C., Guanzhong, D., Libin, Y.: Survey on Spectral Clustering Algorithm. Computer Science 35(7), 14–18 (2008) (in Chinese)
14. Minorities at Risk Project, http://www.cidcm.umd.edu/mar
15. Bach, F.R., Jordan, M.I.: Learning Spectral Clustering with Application to Speech Separation. Journal of Machine Learning Research 7, 1963–2001 (2006)
16. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, New York (2006)
17. Guyon, I., Elisseeff, A.: An Introduction to Variable and Feature Selection. Journal of Machine Learning Research 3, 1157–1182 (2003)
18. Qiu, X., Wu, L.: Stepwise Nearest Neighbor Discriminate Analysis. In: IJCAI, pp. 829–834. Morgan Kaufmann Press, San Francisco (2005)
19. Haw-Ren, F., Sophia, S., Yousef, S.: Multilevel Manifold Learning with Application to Spectral Clustering. In: CIKM 2010, pp. 419–428. ACM Press, New York (2010)
20. Donghui, Y., Ling, H., Jordan, M.: Fast Approximate Spectral Clustering. In: 15th ACM Conference on Knowledge Discovery and Data Mining, pp. 907–915. ACM Press, New York (2010)

Topic Discovery and Topic-Driven Clustering for Audit Method Datasets

Ying Zhao[1], Wanyu Fu[1], and Shaobin Huang[2]

[1] Department of Computer Science and Technology
Tsinghua University
Beijing, China 100084
[2] College of Computer Science and Technology
Harbin Engineering University
Harbin, China 150001

Abstract. As the promotion of China's Golden Auditing Project and the fast growth of on-line auditing, there are thousands of new computer audit methods emerged every year to fulfill various needs of audit practices. How to organize these existing computer audit methods and use them intelligently have become a fundamental and challenging problem. In this paper, we propose to use topic-driven clustering methods to organize computer audit methods according to the system of computer audit methods that is issued by the National Audit Office of China. We also apply Latent Dirichlet allocation (LDA) analysis to audit method datasets at different levels of granularity. Our experimental results on social insurance computer audit methods show that the topic-driven clustering scheme with topics created by domain experts is the overall best scheme. It achieved an average purity of 0.862 across the datasets. Topics discovered by LDA were consistent with classes defined in the taxonomy for four out of five datasets, and they were effective when used in the topic-driven clustering scheme.

Keywords: topic-driven clustering, audit methods, topic discovery.

1 Introduction

As the promotion of China's Golden Auditing Project and the fast growth of on-line auditing, there are thousands of new computer audit methods emerged every year to fulfill various needs of audit practices. A computer audit method is a combination of ways, steps, and techniques taking in the whole process of audit practice for a particular audit task. Audit methods are knowledge and experiences that audit staff learned from past audit practices. One of the fundamental problems is how to manage those existing audit methods, so that audit staff can share this knowledge. Towards this goal, the National Audit Office of China made several efforts in the past years, for example, standards of writing computer audit methods were issued, so that each audit method is written in the same format. The National Audit Office of China also promoted a taxonomy

J. Tang et al. (Eds.): ADMA 2011, Part II, LNAI 7121, pp. 346–358, 2011.

of computer audit methods based on audit objectives. However, sample audit methods for each class in the taxonomy are either missing or very limited, it is still very challenging to organize the existing computer audit methods according to the taxonomy. Hence, this problem is among the problems that need to be solved urgently in the intelligent audit project supported by Scientific and Technical Supporting Programs funded by Ministry of Science & Technology of China.

In this paper, we aim to tackle the problem of organizing audit methods according to the existing taxonomy. As the labeled data is either missing or very limited, traditional classification algorithms cannot solve this problem. If we adopt clustering approaches, the resultant clustering may not match the taxonomy very well. In [1], we have defined the problem of *topic-driven clustering*, in which case we want to organize a document collection according to a given set of topics. *Topic-driven clustering* is suitable in this case, in which domain experts can give descriptions on each class of the taxonomy. Those descriptions can serve as the topics in *Topic-driven clustering*.

In addition, we also apply topic discovery techniques, such as Latent Dirichlet allocation (LDA) analysis to audit method datasets at different levels of granularity. We would like to discover topics from the entire dataset, and see whether we can use those discovered topics in topic-driven clustering. We would also like to apply topic discovery analysis on audit methods within a single class to search for common audit factors that are embedded in audit methods, which in turn can lead to a better understanding for audit staff.

Our experimental results on social insurance audit methods show that the topic-driven clustering scheme with topics created by domain experts is the overall best scheme. It achieved an average purity of 0.862 across the datasets. Topics discovered by LDA were consistent with classes defined in the taxonomy for four out of five datasets, and they were effective when used in the topic-driven clustering scheme.

The contributions of our work are the following:

1. A novel application of topic discovery and topic-driven clustering to the field of audit.
2. A framework to combine topic-driven clustering and LDA analysis.
3. A comprehensive experimental evaluation on real datasets of social insurance computer audit methods.

The rest of this paper is organized as follows. Section 2 dicusses some of the related work. Section 3 describes background knowledge on computer audit methods, the system of computer audit methods, and how audit methods are represented. Section 4 describes the basic concepts of topic-driven clustering and how it is used in this paper. Section 5 describes the goals of topic discovery and a brief review of Latent Dirichlet allocation (LDA). Section 6 provides the detailed experimental evaluation of the various proposed methods. Finally, Section 7 provides some concluding remarks.

2 Related Work

In terms of how the prior knowledge being used, previous semi-supervised approaches fall into three categories: instance-based, metric-based and the combined approaches. Instance-based approaches explicitly modify the objective function or make certain constraints using must- and cannot-links during the clustering process[2,3,4]. Whereas, metric-based approaches parameterize distance metric and learn the metric parameters in a manner, so that the distance between objects connected by must-links is smaller and the distance between objects connected by cannot-links is larger in general [4,5,6]. Finally the combined approaches integrate both of these techniques in the clustering process [7].

The most popular topic modeling techniques are Latent Dirichlet allocation (LDA) [8] and Probabilistic latent semantic indexing (PLSI) [9]. PLSI is a statistical technique for the analysis of two-mode and co-occurrence data. LDA are widely applied to real-world problems, e.g., information integration [10], topic-author relationship analysis [11], subject extraction from digital library books [12], and classifying short text in large scale [13]. Recently, LDA and PLSI were compared as dimensionality reduction techniques in document clustering [14], and they concluded that LDA and PLSI can reduce the dimension of document feature vectors without degrading the quality of document clusters. Our approach is different from theirs, as we do not use LDA for dimension reduction. Instead, we use the topics discovered by LDA as the topic vectors in topic-driven clustering.

3 Background

3.1 Audit Method

An audit method is a combination of ways, steps, and techniques taking in the whole process of audit practice for a particular audit task. For example, the audit method "Unreasonable Expenditures from Pension Insurance Fund" targets the expenditures from Pension Insurance Fund and searches for unreasonable expenditure records.

Each method determines the nature, timing, and extent of the audit procedures, and tries to find suspected non compliances. Thanks to the on-line auditing system, most audit methods also specify the needed data items from database and the actual SQL codes that fulfill the task. Based on the standards of writing computer audit methods promoted by the National Audit Office of China, a computer audit method contains the following components:

1. **ID**: the identification number of the audit method
2. **Name**: the name of the audit method
3. **Functionality**: the audit objective that the audit method tries to reach
4. **Data**: the data that the audit method needs to operate on

5. **Steps**: the detailed steps of the audit method
6. **Flow Chart**: the detailed steps in flow chart of the audit method
7. **SQL Codes**: the actual SQL codes of the audit method
8. **Further Advices**: further advices suggested by the audit method
9. **Laws and Regulations**: related laws and regulations that the audit method based upon
10. **Instructions for Use**: instructions for using this audit method
11. **Authors**: authors who created this audit method
12. **Date**: date when the audit method was created

3.2 System of Audit Methods

Audit methods are knowledge and experiences that audit staff learned from past audit practices. One of the fundamental problems is how to manage those existing audit methods, so that audit staff can share this knowledge. Towards this goal, the National Audit Office of China initiates efforts to create a system of computer audit methods, which is a taxonomy of audit methods based on mainly on audit objectives. However, classifying existing audit methods based on the taxonomy remains challenging for two reasons: classifying each audit method by domain experts is costly, and the labeled data (sample audit methods for each class in the taxonomy) is either missing or very limited.

In this paper, we evaluate various methods based on a dataset that is one part of the system of computer audit methods, namely, "the system of computer audit methods for social insurance auditing", where each class contains several computer audit methods assigned by domain experts. This dataset is a outcome of a two-year project funded by the National Audit Office of China.

On the top level, the audit methods are classified based on types of social insurance into five classes: pension insurance, unemployment insurance, maternity insurance, basic medical insurance, and work injury insurance. Based on this level, we created five datasets, namely, Pension, Unemployment, Maternity, Medical, and Injury. According to the taxonomy, audit methods are further partitioned into management, collection and fees, payment, and financial for Pension, Unemployment, Medical, and Injury, and collection and fees, payment, and financial for Maternity. We summarize the number of audit methods, the number of classes, and class labels for each dataset in Table 1.

Table 1. Summary of datasets used in evaluation

Dataset	# of Docs	# of Classes	Class Label
Pension	78	4	management, collection and fees, payment, financial
Unemployment	57	4	management, collection and fees, payment, financial
Maternity	42	3	collection and fees, payment, financial
Medical	51	4	management, collection and fees, payment, financial
Injury	56	4	management, collection and fees, payment, financial

3.3 Preparation of Audit Methods

We extracted text information from all components for each audit method. Currently we did not include logical information embedded in the flow chart component and SQL codes, we leave those to future work. Since all audit methods are written in Chinese, we apply Chinese words segmentation analysis with user-defined dictionary to take audit vocabulary into consideration. After that we have each audit method as a document that contains a bag of Chinese words.

3.4 Document Representation

We use the symbols n, m, and k to denote the number of documents, the number of terms, and the number of clusters, respectively. We use the symbol S to denote the set of n documents that we want to cluster, $S_1, S_2, \ldots, S_k$ to denote each one of the k clusters, $n_1, n_2, \ldots, n_k$ to denote the sizes of the corresponding clusters, and $T_1, T_2, \ldots, T_k$ to denote the topic prototype vectors given as prior knowledge.

The various criterion functions use the vector-space model [15] to represent each document. In this model, each document d is considered to be a vector in the term-space. In particular, we employed the $tf - idf$ term weighting model, in which each document can be represented as

$$(tf_1 \log(n/df_1), tf_2 \log(n/df_2), \ldots, tf_m \log(n/df_m)). \tag{1}$$

where tf_i is the frequency of the ith term in the document and df_i is the number of documents that contain the ith term. To account for documents of different lengths, the length of each document vector is normalized so that it is of unit length ($\|d_{tfidf}\| = 1$), that is each document is a vector on the unit hypersphere.

Given a set A of documents and their corresponding vector representations, we define the ***composite*** vector D_A to be $D_A = \sum_{d \in A} d$, and the ***centroid*** vector C_A to be $C_A = D_A/|A|$.

In the vector-space model, the cosine similarity is the most commonly used method to compute the similarity between two documents d_i and d_j, which is defined to be

$$\cos(d_i, d_j) = {d_i}^t d_j / (\|d_i\| \|d_j\|). \tag{2}$$

The cosine formula can be simplified to

$$\cos(d_i, d_j) = {d_i}^t d_j, \tag{3}$$

when the document vectors are of unit length. This measure becomes one if the documents are identical, and zero if there is nothing in common between them (*i.e.*, the vectors are orthogonal to each other).

4 Topic-Driven Clustering

4.1 Goals

The goal of *topic-driven clustering* [1] is to organize a document collection according to a given set of topics. It is a means of incorporating prior knowledge

into the clustering process and is best suitable for a situation where the set of topics of the resultant clustering is known, but the labeled data is either missing or very limited.

In this paper, we aim to organize existing computer audit methods according to the system of computer audit methods promoted by the National Audit Office of China. As sample audit methods for each class in the system are either missing or very limited, but domain experts can create descriptions for each class, topic-driven clustering is best suitable for this problem.

4.2 Method

We briefly review the two components of topic-driven clustering: a semi-supervised criterion function that describes the perfect clustering result would be; and an optimization process to achieve a clustering result guided by the criterion function.

Semi-supervised Criterion Function. A good clustering solution must have two properties: the documents from the same topic should also belong to a same cluster; documents from a same cluster should be more similar than those from different clusters. The first property is captured by a supervised criterion function, whereas the second property is captured by a unsupervised criterion function. Finally, these two components are combined together to become a semi-supervised criterion function.

We assume that the description of each topic is available as prior knowledge and can be represented as a vector. Given these topic prototype vectors, the similarity between each document and its topic can be defined as the cosine similarity between the vector of the document d and the prototype vector of the topic T_r. The **internal supervised criterion function**, denoted by $\mathcal{S}_\mathcal{I}$, tries to maximize the similarity between the documents in a cluster to the topic associated with the cluster. The formal definition can be written as

$$\mathcal{S}_\mathcal{I} = \sum_{r=1}^{k} \sum_{d_i \in S_r} \cos(d_i, T_r) = \sum_{r=1}^{k} {D_r}^t T_r. \tag{4}$$

The **internal unsupervised criterion function**, denoted by $\mathcal{U}_\mathcal{I}$ (5) is used by the popular vector-space variant of the K-means algorithm [16,17,18,19]. In this algorithm each cluster is represented by its centroid vector and the goal is to find the solution that maximizes the similarity between each document and the centroid of the cluster that is assigned to.

$$\mathcal{U}_\mathcal{I} = \sum_{r=1}^{k} \sum_{d_i \in S_r} \cos(d_i, C_r) = \sum_{r=1}^{k} \|D_r\|. \tag{5}$$

Finally, we use a *weighted* scheme to combine those two components, which allows fine-tuning of the tradeoffs among the objectives with relatively less complexity. Our **semi-supervised criterion function** can be defined as

$$\mathcal{M}_{\mathcal{I}} = M(\mathcal{U}_{\mathcal{I}}, \mathcal{S}_{\mathcal{I}}) = \alpha \sum_{r=1}^{k} \|D_r\| + (1-\alpha) \sum_{r=1}^{k} {D_r}^t T_r \tag{6}$$

which tries to maximize the similarity between the documents in a cluster to the cluster centroid and the topic associated with the cluster simultaneously. It allows a fine-tuned control of the tradeoffs among the objectives by varying the preference factor α.

Optimization. Our greedy optimizer for the semi-supervised criterion function consists of two phases: (i) ***initial clustering***, and (ii) ***cluster refinement***. In the initial clustering phase, a clustering solution is computed as follows. If k is the number of desired clusters, k topic vectors are set as the *seeds* of these clusters. The similarity of each document to each of these k seeds is computed, and each document is assigned to the cluster corresponding to its most similar seed. The similarity between documents and seeds is determined using the cosine measure of the corresponding document vectors.

The refinement strategy is the following. It consists of a number of iterations. During each iteration, the documents except for cluster seeds are visited in a random order. For each document, d_i, we compute the change in the value of the criterion function obtained by moving d_i to one of the other $k-1$ clusters. If there exist some moves that lead to an improvement in the overall value of the criterion function, then d_i is moved to the cluster that leads to the highest improvement. If no such cluster exists, d_i remains in the cluster that it already belongs to. The refinement phase ends, as soon as we perform an iteration in which no documents moved between clusters.

The optimization procedure for the supervised criterion function is the same as above. The optimization procedure for the unsupervised criterion function differs a little bit, in which cluster seeds are selected randomly from the dataset.

4.3 Topics Induced by LDA and Domain Experts

Now the final problem is how to get those topic vectors as prior knowledge. In this paper, we consider two possible ways of creating topic vectors used in topic-driven clustering.

The best possible topic vectors can be created by domain experts. In this way, domain experts can describe a set of keywords that best describe a class (or a topic). The drawback of this approach again is that creating topic vectors manually could be both time-consuming and costly.

Alternatively, we could apply popular topic modeling tools to the dataset,and see whether the discovered topics are consistent with the system of audit methods. If there is a good match between discovered topics and classes, we can use those as topic vectors in topic-driven clustering. The advantage is that it is actually a unsupervised process. The drawback is that when the discovered topics do not match the classes, we cannot use those topics.

5 Topic Discovery

5.1 Goals

The goals of topic discovery are two fold. First, we would like to discover topics from the entire dataset, and see whether the the discovered topics are consistent with the classes in the taxonomy. If they match well, the discovered topics can be used in topic-driven clustering.

Second, for audit methods within a single class, we would also like to apply topic discovery techniques. The goal of applying topic discovery techniques in this finer granularity is to search for common audit factors that are embedded in audit methods. These common audit factors are too detailed to be included as a class in the system of audit methods, but they are useful in helping audit staff better understanding those existing audit methods.

5.2 Method

Latent Dirichlet allocation (LDA)[8] is a generative model for topic modeling, which explains why parts of observations are similar by unobserved groups. In LDA, each document may be viewed as a mixture of various topics, and the topic distribution is assumed to have a Dirichlet prior. Furthermore, each topic has probabilities of generating various words.

The probability of exhibiting a word sequence in a document can be calculated with three components: selecting a multinomial over the set of topics from the Dirichlet distribution; selecting a topic for each array element of the document according to this multinomial; and selecting a word for the array element according to multinomial over the set of words that corresponds to the selected topic. Then the probability of the word sequence of the whole document set can be calculated. By maximizing the log of this probability, model parameters can be estimated, and we can get word distribution for each topic accordingly.

6 Experiments

6.1 Metric

We use the *purity* metric to evaluate the quality of resulting clusters, which measures the extend to which each cluster contained documents from primarily one class.

Given a particular cluster S_r of size n_r, the purity of this cluster is defined to be

$$P(S_r) = \frac{1}{n_r} \max_i (n_r^i),$$

where n_r^i is the number of documents of the ith class that were assigned to the rth cluster. So it is nothing more than the fraction of the overall cluster size that the largest class of documents assigned to that cluster represents. The overall

purity of the clustering solution is obtained as a weighted sum of the individual cluster purities and is given by

$$Purity = \sum_{r=1}^{k} \frac{n_r}{n} P(S_r).$$

In general, the larger the value of purity, the better the clustering solution is.

6.2 Topics Discovered by LDA

We first present our experimental results of using LDA to identify topics in various datasets. As stated in section 5, we have two goals for the topic discovery task: to verify whether the topics discovered are consistent with the classes of the taxonomy; and to search for common audit factors from the existing audit methods. Towards these two goals, we designed two sets of experiments accordingly. For LDA analysis, we used GibbsLDA++ [13], which employs Gibbs sampling in parameter estimation and inference. We used the default settings for all parameters, and outputted 10 keywords for each topic.

The first set of experiments tried to verify whether the topics discovered are consistent with the classes of the taxonomy. Towards this goal, we applied LDA analysis on each dataset to generate N topics, where N is the number of classes in the dataset according to the taxonomy. In particular, N equals 4 for Pension, Medical, Unemployment, and Injury, and 3 for Maternity. We then asked domain experts to verify whether the topics discovered by LDA are consistent with the taxonomy.

Domain experts verified that for four out of five datasets, namely, Pension, Unemployment, Injury, and Maternity, the topics discovered by LDA were consistent with the taxonomy. For Medical, they did not match very well. We showed the topics discovered by LDA for Maternity and Medical in Table 2 and Table 3, respectively. Note that the keywords shown in tables here are the English translations of the original Chinese keywords for ease of reading.

The keywords of each topic discovered by LDA for Maternity are shown in Table 2. There are three classes in dataset Maternity according to the taxonomy, namely, collection and fees, payment, and financial. There are good correspondences between the three topics and three classes. More specifically, Topic0 corresponds to payment, Topic1 corresponds to collection and fees, and Topic2 corresponds to financial. The entries that are bold-faced correspond to the keywords that are typical for each class. For example,"pay", "allowance" etc. are common keywords in the audit methods from the class payment, as it contains the set of audit methods that audits how maternity insurance is paid to individuals. As another example, "fees","company","collection" are common keywords in the audit methods from the class collection and fees, as it contains the set of audit methods that audits how maternity insurance premiums are collected from companies and individuals.

We only show the results for the Maternity dataset here, the trend is similar for Pension, Unemployment, and Injury. For each discovered topic, domain experts were able to identify a corresponding class from the taxonomy, and vice versa.

Table 2. Topics discovered by LDA on maternity insurance audit methods (the keywords shown here are the English translations of the original Chinese keywords)

Topic0	Topic1	Topic2
maternity	**fees**	method
method	**company**	audit
insurance	**amount**	**income**
audit	personal	insurance
pay	payable	maternity
notice	**register**	**expenditure**
allowance	unit	**Amount in**
employee	**collection**	annual
objective	insurance	**expenditure account**
society	staff	data

Table 3. Topics discovered by LDA on medical insurance audit methods (the keywords shown here are the English translations of the original Chinese keywords)

Topic0	Topic1	Topic2	Topic3
method	method	**Specified**	**company**
prescription	insurance	service	**fees**
item	subject	organization	method
hospital	audit	medical	medical
name	society	**drug**	**insured**
drug	number	**pharmacy**	basic
amount	medical	method	audit
data	basic	type	insurance
audit	data	sale	type
insurance	organization	information	fund

The keywords of each topic discovered by LDA for Medical are shown in Table 3. There are four classes in dataset Medical according to the taxonomy, namely, management, collection and fees, payment, and financial. Different from Maternity, the topics discovered by LDA do not match the classes very well. Domain experts verified that Topic3 corresponds to collection and fees. The entries in Topic3 that are bold-faced correspond to the keywords that are also typical for collection and fees. We also bold-faced entries in Topic0 and Topic2. Actually, both Topic0 and Topic2 can find a meaningful set of audit methods that support the topic. More specifically, Topic0 describes basic medical insurance claims involving prescription, drugs, and hospitals; Topic2 describes basic medical insurance claims involving specified organization, pharmacy, and drugs. Since the topics do not match the classes, we cannot use these topics in topic-driven clustering for Medical.

The second set of experiments tried to search for common audit factors from existing audit methods. Towards this goal, we applied LDA on the set of audit

methods within each class for every dataset. We asked LDA to generate 5 topics, and then asked domain experts to verify whether these topics are meaningful and contain common audit factors.

Table 4. Topics discovered by LDA on pension insurance management audit methods (the keywords shown here are the English translations of the original Chinese keywords)

Topic0	Topic1	Topic2	Topic3	Topic4
accout	method	**age**	fees	retire
personal	staff	quota	amount	date
company	basic	data-set	personal	modify
accumulative	personal	economy	staff	treatment
insurance	audit	employed	insured	**population**
payment	pension	category	computation	category
principal	insurance	staff	pension	information
interest	insured	number	supplement	personal
basic	data	industry	arrear	time
appropriate	**multi-dimension**	administration	transfer	birth

For most cases, domain experts found out that the discovered topics were meaningful and can be used to help audit staff navigating, searching, and preparing audit method in real audit practice. As an example, we show the topic discovered by LDA on pension insurance management audit methods in Table 4. We show the common audit factor identified by domain experts for each topic in bold-face. Except for Topic3, domain experts found out the common audit factor for each topic. Based on this result, we now know that the audit methods in the management class of the Pension dataset focus on several audit factors, such as account, age, and population etc.

6.3 Comparison of Various Clustering Methods

We evaluated our topic-driven clustering algorithms by comparing the purity values of the clustering results obtained by optimizing $\mathcal{U}_{\mathcal{I}}$, $\mathcal{S}_{\mathcal{I}}$, and $\mathcal{M}_{\mathcal{I}}$. The first two values served as our baselines. Across the datasets except for Medical, the topics discovered by LDA were used in our topic-driven clustering scheme. We also ask domain experts to create topic vectors consisting of ten keywords from scratch. Observed from previous study [1], we know that the parameter α used for combining supervised and unsupervised term in $\mathcal{M}_{\mathcal{I}}$ can be set with an value in [0.1..0.4] to obtain better partitional clusters. In our comparison, we set $\alpha = 0.4$.

The results in purity are shown in Table 5. The entries that are bold-faced correspond to the methods that perform the best for a particular dataset.

From Table 5, we can see several interesting trends. First, the topic-driven clustering scheme (*i.e*, optimizing $\mathcal{M}_{\mathcal{I}}$) achieved the best performance among the three clustering schemes when LDA topics were used. The improvements over

Table 5. FScores of the clustering solutions obtained by the various clustering methods

Datasets	$\mathcal{U}_\mathcal{I}$	LDA Topics		Domain Expert Topics	
		$\mathcal{S}_\mathcal{I}$	$\mathcal{M}_\mathcal{I}$	$\mathcal{S}_\mathcal{I}$	$\mathcal{M}_\mathcal{I}$
Pension	0.654	0.841	0.854	0.805	**0.866**
Unemployment	0.623	0.820	0.836	0.852	**0.885**
Maternity	0.738	0.867	0.889	**0.933**	0.911
Medical	0.569	–	–	0.764	**0.782**
Injury	0.625	0.800	0.817	**0.867**	**0.867**

the unsupervised scheme varied from 20% to 34%, whereas the improvements over the supervised scheme were around 2%. Second, the topic-driven clustering scheme achieved the best performance for four out of five datasets when topics created by domain experts were used. The improvements over the unsupervised scheme varied from 23% to 38%, whereas the improvements over the supervised scheme were from 2% to 8%. Third, the topic-driven clustering scheme with topics created by domain experts is the overall best scheme. It achieved an average purity of 0.862 across the datasets. Forth, LDA topics were also effective in driving the clustering process.

7 Conclusion

In this paper, we aimed to solving the problem of organizing existing computer audit methods according to the system of computer audit methods promoted by the National Audit Office of China. We adopted the topic-driven clustering scheme with topics discovered by LDA and topics created by domain experts. We designed comprehensive experiments to evaluate various methods on datasets created based upon computer audit methods for social insurance auditing. Our experimental results showed that the topic-driven clustering scheme with topics created by domain experts is the overall best scheme. It achieved an average purity of 0.862 across the datasets. Topics discovered by LDA were consistent with classes defined in the taxonomy for four out of five datasets, and they were effective when used in the topic-driven clustering scheme.

Acknowledgment. This work was funded by the National Science Foundation China (Project No. 60703058) and Scientific and Technical Supporting Programs funded by Ministry of Science & Technology of China (Project No. 2009BAH42B0202).

References

1. Zhao, Y., Karypis, G.: Topic-driven Clustering for Document Datasets. In: 2005 SIAM International Conference on Data Mining (SDM 2005), pp. 358–369 (2005)
2. Wagstaff, K., Cardie, C., Rogers, S., Schroedl, S.: Constrained k-means clustering with background knowledge. In: 18th International Conference on Machine Learning (ICML 2001), pp. 577–584 (2001)

3. Davidson, I., Ravi, S.S.: Agglomerative Hierarchical Clustering with Constraints: Theoretical and Empirical Results. In: Jorge, A.M., Torgo, L., Brazdil, P.B., Camacho, R., Gama, J. (eds.) PKDD 2005. LNCS (LNAI), vol. 3721, pp. 59–70. Springer, Heidelberg (2005)
4. Bade, K., Nurnberger, A.: Creating a cluster hierarchy under constraints of a partially known hierarchy. In: 2008 SIAM International Conference on Data Mining (SDM 2008), pp. 13–24 (2008)
5. Basu, S., Bilenko, M., Monney, R.: A probabilistic framework for semi-supervised clustering. In: 10th International Conference on Knowledge Discovery and Data Mining (2004)
6. Xing, E.P., Ng, A.Y., Jordan, M.I., Russell, S.: Distance metric learning with application to clustering with side-information. Advances in Neural Information Processing Systerms 15, 505–512 (2003)
7. Bilenko, M., Basu, S., Monney, R.: Integrating constraints and metric learning in semi-supervised clustering. In: 21th International Conference on Machine Learning, ICML 2004 (2004)
8. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. Journal of Machine Learning Research 3(4-5), 993–1022 (2003)
9. Hofmann, T.: Probabilistic Latent Semantic Indexing. In: The Twenty-Second Annual International SIGIR Conference on Research and Development in Information Retrieval (1999)
10. Barnard, K., Duygulu, P., Freitas, N., Forsyth, D., Blei, D., Jordan, M.: Matching Words and Pictures. Journal of Machine Learning Research 3, 1107–1135 (2003)
11. Griffiths, T., Steyvers, M.: Finding Scientific Topics. Proc. of the National Academy of Sciences 101 (suppl. 1), 5228–5235 (2004)
12. Mimno, D., McCallum, A.: Organizing the OCA: Learning Faceted Subjects from a library of digital books. In: Proc. of JCDL 2007, pp. 376–385 (2007)
13. Phan, X., Nguyen, L., Horiguchi, S.: Learning to Classify Short and Sparse Text & Web with Hidden Topics from Large-scale Data Collections. In: Proc. of The 17th International World Wide Web Conference (WWW 2008), pp. 91–100 (2008)
14. Masada, T., Kiyasu, S., Miyahara, S.: Comparing LDA with pLSI as a Dimensionality Reduction Method in Document Clustering. In: Tokunaga, T., Ortega, A. (eds.) LKR 2008. LNCS (LNAI), vol. 4938, pp. 13–26. Springer, Heidelberg (2008)
15. Salton, G.: Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesley (1989)
16. Cutting, D.R., Pedersen, J.O., Karger, D.R., Tukey, J.W.: Scatter/gather: A cluster-based approach to browsing large document collections. In: 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 318–329 (1992)
17. Larsen, B., Aone, C.: Fast and effective text mining using linear-time document clustering. In: 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 16–22 (1999)
18. Dhillon, I.S., Modha, D.S.: Concept decompositions for large sparse text data using clustering. Machine Learning 42(1/2), 143–175 (2001)
19. Zhao, Y., Karypis, G.: Empirical and theoretical comparisons of selected criterion functions for document clustering. Machine Learning 55(3), 311–331 (2004)

Transportation Modes Identification from Mobile Phone Data Using Probabilistic Models

Dafeng Xu[1], Guojie Song[1,⋆], Peng Gao[2], Rongzeng Cao[2], Xinwei Nie[1], and Kunqing Xie[1]

[1] Key Laboratory of Machine Perception(Peking University), Ministry of Education, China
gjsong@pku.edu.cn
[2] IBM Research, Beijing, China

Abstract. Transportation modes identification is an important transportation research problem with wide applications. Traditional methods are mainly done based on GPS, WiFi and some other electronic devices, which are actually not in adequately widespread use. The popularity of mobile phones makes the work of identification by mobile phone data valuable. In this paper, based on mobile phone data without other equipment for assistance, we design a probabilistic method to identify transportation modes. The method consists of a Hidden Markov Model with two sub-models for different traffic conditions. The Speed Distribution Law (SDL) based approach is used under the normal condition; to improve the performance of our method under the congested condition, the Cumulative Prospect Theory (CPT) based approach is adopted as a supplementary way to do identification. Experiments on real data show that our method can reach high accuracy in the normal and congested condition alike.

1 Introduction

Identifying transportation modes is a fundamental study in transportation issues. There are a host of algorithms that identify transportation modes by electronic systems. Many works use sensitive devices, such as GPS [14] [15], accelerometer [5] and pedometer [10]. The above electronic devices can collect multi-features of people's motion, such as location, instantaneous speed and acceleration. However, Many of devices are too expensive, or not commonly carried by people. Hence these methods cannot be widely used, despite their good performances.

Recently, some works are made to identify transportation modes by mobile phones. When people are in motion, the geographic information will be collected by mobile base stations which are nearby. In addition, with some special devices, mobile phones can also get precise features, such as instantaneous speed and acceleration. Several effective algorithms are designed based on mobile phones combined with GPS [9] or WiFi [7]. However, mobile phones are not popularly equipped with GPS or WiFi in some countries, which makes these algorithms hard to be widely used as well.

⋆ Corresponding author, Supported by National Science Foundation of China Project under Grant 60874082, Beijing municipal natural science foundation (4102026)and IBM 2010 SUR-Project.

J. Tang et al. (Eds.): ADMA 2011, Part II, LNAI 7121, pp. 359–371, 2011.

In this paper, we identify transportation modes from mobile phone data, without any extra sensing devices for assistance. Our aim is to learn three transportation modes: *driving, biking and walking*. Restricted by the precision of sensing hardware, we mainly analyze the user's average speed, instead of the instantaneous speed and acceleration. We also rely on user's historical data and the geographical information. Based on above conditions, we establish a Hidden Markov Model, with two sub-models. A Hidden Markov Model has two state sequences. The hidden states are transportation modes, whose switch and maintenance follow the Markov property. The observed states consist of data collected by mobile phones. Under the normal condition, the observed states are solely made up of the information of the average speed. We use the classical probability theory method to analyze transportation tools' **Speed Distribution Law** (SDL). When the traffic condition is congested, the information of the average speed become rough. To improve the accuracy, we adopted **Cumulative Prospect Theory** (CPT) to make use of the user's historical data as the supplementary information. Then, we conduct extensive experiments on real data. Compared with existing algorithms, our method uses more easily accessible and less sophisticated data, but also performs well.

Our major contributions are outlined as follows:

- First, our models only require the average speed data, some historical data and the geographical information, which are more accessible and less sophisticated.
- Second, we design a Hidden Markov Model to identify transportation modes with two sub-approach to analyze the observed states. Under the normal traffic condition, SDL has been used to analyze the speed distribution law. Under the congested traffic condition, we adopt CPT to analyze historical data.
- Third, we conduct extensive experiments on real datasets. Our identifying method can reach fairly high accuracy in all real datasets we tested.

The rest of this paper is organized as follows: Section 2 will give the preliminary knowledge and the problem statement. In Section 3, we discuss the methodology of identification. We introduce the experimental setup and the result in Section 4 and 5. Section 6 discusses related works. Finally we conclude this paper in Section 7.

2 Preliminary and Problem Statement

The Form of Raw Data. Let us call the user's motion between the origin and destination as a **trip**. A mobile base station obtains the switch record when a user passes by this station in a trip. The switch record contains the time this user passes by the base station and the geographic position of this station. Table 1 shows an illustration.

Table 1. An illustration of a piece of raw data

	Record 1	Record 2	Record 3	Record 4
Time	2011-1-1 19:24	2011-1-1 19:33	2011-1-1 19:41	2011-1-1 19:44
Position	33.01° N, 124.19° E	33.06° N, 124.16° E	33.17° N, 124.15° E	33.19° N, 124.05° E

The period of motions determined by two consecutive switching records is said to be a **segment**. A user's trip is made up of at least one segment.

Data Pre-processing. It is necessary to transform the motions among base stations to the motions in the road network. As base stations are regularly located near the road, for every base station, we simply choose the nearest point in the road network as the place the user passes by this station. In this way, we can compute the moving distance in a segment by consulting the local e-map.

Definition 1. ***Average Speed Sequence:*** *In a trip, given the time a user spends and the distance (in road network) this user covers in each segment, we can get the sequence of the speed* $\{\rho_1, \rho_2, \cdots, \rho_n\}$. *It is a trip's* ***average speed sequence****.*

The Form of Pre-processed Current Data. After the data pre-process, We take the average speed into consideration. The information of the time and positions (i.e., the geographical information of two base stations that decide a segment) remain in the form. Generally, we call the data by which we want to identify the user's transportation modes as **current data**. Table 2 shows an illustration.

Table 2. An illustration of a piece of current data

2011-1-1	Segment 1	Segment 2	Segment 3
Starting / Ending Time	19:24	19:33	19:41
Ending Time	19:33	19:41	19:44
Starting Position	33.01° N, 124.19° E	33.06° N, 124.16° E	33.17° N, 124.15° E
Ending Position	33.06° N, 124.16° E	33.17° N, 124.15° E	33.19° N, 124.05° E
Average Speed	33.1 km/h	24.5 km/h	24.6 km/h

Historical Data. In a user's current data, suppose that the starting time T_0 and positions of a segment are known. Given $\{T_1, T_2, \cdots, T_n\}$ which is the set of time that the user passes by the starting position of this segment previously, we say that this user **has the corresponding historical data** of this segment, if 1) this user finally passed by the ending position of this segment, 2) $\min\limits_{1 \le i \le n} |T_i - T_0| < \eta$, where η is a constant (e.g., 30 minutes). It is not required that the time this user passes by the ending position of this segment are approximate. Table 3 shows an illustration.

Table 3. An illustration of a piece of historical data

Starting Time	19:24 ± 5 mins			
Starting / Ending Positions	33.01° N, 124.19° E		33.06° N, 124.16° E	
Date	2011-1-1	2010-12-24	2010-12-22	2010-12-19
Average Speed	33.1 km/h	24.5 km/h	24.6 km/h	32.5 km/h

Definition 2. ***Mode Sequence:*** *The sequence of a user's transportation modes is a* ***mode sequence*** *of a trip. We use 1, 2, 3 to denote the driving, biking and walking mode.*

For example, $\{1, 3, 3\}$ represents the mode sequence $\{driving, walking, walking\}$.

Remark 1. We only consider the most representative mode in which the user covers most distance. For example, if a user drives 1.9 kilometers and walks 3.3 kilometers in a segment, then we judge the walking mode as the transportation mode in this segment.

Observed Data. Call both historical and current data as **observed data** generally.

Problem Statement. Given historical and current data of a trip, the identification method is to discover the most likely mode sequence.

3 Proposed Approaches

In this section, we first introduce the general identification formula. Subsequently, we introduce the method to determine the observed states of our Hidden Markov Model. Two approaches are recommended to be adopted under different circumstances.

3.1 General Identification Formula

The general identification formula is to evaluate the probability that a possible transportation modes sequence $\{M_1, M_2, \cdots, M_n\}$ occurs. Obviously, this formula consists of two parts, involving the observed and hidden states, respectively. Hidden states are transportation modes; the switch and maintenance of modes is a Markov process, with transition probabilities $p_{i,j}$. For example, $p_{1,2}$ is the probability that a user switch to biking mode from driving mode.

Now we discuss how observed data influence transportation modes identification.

Definition 3. ***Influencing Probability of Observed Data:*** *The* ***influencing probability of observed data****, or briefly stated, the* ***influencing probability****, is the probability that a user is in a transportation mode in a segment.* $q_j(i)$ *is the general form of the influencing probability, where* i *is the number of the segment and* j *is the number of the mode.*

Definition 4. ***General Identification Formula:*** *The* ***general identification formula***

$$P(M_1 = m_1, M_2 = m_2, \cdots, M_n = m_n) = \prod_{i=1}^{n} q_i(m_i) \cdot \prod_{j=1}^{n-1} p_{m_j, m_j+1}. \quad (1)$$

is the expression to evaluate the probability of the occurrence of a mode sequence of a trip. The maximization of this formula is the mathematical version of our problem.

3.2 SDL-Based Approach

From this subsection, we start to analyze how to determine the detailed form of the influencing probability $q_i(j)$. Under the normal traffic condition, the motions of transportation tools follow certain laws. Based on this idea, we adopt the **Speed Distribution Law (SDL) based approach** to determine the observed states.

Definition 5. ***Absolute Probability:*** *Given a transportation mode* i $(i = 1, 2, 3)$*, the* ***absolute probability*** $\gamma_i(v)$ *is a probability distribution function.*

In our work, we pick the normal distribution function as the absolute probability.

The symmetry of the normal distribution function is reasonable for biking and walking mode; however, in urban areas, this property cannot be held for driving mode. Hence, we define the **low-speed bias index** (LBI, or more brief, L), where $L \in [1, +\infty)$. This index is to modify the absolute probability of the driving mode:

$$\gamma_1(v) = \frac{1}{L}\underline{\gamma_1(v)}, v > \mu_1. \tag{2}$$

$$\gamma_1(v) = \frac{2L-1}{L}\underline{\gamma_1(v)}, 0 < v \leq \mu_1. \tag{3}$$

Also, we denote that $\gamma_i(v) = \underline{\gamma_i(v)}$ for $i = 2, 3$. In the latter part of this paper, we use $\gamma_i(v)$ to denote the absolute probability, instead of $\underline{\gamma_i(v)}$.

Definition 6. ***Relative Probability:*** *Given a speed v, the **relative probability** $\psi_i(v)$ is the probability that the user is in the transportation mode i. The expression of the relative probability is:*

$$\psi_i(v) = \frac{\gamma_i(v)}{\gamma_1(v) + \gamma_2(v) + \gamma_3(v)}, i = 1, 2, 3. \tag{4}$$

The relative probability is the influencing probability under the normal traffic condition, i.e., $q_j(i) = \psi_j(\rho_i)$.

Without using external data, the SDL-based approach is even effective if there exists short-term traffic congestion. Given an average speed sequence {33.6 km/h, 11.5 km/h, 24.5 km/h} that belongs to a driver who encounters traffic congestion in this trip. Though the average speed in the second segment is like the biking speed, it is still very likely to determine "driving" as the mode in this segment when considering the high probability of maintaining a transportation mode in real life.

3.3 CPT-Based Approach

Though the SDL-based approach is robust in most traffic conditions, long-term traffic congestion may still cause its futility. Here is an illustration. Given a user's average speed sequence {12.24 km/h, 12.33 km/h, 12.41 km/h, 12.24 km/h}. Four numbers of speed are typically the biking speed; however, if this user encounters traffic congestions in the whole trip, it is also possible that four modes are all "driving".

The problem occurs because the driving speed loses its distribution law, and even be similar to the biking speed in traffic congestion. One solution to overcome this problem is to do a subsidiary analysis on historical data. Learning from the economics theory [11], here we introduce a **Cumulative Prospect Theory (CPT) based approach** to simulate user's behaviors of choosing transportation modes by historical data, and thus determine the form of the influencing probability. This approach is useful because it depicts people's choice under risk, which occurs in traffic congestion. The CPT-based approach is a supplementary method in the congested traffic condition. We call the SDL-based approach with the supplementary CPT-based approach as the **SDL + CPT combined approach**, or simply the **combined approach**.

There are two premises that make the CPT-based approach applicable: first, the traffic condition is congested; second, the user's speed lies in a certain realm where the speed information is ambiguous. We mainly consider the most common situation, in which the biking speed is similar to the driving speed.

Definition 7. ***Ambiguous Interval:*** *An **ambiguous interval** is a closed interval in which the biking and driving speed cannot be judged different. The ambiguous interval is $[\mu_2 - \theta\sigma_2, \mu_2 + \theta\sigma_2]$, where μ_2 and σ_2 are the mean and standard deviation of the absolute probability of biking mode. The **ambiguous parameter** θ is a nonnegative number, which decides the measure of the interval.*

Definition 8. ***Reference Time Cost:*** *Given the set of time costs of historical records in the corresponding segment, the **reference time cost** t_r is the median of this set.*

Suppose when choosing driving mode, a user estimates that there are n_1 events which may occur. For the i-th event, the probability of occurrence is $\lambda_i^{(1)}$ and the time cost is $x_i^{(1)}$, where $x_i^{(1)} > x_j^{(1)}$ iff $i < j$. Suppose there are N_1 events whose time costs are no less than the reference time cost t_r, and the rest of events have less time costs than t_r. Similarly, when choosing biking mode, suppose there are n_2 events which may occur. For the i-th event, the probability of occurrence is $\lambda_i^{(2)}$ and the time cost is $x_i^{(2)}$, where $x_i^{(2)} > x_j^{(2)}$ iff $i < j$. Suppose there are N_2 events whose time costs are no less than the reference time cost t_r, and the rest of events have less time costs than t_r.

Definition 9. ***Capacity Function:*** *Given a probability p and a time cost t, the **capacity function** $w(p, x)$ is a function that measures the weight of the probability, where $x = t_r - t$. This function has different expressions when the sign of $t_r - t$ changes. To simplify our denotation, we use $w^+(p)$ to denote $w(p, x)$ when $x = t_r - t > 0$ and $w^-(p)$ to denote $w(p, x)$ when $x = t_r - t < 0$.*

Definition 10. ***Event Value Function:*** *Given an event's time cost t and the transportation mode i, the **event value function** $v(t)$ is a function that measures the value of this event. The **negative value function** $V^-(i)$ is to measure the value of **all** events whose time costs are no less than the reference time cost t_r. The following expression defines the negative value function:*

$$V^-(i) = \sum_{j=1}^{N_i} v(t_r - x_j^{(i)})[w^-(\sum_{k=j}^{N_i} \lambda_k^{(i)}) - w^-(\sum_{k=j+1}^{N_i} \lambda_k^{(i)})], i = 1, 2. \tag{5}$$

*The **positive value function** $V^+(i)$ is to measure the value of **all** events whose time costs are less than t_r. The following expression defines the negative value function:*

$$V^+(i) = \sum_{j=N_i+1}^{n_i} v(t_r - x_j^{(i)})[w^+(\sum_{k=j}^{n_i} \lambda_k^{(i)}) - w^+(\sum_{k=j+1}^{n_i} \lambda_k^{(i)})], i = 1, 2. \tag{6}$$

Definition 11. ***General Value Function:*** *Given a transportation mode i ($i = 1, 2$), the **general value function** $V(i)$ is a function that measures the general value of choosing this mode. $V(i)$ consists of the negative and positive value functions:*

$$V(i) = V^-(i) + V^+(i), i = 1, 2. \tag{7}$$

Definition 12. ***Minimum Acceptable Value:*** *Define the* ***minimum acceptable probability*** λ_{min} *as a probability threshold. For* $i = 1, 2, \cdots, n_1$ *and* $j = 1, 2, \cdots, n_2$*, find the minimum* $n_{min}^{(1)}$ *and* $n_{min}^{(2)}$ *such that* $\sum_{j=1}^{n_{min}^{(1)}} \lambda_j^{(1)} > \lambda_{min}$ *and* $\sum_{j=1}^{n_{min}^{(2)}} \lambda_j^{(2)} > \lambda_{min}$*. Define the* ***minimum acceptable value*** *by the following expression:*

$$V(0) = min\{v(t_r - x_{n_{min}^{(1)}}), v(t_r - x_{n_{min}^{(2)}})\}. \tag{8}$$

Definition 13. ***Prospect Probability:*** *Given historical data and geographic information, the* ***prospect probability*** *of choosing transportation modes* $\tau(i)$ *are defined as*

$$\tau(i) = \frac{V(i) - V(0)}{[V(1) - V(0)] + [V(2) - V(0)]}, i = 1, 2. \tag{9}$$

All definitions are for the single segment. To make denotations brief, we omit the number of segment. If it is necessary to mark a segment, we can add a subscript. We denote $\tau_j(i)$ as the prospect probability of choosing mode i in the j-th segment. The prospect probability is the influencing probability, i.e., $q_j(i) = \tau_j(i)$.

4 Experiment Setting

This section details settings of experiments. Algorithms were implemented with C++. Experiments were conducted on 1.7GHz, 2G memory AMD PC running Windows XP.

The database contains 20 day real mobile records provided by China Mobile Communication Corporation. The mobile data of all citizens in a small city are obtained, and for every individual, 24-hour mobile records are equipped. There are approximately 500 individual transportation mode sequences, labeled from individual traveling questionnaire surveys, to test and verify our methodologies. All mobile data are collected in the trunk roads of the city. About 80 percent of data were used as the training set, while the rest was used as the test data.

We make classifications on our test data in several ways, in order to examine the effectiveness of our models. We first call the data set which contains all test data as **all-data set**. Beside the all-data set, we mark two special subsets from the test data: the **normal data set** and the **congested data set**. In the normal data set, users' trips are fully in the normal environment, where the maximum speed of all users is higher than 40 km/h. In the congested data set, users' trips are fully or partly in the congested environment, where the average speed of all users is less than 16 km/h.

Now we discuss the parameter selection. First, we need to select parameters to describe the transition probabilities of modes:

$$P_{i,j} = (p_{i,j})_{i,j \in \{1,2,3\}} = \begin{pmatrix} 0.855 & 0.005 & 0.14 \\ 0.01 & 0.84 & 0.15 \\ 0.17 & 0.08 & 0.75 \end{pmatrix} \tag{10}$$

Second, we select the parameters of the SDL-based approach by the statistical result in training set. In experiments, the means the biking and walking speed are $\mu_2 = 12$ km/h and $\mu_3 = 5$ km/h. The corresponding standard deviations are $\sigma_2 = 2$ and $\sigma_3 = 0.5$. The

selection of the driving speed's mean largely depends on the traffic condition. Hence, we do not fix this parameter; in experiments μ_1 ranges from 25 km/h to 40 km/h. The standard deviation of the driving speed $\sigma_1 = 15$ remains unchanged.

Finally, we will determine the form of the capacity function and the event value function in the CPT-based approach. The capacity function is set as

$$w^+(p) = \frac{p^\gamma}{(p^\gamma + (1-p)^\gamma)^{\frac{1}{\gamma}}},\ w^-(p) = \frac{p^\delta}{(p^\delta + (1-p)^\delta)^{\frac{1}{\delta}}}. \tag{11}$$

where $\gamma = 0.61$ and $\delta = 0.69$. The event value function is set as $v(x) = x^\alpha$ when $x \geq 0$, $v(x) = -\lambda(-x)^\alpha$ when $x < 0$, where $\alpha = 0.88$ and $\lambda = 2.25$.

5 Results

In experiments, F1 score is employed to evaluate the performances [12]. This metric takes a balanced look at the precision and recall values, hence is more comprehensive. The F1 score values of all transportation modes are actually the identification accuracy. There are three tests based on the variations of the low-speed bias L, the mean of driving speed μ_1 and the ambiguous parameter θ. In first two tests, experiments are conducted on the all-data set and the congested data set by the SDL-based approach and the combined approach, and on the normal data set by the SDL-based approach only. In the last test, experiments are conducted on the all-data set and the congested data set by the combined approach. In the tables, we use "SDL" and "SDL + CPT" to simply represents the SDL-based approach and the combined approach.

5.1 The Low-Speed Bias Index Variation

To test how the low-speed bias index L influences the effectiveness of our approaches, we change this parameter in the value range 1, 2, 4, 8, 20, when two other parameters μ_1 and θ are fixed as 32.5 km/h and 1.

All-data Set. Table 4 shows the F1 score values in the all-data set.

Table 4. F1 score results for two approaches in the all-data set

	Driving		Biking		Walking		All (Accuracy)	
	SDL	combined	SDL	combined	SDL	combined	SDL	combined
$L = 1$	0.888	0.938	0.617	0.756	0.906	0.902	0.849	0.909
$L = 2$	0.895	0.939	0.635	0.778	0.900	0.896	0.858	0.914
$L = 4$	0.900	0.940	0.645	0.784	0.901	0.895	0.862	0.916
$L = 8$	0.902	0.941	0.651	0.789	0.901	0.895	0.865	0.917
$L = 20$	0.902	0.941	0.651	0.789	0.901	0.895	0.865	0.917

Generally speaking, a larger L can make the F1 score values higher. The low-speed biased absolute probability of the driving mode exactly depicts the reality of urban transportation. It is plausible that the SDL + CPT combined approach is more effective.

Table 5. F1 score results for the SDL-based approach in the normal data set

	Driving	Biking	Walking	All (Accuracy)
$L = 1$	0.962	0.864	0.933	0.944
$L = 2$	0.962	0.889	0.924	0.946
$L = 4$	0.964	0.897	0.924	0.946
$L = 8$	0.964	0.900	0.924	0.947
$L = 20$	0.964	0.900	0.924	0.947

Normal Data Set. Table 5 shows the F1 score values in the normal set. Notice that only the SDL-based approach is applicable in this dataset.

In the normal data set, the speed features of all transportation modes are all obvious, thus the metrics do not change much when L increases.

Congested Data Set. Table 6 shows the F1 score values in the congested data set.

Table 6. F1 score results for two approaches in the congested data set

	Driving		Biking		Walking		All (Accuracy)	
	SDL	combined	SDL	combined	SDL	combined	SDL	combined
$L = 1$	0.595	0.879	0.408	0.613	0.921	0.896	0.587	0.842
$L = 2$	0.695	0.910	0.425	0.637	0.927	0.896	0.612	0.853
$L = 4$	0.716	0.911	0.434	0.639	0.930	0.896	0.627	0.860
$L = 8$	0.733	0.914	0.441	0.647	0.930	0.896	0.638	0.865
$L = 20$	0.733	0.914	0.441	0.647	0.930	0.896	0.638	0.865

The results in this set is similar to the results in the all-data set. Generally speaking, a larger L makes the identification more accurate. Compared with the situation in the all-data set, the weaknesses of the SDL-based approach are more obvious here.

5.2 The Mean of the Driving Speed Variation

To test how the mean of the driving speed μ_1 influences the effectiveness of our approaches, we change this parameter in the value range 25, 30, 33, 37, 40, when two other parameters L and θ are fixed as 4.5 and 1.

All-data Set. Table 7 shows the F1 score values in the all-data set.

Generally speaking, a smaller μ_1 can make F1 score values higher. The lower mean of the driving speed exactly depicts the reality of urban transportation. Also, it is plausible that the SDL + CPT combined approach is more effective.

Normal Data Set. Table 8 shows the F1 score values in the normal data set. The CPT-based approach is not applicable in this set, and we will present the results of the SDL-based approach only.

In the normal data set, the speed features of all transportation modes are all obvious, thus the metrics do not change much when μ_1 increases.

Table 7. F1 score results for two approaches in the all-data set

	Driving		Biking		Walking		All (Accuracy)	
	SDL	combined	SDL	combined	SDL	combined	SDL	combined
$\mu_1 = 25$	0.908	0.941	0.671	0.797	0.911	0.893	0.873	0.919
$\mu_1 = 30$	0.904	0.940	0.666	0.801	0.899	0.893	0.868	0.917
$\mu_1 = 33$	0.899	0.940	0.642	0.784	0.901	0.894	0.860	0.916
$\mu_1 = 37$	0.891	0.939	0.619	0.760	0.906	0.902	0.852	0.911
$\mu_1 = 40$	0.889	0.937	0.619	0.755	0.905	0.902	0.844	0.905

Table 8. F1 score results for the SDL-based approach in the normal data set

	Driving	Biking	Walking	All (Accuracy)
$\mu_1 = 25$	0.943	0.902	0.920	0.946
$\mu_1 = 30$	0.941	0.908	0.921	0.947
$\mu_1 = 33$	0.933	0.897	0.923	0.946
$\mu_1 = 37$	0.926	0.873	0.934	0.946
$\mu_1 = 40$	0.924	0.837	0.933	0.940

Congested Data Set. Table 9 shows the F1 score values in the congested data set.

The results in this set is similar to the results in the all-data set. Generally speaking, a smaller μ_1 makes the identification more accurate. However, compared with the situation in the all-data set, the weaknesses of the SDL-based approach are more obvious in traffic congestion. It severely confuses the biking mode with the driving mode.

Table 9. F1 score results for two approaches in the congested data set

	Driving		Biking		Walking		All (Accuracy)	
	SDL	combined	SDL	combined	SDL	combined	SDL	combined
$\mu_1 = 25$	0.709	0.889	0.464	0.662	0.946	0.896	0.689	0.873
$\mu_1 = 30$	0.691	0.887	0.453	0.651	0.936	0.896	0.669	0.866
$\mu_1 = 33$	0.647	0.883	0.430	0.639	0.921	0.896	0.587	0.842
$\mu_1 = 37$	0.598	0.879	0.409	0.613	0.921	0.896	0.587	0.842
$\mu_1 = 40$	0.584	0.878	0.403	0.604	0.921	0.896	0.580	0.838

5.3 The Ambiguous Parameter Variation

To test how the ambiguous parameter θ influences the effectiveness of the combined approach, we change this parameter in the value range 0, 0.5, 1, 1.5, 2, when two other parameters μ_1 and L are fixed as 32.5 km/h and 4.5. When $\theta = 0$, the influencing probabilities are actually entirely determined by the SDL-based approach, and this part of data is the control in the test. There will be no experiment on the normal data set.

All-data Set. Table 10 shows the F1 score values in the all-data set.

When $\theta > 0$, results are better than the result when $\theta = 0$, which means the advantage of the combined approach over the SDL-based approach. It is because that more

Table 10. F1 score results for the combined approach in the all-data set

	Driving	Biking	Walking	All (Accuracy)
$\theta = 0$	0.917	0.675	0.906	0.888
$\theta = 0.5$	0.929	0.730	0.903	0.903
$\theta = 1$	0.941	0.775	0.910	0.923
$\theta = 1.5$	0.946	0.794	0.909	0.925
$\theta = 2$	0.946	0.796	0.907	0.925

subsidiary analysis on historical information are done for the ambiguous data.

Congested Data Set. Table 11 shows the F1 score values in the congested data set.

Table 11. F1 score results for the combined approach in the congested data set

	Driving	Biking	Walking	All (Accuracy)
$\theta = 0$	0.758	0.296	0.930	0.697
$\theta = 0.5$	0.833	0.388	0.906	0.779
$\theta = 1$	0.904	0.434	0.910	0.882
$\theta = 1.5$	0.926	0.484	0.949	0.895
$\theta = 2$	0.927	0.448	0.935	0.895

In the congested data set, the advantage of the combined approach is more obvious, as we can see the results when $\theta = 0$ and $\theta > 0$. It is also interesting that the F1 score is not always monotone. There are two possible reasons: 1) the analysis on historical data may cause errors, as people do not always follow their habits; 2) the psychological assumption of the CPT-based approach may not hold under lighter traffic congestion.

6 Related Works

Transportation modes identification has been an interesting topic for many years. To get sophisticated features of people's motions, some special electronic devices are invented, such as GPS, accelerometer and pedometer. Researchers have developed algorithms to infer transportation modes based on GPS [14][15], accelerometer [5] and pedometer [10].

Because of the widespread use of mobile phones, identification algorithms based on mobile data have received much attention in past decades. Different from our analysis on the average speed, some researches focus on the mobile signal strength fluctuation of phones [1]. Also, it is desirable to use the mobile phones to locate the position , which is a main feature of people's motions. However, as the mobile phone can only provide low-level data, many researches are conducted based on the combination of mobile phones and other assistant devices to extract more features of people's motions. There have been some inferring algorithms based on mobile phones equipped with GPS [9] or WiFi [7].

When analyzing observed data, we mainly use the speed distribution law and the cumulative prospect theory. So far, there are few similar studies with our methods; however, there have been some applications of the above methods in other fields of urban transportation. There are proposals to measure the level of traffic congestion by analyzing the speed distribution [6]; also, there are proposals to make route choice using cumulative prospect theory [4]. There are also applications of the transportation modes identification. This work can help analyze and evaluate people's habits [3]; also, this work is valuable in explaining social phenomena [2].

7 Conclusion and Discussions

In this paper, we have revisited the transportation modes identification problem. Only using mobile phones to collect information, we proposed a probabilistic model (with two sub-models) to do inference based on observed data. Under the normal traffic condition, we analyze the average speed of transportation tools and thus adopt the speed distribution law (SDL) based approach; under the congested traffic condition, we offer an alternative approach, i.e., the cumulative prospect theory approach (CPT) based approach to analyze people's choices of transportation modes under uncertainty of traffic congestion. Our method effectively use both current and historical data in different cases. Experiments show that our method performs well in the normal and congested condition alike.

In future, we plan to improve our method in following aspects. First, we will reconsider some basic assumptions of our methods. For example, we can make the refinement on the Markov property in our current work. Second, we can extract some more features from mobile phone data. For example, it may be useful to analyze the ratio of two consecutive average speed, i.e., $\frac{\rho_i}{\rho_{i+1}}$. It is possibly a feature that can distinguish different transportation modes.

References

1. Anderson, I., Muller, H.: Practical Activity Recognition using GSM Data. In Technical Report CSTR-06-016, Department of Computer Science, University of Bristol (2006)
2. Foddy, M., Smithson, M., Schneider, S., Hogg, M.A.: Resolving Social Dilemmas: Dynamic, Structural, and Intergroup Aspects. Psychology Press (1999)
3. Froehlich, J., Dillahunt, T., Klasnja, P., Mankoff, J., Consolvo, S., Harrison, B., Landay, J.: UbiGreen: Investigating a Mobile Tool for Tracking and Supporting Green Transportation Habits. In: Computer Human Interaction (CHI). ACM, New York (2009)
4. Gao, S., Frejinger, E., Ben-Akiva, M.: Adaptive Route Choices in Risky Traffic Networks: A Prospect Theory Approach. Transportation Research Part C 18(5), 727–740 (2009)
5. Kern, N., Schiele, B., Schmidt, A.: Multi-Sensor Activity Context Detection for Wearable Computing. In: Aarts, E., Collier, R.W., van Loenen, E., de Ruyter, B. (eds.) EUSAI 2003. LNCS, vol. 2875, pp. 220–232. Springer, Heidelberg (2003)
6. Ko, J., Guensler, R.L.: Characterization of Congestion Based on Speed Distribution: A Statistical Approach Using Gaussian Mixture Model. In: TRB Committee on Highway Capacity and Quality of Service (AHB40) in Division A (2004)

7. Mun, M., Estrin, D., Burke, J., Hansen, M.: Parsimonious Mobility Classification using GSM and WiFi Traces. In: Proceedings of the 5th Workshop on Embedded Networked Sensors, HotEmNets (2008)
8. Patterson, D.J., Liao, L., Fox, D., Kautz, H.: Inferring High-Level Behavior from Low-Level Sensors. In: Dey, A.K., Schmidt, A., McCarthy, J.F. (eds.) UbiComp 2003. LNCS, vol. 2864, pp. 73–89. Springer, Heidelberg (2003)
9. Reddy, S., Burke, J., Estrin, D., Hansen, M., Srivastava, M.: Determining Transportation Mode On Mobile Phones. In: ISWC (2008)
10. Schneider, P., Crouter, S., Bassett, D.: Pedometer Measures of Free-living Physical Activity: Comparison of 13 Models. Med. Sci. Sports Exerc. 36(2), 331–335 (2004)
11. Tversky, A., Kahneman, D.: Advances in Prospect Theory: Cumulative Representation of Uncertainty. Journal of Risk and Uncertainty 5, 297–323 (1992)
12. van Rijsbergen, C.J.: Information Retrieval, 2nd edn. Butterworth (1979)
13. Zheng, Y.: Mobile Phone Location Determination and Its Impact on Intelligent Transportation Systems. IEEE Transactions on Intelligent Transportation Systems 1(1), 55–64 (2000)
14. Zheng, Y., Xie, X.: Learning Transportation Mode from Raw GPS Data for Geographic Application on the Web. In: WWW 2008 (2008)
15. Zheng, Y., Li, Q., Chen, Y., Xie, X., Ma, W.: Understanding mobility based on GPS data. In: Ubiquitous Computing. ACM, New York (2008)

User Graph Regularized Pairwise Matrix Factorization for Item Recommendation

Liang Du[1,2], Xuan Li[1,2], and Yi-Dong Shen[1]

[1] State Key Laboratory of Computer Science,
Institute of Software, Chinese Academy of Sciences, Beijing 100190, China
[2] Graduate University, Chinese Academy of Sciences, Beijing 100049, China
{duliang,lixuan,ydshen}@ios.ac.cn

Abstract. Item recommendation from implicit, positive only feedback is an emerging setup in collaborative filtering in which only one class examples are observed. In this paper, we propose a novel method, called User Graph regularized Pairwise Matrix Factorization (UGPMF), to seamlessly integrate user information into pairwise matrix factorization procedure. Due to the use of the available information on user side, we are able to find more compact, low dimensional representations for users and items. Experiments on real-world recommendation data sets demonstrate that the proposed method significantly outperforms various competing alternative methods on top-k ranking performance of one-class item recommendation task.

Keywords: Pairwise Matrix Factorization, User Graph, Item Recommendation.

1 Introduction

Recommender systems have become a core component for today's personalized online businesses. Most recent work is on scenarios where users provide explicit feedback, e.g. in terms of ratings. One of the well known example is the Netflix Prize problem. However, Such explicit feedback is hard to collect in many applications because of the intensive user involvement. In fact, most of the feedback in real-world applications is not explicit but implicit. Typical examples include web page bookmarking and Amazon's production recommendation. Therefore, in this paper we focus on item recommendation from implicit, positive only feedback [1]. It is also known as one class collaborative filtering (OCCF) [2]. Its task is to provide personalized ranking on a set of items according to the score of user-item elements based on the feedback information.

In recent years, some matrix factorization models have been proposed to tackle this problem. Pan et.al. [2] proposed to weight and sample unobserved user-item elements and learn a matrix factorization model by minimizing the weighted element-wise squared loss. The essential idea is to treat all missing user-item examples as negative and assign proper weights on these entries. The weighting schemes include uniform, user-specific, and item specific weightings. [3] proposed

J. Tang et al. (Eds.): ADMA 2011, Part II, LNAI 7121, pp. 372–385, 2011.

to approximate the weight matrix with low-rank matrices for large-scale OCCF. [4] proposed to improve OCCF accuracy by exploiting rich context information to weight the binary matrix. One drawback of these methods is that they focus on the optimization of the squared loss on user-item elements, without considering the partial order induced from pairwise element comparison. Such partial order information is valuable in the ranking of items. Rendle et.al. [1] proposed a Bayesian personalized ranking (BPR) optimization framework, which is used for optimizing different kinds of models based on implicit feedback, such as k-nearest-neighbor (KNN) [1], matrix factorization [1] and tensor factorization [5] for the task of item recommendation. BPR's key idea is to make use of partial order of items, instead of using single user-item elements, to train a recommendation model.

On the other hand, the idea of incorporating contextual information to a recommendation algorithm to improve the performance of rating based prediction tasks is used in [6,7]. Such contextual information includes neighborhood information in the user-item rating matrix [6], content information such as user's occupation and item's genre [6,7], and social network connections such as social trust/distrust network and social friend relationships [8,9,10,11]. These models can be seen as an integration of memory based and model based approaches. Apparently, these extensions are based on element-wise matrix factorization models for rating prediction tasks.

These observations suggest that it is desirable to combine the pairwise matrix factorization based on BPR and the contextual information in a unified model. In this paper we propose such a model, called *user graph regularized pairwise matrix factorization* (UGPMF), to seamlessly integrate user side contextual information into the pairwise matrix factorization procedure. Specifically, we construct similarity graphs on user side by exploiting the available user information, and use the graphs to regularize the pairwise matrix factorization procedure based on BPR. Our method inherits the advantage of pairwise matrix factorization method [1]. Due to the use of user information, it also has the merits of memory based methods. It can alleviate the overfitting problem suffered by the matrix factorization model by the additional graph regularization. Experiments on real-world data sets show that the proposed method can effectively improve the top-k ranking performance.

The paper is organized as follows. Section 2 introduces some notation and reviews the matrix factorization based on Bayesian personalized ranking [1]. Section 3 describes the details of our model. Section 4 gives the experimental results, and Sction 5 concludes the paper and presents some future work.

2 Matrix Factorization Based on Bayesian Personalized Ranking

Let $U = \{u_1, u_2, \ldots, u_n\}$ be a group of users, $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items, and $\boldsymbol{X} = (X_{ui})_{n \times m} \in \{0, 1\}^{n \times m}$ be the user-item binary preference matrix with n users and m items, such as a typical who-bought-what customer-

product matrix. $\boldsymbol{X}_{ui} = 1$ denotes customer-product purchases, while $\boldsymbol{X}_{ui} = 0$ means that no purchase was made.

In a matrix factorization based model, we usually seek two low-rank matrices $\boldsymbol{W} \in \mathcal{R}^{n \times D}$ and $\boldsymbol{H} \in \mathcal{R}^{m \times D}$. The row vectors $\boldsymbol{W}_u, 1 \leq u \leq n$ and $\boldsymbol{H}_i, 1 \leq i \leq m$ represent the low dimension representations of users and items respectively.

2.1 Bayesian Personalized Ranking

BPR's key idea is to use partial order of items, instead of single user-item examples, to train a recommendation model. It allows the interpretation of positive-only data as partial ordering of items. When we observed a positive user-item example of user u on an item i, e.g. user u viewed or purchased item i, we assume that the user prefers this item than all other non-observed items. Formally we can extract a pairwise preference dataset $\mathcal{P} : U \times I \times I$ by

$$\mathcal{P} := \{(u, i, j) | i \in I_u^+ \wedge j \in I \setminus I_u^+\} \tag{1}$$

where I_u^+ and $I \setminus I_u^+$ are the positive item set and missing set associated with user u, respectively. Each triple $(u, i, j) \in \mathcal{P}$ says that user u prefers item i than j. BPR optimization criterion [1] aims to find an arbitrary model class to maximize the following posterior probability over these pairs.

$$\text{BPR-OPT} = -\sum_{(u,i,j)\in\mathcal{P}} \ln \sigma(\hat{x}_{uij}) + \lambda_\Theta(\Theta) \tag{2}$$

where Θ represents the parameter of an arbitrary model class, and λ_Θ are model specific regularization parameters.

Note that extracting pairwise preferences has been widely used in learning to rank tasks [12]. The BPR optimization criterion is actually the cross entropy cost function (logistic loss) over pairs [13].

2.2 Matrix Factorization Based on BPR (BPR-MF)

BPR-MF learns two low-rank matrices $\boldsymbol{W}$ and $\boldsymbol{H}$. For a specific user-item example the score is computed by

$$\hat{x}_{ui} = \sum_{d=1}^{D} W_{ud} H_{id}$$

In order to estimate whether a user prefers one item over another, BPR-MF optimizes the following objective function:

$$\mathcal{O}_1 = -\sum_{u=1}^{n} \sum_{i \in I_u^+} \sum_{j \in I \setminus I_u^+} \ln \sigma(\hat{x}_{uij}) + \alpha(||\boldsymbol{W}||^2 + ||\boldsymbol{H}||^2) \tag{3}$$

where $\hat{x}_{uij} = \hat{x}_{ui} - \hat{x}_{uj}$, σ denotes the logistic function $\sigma(x) = 1/(1 + \exp(-x))$, and α is a regularization parameter for complexity control.

3 User Graph Regularized Pairwise Matrix Factorization

In this section we first describe how to construct user graphs based on different user information, then we propose our UGPMF model based on the computed user graphs.

3.1 User Graph Regularization

Graph regularization has been wildly used in dimensionality reduction [14], clustering [15] and semi-supervised learning [16]. The key assumption of graph regularization is that if two users u and v are similar, the latent features W_u and W_v discovered by pairwise matrix factorization (PMF) procedure are also close to each other. This can be achieved by minimizing the following objective function:

$$\begin{aligned}\mathcal{O}_2 &= \frac{1}{2}\sum_{u}^{n}\sum_{v}^{n} S_{u,v}||\boldsymbol{W}_{u.} - \boldsymbol{W}_{v.}||^2 \\ &= \frac{1}{2}\sum_{u=1}^{n}\sum_{v=1}^{n}\left[S_{u,v}\sum_{d=1}^{D}(W_{ud} - W_{vd})^2\right] \\ &= \frac{1}{2}\sum_{d=1}^{D}\left[\sum_{u=1}^{n}\sum_{v=1}^{n} S_{u,v}(W_{ud} - W_{vd})^2\right] \\ &= \sum_{d=1}^{D} W_{*d}^T \boldsymbol{L} W_{*d} = \mathrm{tr}(\boldsymbol{W}^T\boldsymbol{L}\boldsymbol{W})\end{aligned} \tag{4}$$

where $S_{u,v}$ is the similarity between user u and v computed from the available information, $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{S}$ is called the graph Laplacian [17] with $\boldsymbol{D}$ being a diagonal matrix whose diagonal entries are row sums of $\boldsymbol{S}$, $D_{uu} = \sum_v S_{u,v}$, and $\mathrm{tr}(\cdot)$ denotes the trace of a matrix.

The crucial part of graph regularization is the definition of the user graph $\boldsymbol{S}$, which encodes desired information. In the following, we will describe how to construct proper user graph to encode various useful information, such as neighborhood information, demographic information, and social networks.

Neighborhood Information. The neighborhood information of users in the user-item matrix have been widely used in user-oriented memory-based methods [18,19]. The basic assumption of user-oriented memory-based methods is: if two users have similar interests on common items, then they probably have similar interests on the other items. This can be embodied by a user similarity graph defined as follows:

$$S_{u,v} = \begin{cases} sim(X_u, X_v) \text{ if } u \in \mathcal{N}(v) \text{or } v \in \mathcal{N}(u) \\ 0 \qquad\qquad\qquad\qquad \text{otherwise} \end{cases} \tag{5}$$

where $\mathcal{N}(u)$ denotes the K-nearest neighbor of user u, and $sim(X_u, X_v)$ is the similarity between user u and user v computed based on user-item binary matrix.

User's Demographic Information. Demographic information of users is popular in content-based recommendation systems. User demographic information includes age, gender and occupation. We denote by F_u the feature vector which characterizes the demographic information of user u. The above assumption on users similar interests can be captured by a user similarity graph constructed as follows:

$$S_{uv} = sim(F_u, F_v)$$

S_{uv} is the similarity between the feature vectors of user u and user v.

Social Network. Traditional recommender systems ignore social relationships among users. However, the recommendation is sometimes a social activity. For example, we ask our friends for recommendation of movies or cellphones. [11] shows that social friendship can be employed to improve traditional recommender systems. Given a social network, the user graph can be represented by the similarity between two users. It can be defined in terms of the number of shared friends or whether they are friends.

It is important to note that other sources which contain different user information can also be used to construct the user graph, such as user tagging history in a social tagging system, user clickthrough log in web search, etc.

3.2 Integration of PMF and User Graph Regularization

We propose a user graph regularized pariwise matrix factorization (UGPMF) to integrate user information into the pairwise matrix factorization. It combines the above two criteria (3) and (4) and minimizes the following objective function:

$$\begin{aligned} f = & -\sum_{u=1}^{n} \sum_{i \in I_u^+} \sum_{j \in I \setminus I_u^+} \ln \sigma(\hat{x}_{uij}) \\ & + \frac{\alpha}{2}(||\boldsymbol{W}||^2 + ||\boldsymbol{H}||^2) \\ & + \frac{\beta}{2} tr(\boldsymbol{W}^T \boldsymbol{L} \boldsymbol{W}) \end{aligned} \tag{6}$$

where β is an additional regularization parameter used to balance the information from the user-item matrix and the user side information.

The above formulation inherits the advantage of pairwise matrix factorization method [1] and has the merits of memory based methods. It can also alleviate the overfitting problem suffered by the matrix factorization procedure with the additional graph regularization.

Due to the huge number of preference pairs (see (1)), it is expensive to update the latent features over all pairs. Hence, like in BPF-MF [1], we adopt a stochastic gradient descent procedure to compute $\boldsymbol{W}$ and $\boldsymbol{H}$. We choose the preference triples (u, i, j) randomly (uniformly distributed) and update the corresponding latent features by the following gradients

$$\frac{\partial f}{\partial W_u} = -\frac{\exp(-\hat{x}_{uij})}{1+\exp(\hat{x}_{uij})}(H_i - H_j) + \alpha W_u + \beta \sum_{v \in \mathcal{N}^+(u)} S_{uv}(W_u - W_v) + \beta \sum_{v \in \mathcal{N}^-(u)} S_{uv}(W_u - W_v) \tag{7}$$

$$\frac{\partial f}{\partial H_i} = -\frac{\exp(-\hat{x}_{uij})}{1+\exp(\hat{x}_{uij})} W_u + \alpha H_i \tag{8}$$

$$\frac{\partial f}{\partial H_j} = \frac{\exp(-\hat{x}_{uij})}{1+\exp(\hat{x}_{uij})} W_u + \alpha H_j \tag{9}$$

where $\mathcal{N}^+(u)$ and $\mathcal{N}^-(u)$ denote the outlink neighbors and inlink neighbors of user u respectively. We use a constant learning rate to update the latent features. The process of estimating the low-rank matrices $\boldsymbol{W}$ and $\boldsymbol{H}$ is described in algorithm 1.

Algorithm 1. Learning procedure of UGPMF

Input: training data $\boldsymbol{X}$, user graph $\boldsymbol{S}$, learning rate η, regularization parameters α and β

Output: $\boldsymbol{W}$ and $\boldsymbol{H}$

```
Compute Laplacian matrix L based on S
Initialize W and H
repeat
    draw (u, i, j) uniformly from U × I × I
    x̂_uij ← x̂_ui − x̂_uj
    update W_u, the u-th row of W according to Eq. (7)
    update H_i, the i-th row of H according to Eq. (8)
    update H_j, the j-th row of H according to Eq. (9)
until convergence
return W and H
```

It is important to note that due to the pairwise comparison in pairwise matrix factorization procedure, the optimization and regularization of PMF are different from other graph regularized element-wise matrix factorization [7,11]. In [11,7] they both take a batch algorithm to update the latent features where the gradient of these latent features are accumulated over all pairs. In PMF, the number of preference pairs derived from the original binary dataset is $n\bar{m}(m-\bar{m})$, where $\bar{m}$ is the average positive items for each user. This number is extremely huge. For example the MovieLens dataset (see section 3.1) used in our experiments contains only 100,000 positive examples, but the number of induced pairs is more than 140,000,000. This size is usually prohibitive for batch updating and storage. Hence we resort to stochastic optimization algorithms. In element-wise matrix factorization (MF) approaches, both user and item graphs can be constructed

to regularize the MF procedure. However, adding graph regularization on item side in PMF makes the optimization procedure costly; e.g. to update the latent features of each triple (u, i, j), both the in-degree neighbors and out-degree neighbors of item i and j are involved.

4 Experiments

We conduct several experiments to compare the recommendation quality of the proposed method with other state-of-the-art item recommendation methods.

4.1 Datasets

We use two datasets to evaluate our algorithm. The first one, Movielens[1], is a widely used movie recommendation dataset. It contains 100,000 ratings with scale 1-5, which are obtained from 943 users over 1682 movies. To simulate binary responses for item recommendation task, we removed all ratings below 4, and relabeled ratings 4 and 5 as 1. This treatment has also been used in [20]. To evaluate the impact of user demographic information, we extract 30 features (2 features to characterize the user's gender, and 7 features to category the user's age and 21 features to describe the user's occupation) to represent each user.

The second dataset, User-Tag[2] is crawled from a social bookmarking site[3], which has been used in [2]. It contains the tag history of 3000 users on 2000 tags. In total 246,346 posts are recorded.

In our experiments, we choose cosine similarity to construct the user graph based on user-item binary matrix or user demographic information (other similarity measures can also be used).

4.2 Evaluation Criteria

We assess the recommendation performance of each model by comparing the top suggestions of the model to the true positive actions by a user. We consider three measures commonly used to evaluate top-k ranking performance in the IR community. The reason for concentrating on top-k results is that in a recommender system users are usually interested in the top-k results than a sorted order of the entire items. We choose $k = 5$ since many recommender systems recommend a similar number of items for each user.

– Prec@k: The precision at position k of a ranked item list for an given user is defined as

$$\text{Prec@}k = \frac{1}{k}\sum_{i=1}^{k}\mathbf{1}(x_{r_i} = 1)$$

[1] http://www.grouplens.org/
[2] http://www.rongpan.net/
[3] http://del.icio.us/

where $\mathbf{1}(x_{r_i} = 1)$ is 1 if x_{r_i}, the label of item ranked at position i, is positive, and 0 otherwise. Hence, Prec@k considers the positive items and computes the fraction of such items in the top-k elements of the ranked list.

– NDCG@k: The normalized discounted cumulated gain (NDCG) at position k of a ranked item list for an given user is defined by a position discounting function and a grades weighting function. It is normalized by the NDCG value of perfect ranking of these items.

$$\text{NDCG@}k = Z \sum_{i=1}^{k} \frac{2^{x_{r_i}} - 1}{\log(1+i)}$$

where Z is the DCG value of ideal ranking. In our binary item recommendation task, NDCG@k can be viewed as a position discounted and normalized version of Prec@K.

– MAP: MAP denotes the mean of the Average Precision over all test users. Average Precision (AP) is defined as the mean over the precision scores for all positive items. It is given by

$$\text{AP} = \frac{\sum_{i=1}^{m} \text{Prec@i} \cdot \mathbf{1}(x_{r_i} = 1)}{\sum_{i=1}^{m} \mathbf{1}(x_i = 1)}$$

where $\mathbf{1}(x_{r_i} = 1)$ is defined above.

4.3 Comparison Settings

We create random training-test splits of positive user-item entries in the ratio 60%-to-40% respectively. All results reported here are averaged over 10 random rounds. The hyperparameters for all methods are optimized via grid search in the first round and kept constant in the remaining 9 rounds.

In order to show the effectiveness of our recommendation approach, we compare the recommendation results with the following 6 baseline models. The first two models are considered as weak baselines, while the other four are considered as strong baselines.

– PopRank: The first method sorts all the items based on their popularity, so that the top recommended items are the most popular one in terms of the number of times bought by users. This simple measure is supposed to have reasonable performance, as people tend to focus on few popular items.
– AMAN: AMAN stands for All Missing as Negative examples. In AMAN, all non-positive elements are assigned to 0, and the Alternative Least Squares optimization is adopted to optimize the factorization. An ordered list of items can be obtained by sorting the scores calculated by multiplying two low-rank matrices.
– wAMAN (uniform) [2]: It is a weighted version of AMAN where negative elements are treated as zeros but a uniform weight with value less than 1 is additionally imposed. The intuition behind this is that the confidence of

unknown user-item examples being negative is lower than the confidence of positive examples. In particular, we adopt the same weight scheme in [20] and report the best performance over the following weights

$$\delta_k = \frac{n_+}{2^k n_0} \tag{10}$$

where n_+ is the number of positive examples and n_0 is the number of zero-entries in X, with $0 \leq k \leq 5$.

- wAMAN (user) [2]: The weights of zero-entries are proportional to the number of positive items associated with each user. We report the best performance over the following weights $\forall u : w_{ui} = \delta_k \sum_i X_{ui}, k = 0, 1, , 2, 3$, where δ_k is as defined above. The intuition behind this is that if a user has more positive examples, then it is more likely that the other items are less preferred, that is, the missing data for this user is negative with higher probability.
- wAMAN (item) [2]: The weights of zero-entries are linear to the number of users associated with each item. We report the best performance over the weights $\forall i : w_{ui} = \delta_k(m - \sum_u X_{ui}), k = 0, 1, 2, 3$, where δ_k is the same as above. The intuition is that if an item has less positive examples, then the missing data for this item is negative with higher probability.
- BPR-MF: This is the pairwise matrix factorization approach with BPR optimization criterion proposed in [1]. We use the MyMediaLite[4] package in their implementation.

4.4 Results

To make a fair comparison, we report the results of all matrix factorization approaches by setting the number of the latent features $D = 10$. Similar improvement is also observed with other settings of D. The results reported in Table 1 are the top-k rank performance of UGPMF and other baselines on MovieLens dataset. The better results are shown in bold. We observe that our method with user graph constructed by demographic information and neighbor information

Table 1. Performance on the MovieLens dataset

Methods	Prec@5	NDCG@5	MAP
PopRank	0.2375	0.2566	0.1503
AMAN	0.2442	0.2624	0.1440
wAMAN (Uniform)	0.3192	0.3429	0.2279
wAMAN (Item)	0.3417	0.3672	0.2259
wAMAN (User)	0.3556	0.3709	0.2469
BPRMF	0.3590	0.3720	0.2505
UGPMF (User Demographic)	0.3769	0.3906	0.2578
UGPMF (User Neighbor)	**0.3826**	**0.3986**	**0.2633**

[4] http://www.ismll.uni-hildesheim.de/mymedialite/index.html

in binary matrix significantly improves the top-k ranking performance of item recommendation, and outperforms the compared baselines consistently. We report the results of UGPMF and compared baselines on User-Tag dataset in Table 2. It is clear that UGPMG achieves better performance in all the measures than the compared baselines. We also conduct a pairwise t-test with a standard 0.05 significance level, which further indicates that all the improvements obtained by UGPMF are significant.

Table 2. Performance on the User-Tag dataset

Methods	Prec@5	NDCG@5	MAP
PopRank	0.2408	0.2497	0.1178
AMAN	0.2409	0.2487	0.1155
wAMAN (Uniform)	0.2552	0.2563	0.1376
wAMAN (Item)	0.2546	0.2640	0.1271
wAMAN (User)	0.2560	0.2645	0.1274
BPRMF	0.2607	0.2682	0.1421
UGPMF (User Neighbor)	**0.2731**	**0.2767**	**0.1489**

We further compare UGPMF with BPR-MF in their performance on users with different number of observed positive examples. The results are shown in Figure 1, from which we see that UGPMF outperforms BPR-MF for all users and the improvement is more significant for uses with only few observed positive examples. This is a very promising property of UGPMF because most users have only a small number of positive examples in real-world situations.

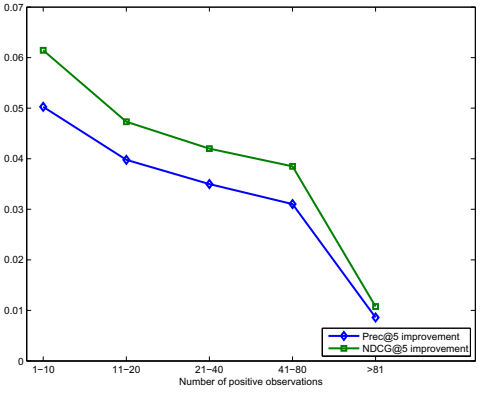


Fig. 1. Top-k ranking performance improvement of UGPMF over that of BPR-MF on different user scales

4.5 Impact of Parameters

UGPMF and BPR-MF have the common parameter D (the number of latent features). UGPMF has two additional parameters, the regularization parameter β and the number of nearest neighbors K in user graphs. We conduct several experiments on the MovieLens dataset to show the impacts of these parameters.

Impact of Parameter β. The main advantage of UGPMF is that it incorporates user information, which helps predict user's preferences. In our model β is used to balance the information coming from the user-item matrix and other user related sources. If $\beta = 0$, we do not use additional user-side information at all and hence our model degenerates to BPR-MF. Table 3 shows the impact of β on Prec@5 and NDCG@5, where we set $D = 10$ and $K = 50$.

As we can see from Table 3, the value of β impacts the performance significantly, which demonstrates that integrating the user-item matrix and user side information greatly improves the top-k recommendation results. It can be also observed that our method achieves the best performance when $\beta \in [0.005, 0.05]$. This relatively wide range shows that the parameter β of our model is easy to tune.

Table 3. The performance of UGPMF vs. parameter β

	0.001	0.005	0.01	0.05	0.1
Prec@5	0.35	0.377	0.38	0.361	0.33
NDCG@5	0.356	0.391	0.398	0.384	0.35
MAP	0.242	0.26	0.263	0.25	0.23

Impact of Parameter K. Similar to memory based collaborative filtering algorithms, the size of neighborhood will affect the performance of our algorithm since it needs to construct K nearest neighbor user graph. We run our algorithm with varying neighborhood size. Figure 2 shows the change of performance (Prec@5 and NDCG@5) when the neighborhood size K increases from 10 to 100 with $\beta = 0.01$ and $D = 10$.

We can see that Prec@5 and NDCG@5 gradually increase as the neighborhood size increases from 10 to 40 since the similarities can be estimated more accurately given more neighbors. However, we also observe that the performance starts to decrease as the neighborhood size exceeds 30, which is due to that many non-similar users introduce incorrect information in regularization as the neighborhood size is too large.

Impact of Parameter D. We report the results of our method and BPF-MF with varying D with $K = 50$ and $\beta = 0.01$. Figure 3 shows that the top-k ranking performance increases as D increases. This agrees with our intuition, that is, the more latent features we have, the more information can be represented by the latent feature vectors. The figure also shows that the improvement in Prec@5 and NDCG@5 becomes smaller as D continues to increase. When D becomes

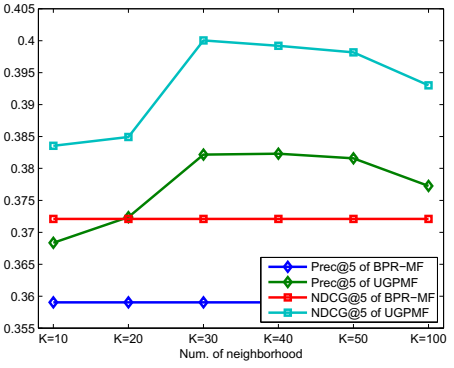


Fig. 2. Top-k ranking performance as a function of size of neighborhood

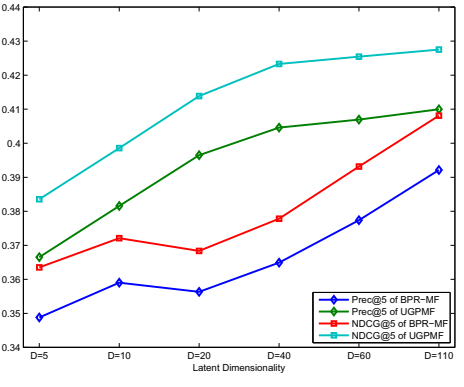


Fig. 3. Top-k ranking performance as a function of latent dimensionality

large enough, there is essentially no significant improvement because the useful information has already been represented well by the existing latent features. It is clear that UGPMF performs better than BPR-MF at all levels.

5 Conclusion

In this paper, we integrate the internal and external user information into the process of pairwise matrix factorization procedure for one class item recommendation task, and propose a unified model UGPMF. One interesting property of the proposed method is that it can alleviate the overfitting problem suffered by most matrix factorization based models [1, 2], since it uses additional user information to regularize the factorization process. Experiments on benchmark datasets demonstrate that the proposed method outperforms many state-of-the-art methods in top-k ranking performance, such as wAMAN [2], BPR-MF [1], etc.

As ongoing work, we are testing our method over more benchmark datasets. We are also investigating more sophisticated models that accommodate valuable user information.

Acknowledgments. We would like to thank all anonymous reviewers for their helpful comments. This work is supported in part by the National Natural Science Foundation of China (NSFC) grants 60970045 and 60833001.

References

1. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: Bpr: Bayesian personalized ranking from implicit feedback. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, pp. 452–461. AUAI Press (2009)
2. Pan, R., Zhou, Y., Cao, B., Liu, N., Lukose, R., Scholz, M., Yang, Q.: One-class collaborative filtering. In: IEEE International Conference on Data Mining (ICDM), pp. 502–511 (2008)
3. Pan, R., Scholz, M.: Mind the gaps: weighting the unknown in large-scale one-class collaborative filtering. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 667–676. ACM (2009)
4. Li, Y., Hu, J., Zhai, C., Chen, Y.: Improving one-class collaborative filtering by incorporating rich user information. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, pp. 959–968. ACM (2010)
5. Rendle, S., Schmidt-Thieme, L.: Pairwise interaction tensor factorization for personalized tag recommendation. In: Proceedings of the Third ACM International Conference on Web Search and Data Mining, pp. 81–90. ACM (2010)
6. Gu, Q., Zhou, J., Ding, C.: Collaborative Filtering: Weighted Nonnegative Matrix Factorization Incorporating User and Item Graphs. In: Proceedings of the SIAM International Conference on Data Mining, pp. 199–210. SIAM (2010)
7. Zhen, Y., Li, W., Yeung, D.: TagiCoFi: tag informed collaborative filtering. In: Proceedings of the Third ACM Conference on Recommender Systems, pp. 69–76. ACM (2009)
8. Ma, H., Yang, H., Lyu, M., King, I.: SoRec: social recommendation using probabilistic matrix factorization. In: Proceeding of the 17th ACM Conference on Information and Knowledge Management, pp. 931–940. ACM (2008)
9. Ma, H., King, I., Lyu, M.: Learning to recommend with social trust ensemble. In: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 203–210. ACM (2009)
10. Ma, H., Lyu, M., King, I.: Learning to recommend with trust and distrust relationships. In: Proceedings of the Third ACM Conference on Recommender Systems, pp. 189–196. ACM (2009)
11. Ma, H., Zhou, D., Liu, C., Lyu, M., King, I.: Recommender systems with social regularization. In: Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, pp. 287–296. ACM (2011)
12. Liu, T.: Learning to rank for information retrieval. Foundations and Trends in Information Retrieval 3(3), 225–331 (2009)
13. Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G.: Learning to rank using gradient descent. In: Proceedings of the 22nd International Conference on Machine Learning, pp. 89–96. ACM (2005)

14. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural Computation, 1373–1396 (2003)
15. Cai, D., He, X., Han, J., Huang, T.: Graph regularized non-negative matrix factorization for data representation. IEEE Transactions on Pattern Analysis and Machine Intelligence (2010)
16. Belkin, M., Niyogi, P., Sindhwani, V.: Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. The Journal of Machine Learning Research 7, 2399–2434 (2006)
17. Chung, F.: Spectral graph theory. Regional Conference Series in Mathematics American Mathematical Society, vol. 92, pp. 1–212. American Mathematical Society (1997)
18. Breese, J., Heckerman, D., Kadie, C., et al.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, vol. 52, Morgan Kaufmann Publishers, Madison (1998)
19. Sarwar, B., Karypis, G., Konstan, J., Reidl, J.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th international conference on World Wide Web, pp. 285–295. ACM (2001)
20. Sindhwani, V., Bucak, S., Hu, J., Mojsilovic, A.: One-class matrix completion with low-density factorizations. In: 2010 IEEE International Conference on Data Mining, pp. 1055–1060. IEEE (2010)

Using Predicate-Argument Structures for Context-Dependent Opinion Retrieval

Sylvester Olubolu Orimaye, Saadat M. Alhashmi, and Siew Eu-Gene

Monash University, Sunway Campus, Malaysia
{sylvester.orimaye,alhashmi,siew.eu-gene}@monash.edu

Abstract. Current opinion retrieval techniques do not provide context-dependent relevant results. They use frequency of opinion words in documents or at proximity to query words, such that opinionated documents containing the words are retrieved regardless of their contextual or semantic relevance to the query topic. Thus, opinion retrieved for the qualitative analysis of products, performance measurement for companies, and public reactions to political decisions can be largely biased. We propose a sentence-level *linear relevance model* that is based on subjective and semantic similarities between predicate-argument structures. This ensures opinionated documents are not only subjective but semantically relevant to the query topic. The *linear relevance model* performs a linear combination of a popular *relevance model*, our proposed *transformed terms similarity* model, and a popular *subjectivity* mechanism. Evaluation and experimental results show that the use of predicate-argument structures improves performance of opinion retrieval task by more than 15% over popular TREC baselines.

Keywords: Context-dependent opinion search, subjectivity, natural language processing, semantics, predicate-argument structures.

1 Introduction

Opinionated documents (documents that contain opinion) in opinion-driven sources such as collection of blogs called *blogosphere* (e.g. Google Blogs Search[1] and Technorati Blog Directory[2]) and review websites (e.g. *reviewcenter*[3]) often contain opinion words used in different semantic contexts. However, current opinion retrieval systems only retrieve opinionated documents as long as they contain certain opinion words. Usually, the opinion words may be within a proximity window [1] (a certain distance of opinion words to a query word) or randomly located within the document. For example, consider an opinion finding scenario below:

User query topic: *Why Apple made huge sales on iPhones?*
A relevant blog: "*One [interesting] piece of information revealed by the report is that [Apple] may be its own [worst] [enemy]. The numbers show that iPod sales*

[1] http://blogsearch.google.com/
[2] http://technorati.com/
[3] http://www.reviewcentre.com/

J. Tang et al. (Eds.): ADMA 2011, Part II, LNAI 7121, pp. 386–402, 2011.

[fell] by about 20 percent – a trend that has been in place since the [iPhone] 3G hit the market. While [Apple] noted that most of those iPod [sales] are iPod touches, which run on [Apple's] iOS platform and therefore spread its mobile software's influence and gather revenue through the iTunes App Store."

If we assume a unified opinion retrieval model that combines topical relevance and opinion relevance to retrieve the above document, it is quite obvious that the result has been favored by only the frequency or co-occurrence of query words (i.e. Apple, iPhone, sales) used in the *topical relevance* and then the proximity or presence of opinion words (i.e. interesting, worst, enemy, fell) used in the *opinion relevance*. However, if we consider the semantic context of the query topic based on human judgment, it could be observed that the query topic and the retrieved document are not semantically relevant. Consequentially, opinions retrieved for the qualitative analysis of products, performance measurement for companies, and public reactions to political decisions are more likely to be largely biased.

Few approaches have made attempts to perform sentence-level opinion retrieval, for example, by merging opinionated nature of individual sentences (sentiment flow) with the degree of relatedness of individual sentences to the given query (topic relevance flow). The result forms a *sentiment-relevance flow* (SRF) [2] that is still largely based on frequency of opinion words within sentences.

In this paper, our focus is to make opinionated documents more semantically relevant to the given query topic. We present a novel approach that uses semantic similarities between predicate-argument structures in grammatical-tree derivations of natural language queries and well formed subjective sentences to retrieve context-dependent opinions from blog documents. We also show that with careful analysis and evaluations, a linear combination of models can still perform as good as non-linear combination model for opinion retrieval task such as presented in [1].

Inspired by the fact that query topics do contain opinion words [3], we show that it is possible to model opinion relevance from query topics (assuming queries given in natural language sentences) by using a popular *relevance model* [4] to form a more efficient *linear relevance model* for opinion retrieval. To the best of our knowledge, no existing work has proposed to model opinion relevance directly using predicate-argument structures. Most works use standard information retrieval (IR) techniques (e.g. BM25, Language Models (LM), Machine Learning (ML), Lexicon-based) [5]. Rather, we exploit the use of syntactic parser to derive predicate-argument structures from grammatical parse-trees. We use the description fields of the 2008 TREC query topics available in natural language sentences with an average of 10 words per each query. Therefore, the scope of this study does not cover queries or sentences that are not well formed.

We compare the predicate-argument of each query with the predicate-argument of each subjective sentence in a given document [6]. Intuitively, opinionated documents are ranked by considering subjective sentences that have the same *predicate-argument relations* with the given query topic. Predicate-argument relation has been used as viable feature to determine contextual or semantic similarity between two sentences [7,6,8]. In the same manner, such similarity defines relevance between a query topic and subjective sentences within an opinionated document.

We perform experiments on TREC Blog08 using only predicate-argument relations to retrieve relevant blog documents. The result of our experiment showed improved

performance. Our study show s semantically relevant subjective sentences determine the opinion required by the user and opinionated information is better shown in each relevant sentence than its surroundings. Thus, frequency of semantically relevant subjective sentences is used to retrieve relevance of opinionated documents instead of frequency or proximity of opinion words to query words.

For the rest of this paper, we discuss related work and their limitations in section 2. In section 3, we discuss the problem formulations and our methodologies. In section 4, we discuss the evaluations and experiment performed on TREC Blog08. Finally, section 5 presents conclusion and future work.

2 Related Work

Opinion polarity detection techniques have recorded some level of success [9]. In opinion polarity detection, specific keywords within a document are labeled with a particular polarity (e.g. positive or negative). However, determining an effective way to differentiate between what is positive and what is negative is still an unsolved problem[10][16]. For example in [10], the presence of *ironic* phrases and inverted polarity in opinionated documents led to lower precision for positive opinions with just 77% accuracy. As a result of this, the choice of individual words for polarity detection in opinionated documents is still a big challenge.

Subjectivity detection shows if a sentence contains opinion or not [11]. Words or sentences that contain opinions are systematically categorized according to the degree of subjectivity shown by their lexicons. However, an automatic and effective way to detect hidden and inverted polarities in phrases and sentences is still a huge research challenge [10]. According to Pang and Lee [11], subjectivity is a two-state task and can be interpreted differently in some cases. It is still very challenging to effectively determine appropriate subjectivity state in many documents that contain opinion.

Lexicon-based approaches consider domain-specific evidences to form lexicons for opinion retrieval [12]. For generating lexicons, individual opinionated keywords are selected from each sentence in a document. However, we believe opinionated words alone cannot completely and independently express the overall opinion contained in a document, without taking into consideration the grammatical dependencies between words. We suggest opinion should be syntactically retrieved from a complete sentence rather than individual words. Individual keywords in the lexicon might have been selected from varying grammatical contexts.

Probabilistic approaches are commonly used to evaluate and explain theoretical foundations for information retrieval models [13-14]. In many cases, *estimates* or *assumptions* made in probabilistic approaches may not be practically applicable to real scenarios. It is only effective where there are high chances of frequency of observations as applicable to *BM25*[15] and *Divergence from Randomness* probabilistic models. Some probabilistic models use proximity density information to calculate probabilistic opinion score for individual query terms [16]. However, proximity of words to some of the query terms may not reflect the semantic context at which opinion is required.

Language model approach combines prediction of occurrence for natural language words and then shows a probabilistic interpretation of such occurrences. A common

limitation is the estimation of model parameters. It is often difficult to effectively model a document to give a higher probability of relevance to user query. For example, [17] addressed the parameter tuning or optimization problem in higher order language models by incorporating different component models as features. A common approach is to perform smoothing at varying levels to ensure high chance of document retrieval. However, this approach usually leads to having higher number of model parameters, where all of such parameters would also require optimization at different levels.

3 Problem Formulation

3.1 The Natural Language Approach

It is nontrivial to identify contextual opinions that are relevant to the given query. There is often a need for deep understanding of natural language by using a technique that structurally combines different natural language processing (NLP) techniques such as Part-Of-Speech (POS) tagging, Stemming, Stop Words contributions, and Word Sense disambiguation. The individual usages and the different ad-hoc combinations of some of the above NLP techniques amount to greater computational cost even at lesser efficiency. We attempt to overcome this challenge by avoiding the ad-hoc usages of different NLP techniques in opinion retrieval. Thus, we use a full syntactic parser, specifically, Categorial Combinatory Grammar (CCG) [18]. CCG parser is a more lexicalized grammar than phrase structure grammar. We chose CCG because of its distinct ability to integrate syntactic and semantic information in its parse process. Its derivation also shows the short and long-range word-word dependencies in the corresponding predicate-argument structure [6].

CCG does one-time complete NLP processes required for showing the underlying meaning in each given natural language query or well formed sentence. It has a relatively straightforward way of providing a compositional semantics for the intending grammar by providing completely transparent interface between syntax and semantics [19]. We use a log-linear CCG parsing model by Clark and Curran [19], supported by the result of their study that highly efficient parsing and large-scale processing is possible with linguistically motivated grammar such as CCG [20]. CCG has the advantage to recover long-range dependencies in natural language sentences with labeled precision and recall values of 99.80% and 99.18% respectively [19]. Moreover, it has a parsing speed of 1.9 minutes on section 23 of the Wall Street Journal (WSJ) Penn Treebank [19], compared to 45 minutes by Collins parser, 28 minutes by Charniak parser, and 11 minutes by Sagae parser. The resulting CCG output parse tree for each query or sentence is the grammatical tree derivation from which the needed predicate-argument structure is derived.

3.2 Grammatical Tree Derivations

The CCG parser shows its output as the syntactic components of a sentence. The syntactic components are set of lexical entries containing syntactic categories which are often represented with symbolic notations. The syntactic categories thus describe

valency and directionality that defines semantic interpretation of the sentence. The following figure shows example of grammatical tree derivations using CCG.

Sentence: *China proposed regulations on Tobacco.*

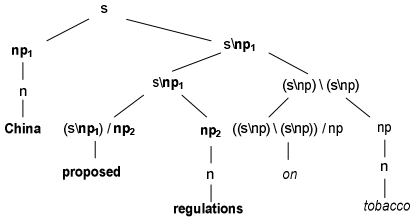


Fig. 1. Grammatical tree derivation using CCG notations

In CCG, there are *simple* and *comprehensive* categories. For simple categories, a sentence is denoted as *S*, noun is denoted as *N*, noun phrase is denoted as *NP*, and prepositional phase is denoted as *PP*. Similarly, comprehensive categories are formed by combining simple categories together using set of grammatical rules [18]. The type of category and the directionalities of its arguments are specified using slashes (e.g. \ or /). The deep understanding of grammatical transformation of sentences using CCG is beyond the scope of this paper as it has been thoroughly discoursed in existing literatures [18-19]. For the purpose of constructing grammatical tree derivations for sentences, we use a log-linear CCG parsing model by Clark and Curran [19]. The parsing model is described in [20] as part of the popular C&C tools[4] and it is freely available for research purposes.

3.3 Predicate-Argument Relations

Predicate-argument relations can be defined in terms of the argument slots in lexical categories that are shown by the grammatical tree derivations of the CCG parser. Previous research works show that predicate-argument relations can be recovered from grammatical tree derivations with F-measure accuracy of about 83% labeled recovery and 91% unlabeled recovery. The predicate-argument relation essentially shows the local and long-range word-word dependencies structures for a sentence. For example, a transitive verb category denoted as *(S\NP)/NP*, contains two predicate-argument relations. One is the object NP argument and the other is the subject NP argument. To differentiate between argument slots, the arguments are indexed from left to right.

Indices can be used to number the argument slots of the categories. For example, the transitive verb can be represented as $(S\backslash NP_1)/NP_2$. Here, the predicate-argument semantic structure can be presented as tuples $\langle w_H, c_{LH}, i_{args}, w_i \rangle$, where w_H is the *head word*, c_{LH} is the *lexical category* of the head word w_H, i_{args} is the *index* of the argument slot, and w_i is the head word that shows the *dependency relation* of c_{LH}. A similar illustration for representing predicate-argument semantic structure as tuples can be found in [6,19].

[4] http://svn.ask.it.usyd.edu.au/trac/candc/wiki

Table 1. Predicate-argument structure for the sentence "*China proposed regulations on Tobacco"*

w_H	c_{LH}	i_{args}	w_i
proposed	$(S\backslash \mathbf{NP_1})/NP_2$	1	China
proposed	$(S\backslash NP_1)/\mathbf{NP_2}$	2	regulations
on	$((S\backslash NP_1)\backslash(\mathbf{S\backslash NP})_2)/NP_3$	2	proposed
on	$((S\backslash NP_1)\backslash(S\backslash NP)_2)/\mathbf{NP_3}$	3	Tobacco

Table 2. Predicate-argument structure for the sentence "*Regulations that China proposed"*

w_H	c_{LH}	i_{args}	w_i
proposed	$(S\backslash \mathbf{NP_1})/NP_2$	1	China
proposed	$(S\backslash NP_1)/\mathbf{NP_2}$	2	regulations
that	$((S\backslash NP_1)\backslash(\mathbf{S\backslash NP})_2)$	2	proposed

Table 1 and Table 2 above show the tuple representation of predicate-argument semantic structures for two assumed sentences respectively. Both sentences have the same predicate-argument relation between *China* and *proposed* and *regulations* and *proposed*. Intuitively, in both sentences, the subject *China* holds index 1of the argument slots of the predicate *proposed,* that is, NP_1 in $(S\backslash NP_1)/NP_2$, and the object *regulations* holds index 2 of the argument slots of the predicate *proposed,* that is, NP_2 in $(S\backslash NP_1)/NP_2$. To the best of our knowledge, available predicate-argument extractor[5] does not support CCG parser. Thus, for the purpose of our evaluations and experiment, we use the Boxer[6] component of the popular C&C tools [20], to represent predicate-argument semantic structure from the CCG grammatical tree derivation output. The output of the Boxer component is arguably comparable to the output of a predicate-argument extractor.

3.4 The Subjective Component Score

The idea of opinion retrieval must be centered on retrieving sentences that are subjectively relevant to the query topic. Therefore, it is very important to differentiate between subjective and objective sentences such that subjective sentences get higher weights compared to objective sentences. However, given the many ways subjectivity can be expressed, it is often difficult to efficiently identify subjective sentences from opinionated documents. Subjectivity annotation is well discussed in [21] and [22]. In this paper, we adopted a popular subjective sentence annotation scheme[7] provided by Wiebe et al. [22] to compensate subjective sentences with higher weights during our final scoring process. The subjective annotation scheme annotates sentences by showing respective attributes values that are scaled according to the opinion and polarity components of the sentence.

Since we can only identify subjectivity from the original natural language format of each sentence, we perform the subjectivity identification process before the parsing

[5] http://www.semanticsoftware.info/pax
[6] http://svn.ask.it.usyd.edu.au/trac/candc/wiki/boxer
[7] http://www.cs.pitt.edu/~wiebe/resources.html

of sentences and the derivation of the predicate-argument structures. We then tag the predicate-argument structures of equivalent subjective sentences with a string constant "*subj*". The subjectivity of each sentence is determined by using the annotation output as follows:

INSIDE(text: "Alice love Chocolate"; source: writer/Ryan);
DIRECT-SUBJECTIVE(text: "love"; intensity: high; expression – intensity: high; polarity: positive; insubstantial: false;source: writer/Alice; target: Chocolate);

Thus, a sentence is considered subjective if its annotation output contains a "direct-subjective" or "expressive-subjectivity" annotation component. Elements of the subjective component can also be used to determine the degree of subjectivity as required. Given a subjective annotation component, the subjectivity score $score_{subj}$ of a sentence is computed as follows:

$$score_{subj} = (annot_{comp} \ni v_{subj}) = 0.5 \quad (1)$$

where $annot_{comp}$ is the annotation output, v_{subj} denotes the presence of a subjective component, and the constant 0.5 can vary empirically while 0 is assigned to a non-subjective sentence. We will make use of the subjective score in equation 8.

3.5 Context-Dependent Opinion Relevance

Having derived predicate-argument structures that represent underlying meaning of sentences, we propose to identify subjective predicate-argument structures that shares similar context. Given that $D = \{s_1, \dots, s_m\}$ is a set of well formed sentences in document *d*, the opinion relevance function *f* is used to estimate the contextual relevance score between a given natural language query q and each well formed sentence *s,* i.e., $f(q, s)$. This denotes how well a given query is contextually similar to each sentence in a given document *d.* Thus, using the predicate-argument relation between the CCG output parse trees of q and *s*, it is relatively straightforward for us to find similarities between the underlying syntactic structures of q and *s* as shown by their respective predicate-argument structures.

Given that $Q_{predargs} = \{t_1, \dots, t_m\}$ is a set of predicate-argument terms derived from q, $S_{predargs} = \{w_1, \dots, w_m\}$ is a set of predicate-argument terms derived from *s*. Since the contextual similarity between q and *s* can be shown using the predicate-argument *relation*, for each *s*, $f(q, s)$ can either be 1 (denoted as the relevance of q to *s*) or 0 (denoted as q not relevant to *s*). However, it is often difficult to determine under what condition can $f(q, s)$ be 1 or 0. One straightforward way is to model similarity between each term in $Q_{predargs}$ and $S_{predargs}$. This can be achieved by using topic model such as LSA [23] and PLSA [24] to measure term co-occurrences (*t, s*) between $Q_{predargs}$ and $S_{predargs}$. We use PLSA because of its advantages over the conventional LSA [24].

$$rel(s, t) = P(s)P(t|s), P(t|s) = \sum_{z \in Z} P(t|z)P(z|s) \quad (2)$$

where *rel* is the *relation* score between $Q_{predargs}$ and $S_{predargs}$, *s* denotes the derived predicate-argument *structure* for sentences, *t* denotes the *terms* in the derived predicate-argument *structure,* and z is an un-observed class variable which acts as a

bottleneck variable in predicting terms for each observation. Intuitively, similar predicate-argument structures would show a *relation* score that tends towards 1. However, this process suffered a particular set back. We observed direct *word overlap* between q and s since the *relation* score ranges between 0 and 1. For example, a *relation* score of 0.85 does not necessarily indicate relevance. A single dissimilarity among terms in the predicate-argument structures may change the contextual meaning of the concerned sentence. Consider, for example q and s, in the following *word overlap* problem as shown by their respective predicate-argument structures.

$\boldsymbol{q}$: Amazing clothes are designed for Oscar awards.
$\boldsymbol{Q_{predargs}}$: designed\amazing/clothes/Oscar/awards
$\boldsymbol{s_1}$: The amazing fashion show at Oscar awards.
$\boldsymbol{S1_{predargs}}$: show\amazing/fashion/Oscar/awards

Above, the predicate-argument structure for query q (i.e. $Q_{predargs}$) has *word overlap* (i.e. *designed/show*) with the predicate-argument structure for s_1 (i.e. $S1_{predargs}$). Although, the two predicate-argument structures share the same words like "amazing" and "awards", q and s_1 turned out to have different semantic meaning. Thus, we seek a more intuitive method that can help model opinion relevance such that the relations score of $Q_{predargs}$ to $S_{predargs}$ can be equal to 1.

3.6 Constructing a Relevance Function

Having faced with *word overlap problem*, we propose to utilize a straightforward approach to construct an intuitive relevant function. Thus, for each structure pair of $Q_{predargs}$ and $S_{predargs}$, we propose a *transformed term-term similarity matrix* on which Jaccard Similarity Coefficient value can be calculated for each pair. We chose Jaccard Similarity Coefficient since it is largely useful to measure word overlaps that exist between the attributes of paired sets. It is also efficient for sparse text such as commonly observed in terms within predicate-argument structures. Ideally, the best Jaccard Similarity Coefficient is higher towards 1. For each pair structures, we then considered *linear combinations* of Jaccard Similarity Coefficients and the relevance scores derived from a *relevance model*. The chosen relevance model was proposed by Lavrenko and Croft [4], and it has shown substantial performance for effectively estimating the probabilities of words in relevant class using the query alone [25]. Our main purpose is to use the relevance model to compensate for possible non-squared relationship between $Q_{predargs}$ and $S_{predargs}$.

3.7 Transformed Terms Similarity (TTS)

Since we can no longer apply PLSA because of its word overlap problem, our model may be prone to implicit relevance problem resulting from *synonymy* and *polysemy*. These problems are without doubt solved in LSA and PLSA (combined with other topic models) respectively. We understand that two terms with the same implicit meaning (e.g. *clothes and dresses)* can exist among $Q_{predargs}$ and $S_{predargs}$. Thus, we propose a straightforward transformation mechanism that uses *synonyms* and

hyponyms of individual terms in both $Q_{predargs}$ and $S_{predargs}$. Arguably, we use the method to determine implicit similarity between terms, such that two paired terms i and j have implicit meaning if either term is the synonym or hyponym of the other. For this purpose, we form a *similarity matrix* distribution over terms to measure implicit relationship between $Q_{predargs}$ and $S_{predargs}$. This same methodology has been used in LSA commonly regarded as LSA term-term matrix.

Table 3. Term-term similarity matrix with word overlap problem

		$Q_{predargs}$				
		~~**designed**~~	amazing	clothes	Oscar	awards
$S_{predargs}$	~~**show**~~	**0**	0	0	0	0
	amazing	0	**1**	0	0	0
	fashion	0	0	**1**	0	0
	Oscar	0	0	0	**1**	1
	awards	0	0	0	1	**1**

Given that P is a set of terms in each predicate-argument structure such that each term $\overrightarrow{p_i} \in R^m (i = 1, \dots n)$ is a vector of dimension m. $S \in R^{n\,X\,n}$ is a similarity matrix where each $sim_{ij} (0 \leq s_{ij} \leq 1)$ is the similarity between $\overrightarrow{p_i}$ and $\overrightarrow{p_j}$. The similarity sim_{ij} is calculated as the absolute observation of $i \equiv j$, or the presence of i in sets j_{sym} or j_{hym}, where j_{sym} and j_{hym} denotes the synonyms and hyponyms of term j respectively.

In order to calculate the Jaccard Similarity Coefficient, we use the Binary Independent Model (BIM) [26] to construct the term-term similarity matrix. The BIM method avoids using frequencies of terms but takes each term as binary vector over the vocabulary space of j_{sym} and j_{hym}. That is, sim_{ij} shows the similarity between term i and j as a binary attribute 1 (denoted as term i is similar to j) if i is j or i is present in j_{sym} or i is present in j_{hym}. Conversely, sim_{ij} can also show dissimilarity between term i and j as a binary attribute 0 (denoted as term i not similar to j) if i is not j or i is not present in j_{sym} or i is not present in j_{hym}. Synonyms in j_{sym} and hyponyms in j_{hym} were derived using Wiktionary [27]. More importantly, the binary attributes help in calculating the Jaccard Similarity Coefficient for the term-term similarity matrix which can only contain asymmetric binary attributes of each pair predicate-argument structures.

Let $S_{predargs} = (i_1, \dots, i_n)$ and $Q_{predargs} = (j_1, \dots, j_m)$ be the pair predicate-argument structures respectively. Thus, using BIM, we calculate sim_{ij} as the probability of term incidence vectors $P(R|\vec{i}, \vec{j})$.

$$P(R = 1|\vec{i}, \vec{j}) = \frac{P(\vec{i}\,|R=1, \vec{j})P(R=1|\vec{j})}{P(\vec{i}|\vec{j})} \tag{3}$$

$$P(R = 0|\vec{i}, \vec{j}) = \frac{P(\vec{i}\,|R=0, \vec{j})P(R=0|\vec{j})}{P(\vec{i}|\vec{j})} \tag{4}$$

In the above, $P(\vec{i}\,|R = 1, \vec{j})$ in equation 3 and $P(\vec{i}\,|R = 0, \vec{j})$ in equation 4 are the probabilities that term i is j and term i is not j respectively. However, since the actual probabilities are not known, the prior probability of $\vec{i}$ is $\vec{j}$ or $\vec{i}$ is not $\vec{j}$ is defined as

$P(R = 1|\vec{j})$ and $P(R = 0|\vec{j})$ respectively. Since $\vec{i}$ can either be present or not present in $Q_{predargs}$, then:

$$P(R = 1|\vec{i},\vec{j}) + P(R = 0|\vec{i},\vec{j}) = 1 \quad (5)$$

Consider, for example, a term-term similarity matrix with asymmetric binary attributes for the following:

$\boldsymbol{q}$: Amazing clothes are designed for Oscar awards.
$\boldsymbol{Q_{predargs}}$: designed\amazing/clothes/Oscar/awards
$\boldsymbol{s_1}$: The amazing clothes were made for Oscar awards.
$\boldsymbol{S1_{predargs}}$: made\amazing/clothes/Oscar/awards

Table 4. Term-term similarity matrix showing semantic relevance for a subjective sentence

		$Q_{predargs}$				
		designed	amazing	clothes	Oscar	awards
$S_{predargs}$	made	**1**	0	0	0	0
	amazing	0	**1**	0	0	0
	clothes	0	0	**1**	0	0
	Oscar	0	0	0	**1**	1
	awards	0	0	0	1	**1**

The term-term similarity matrix shown in Table 4 is constructed for a sample $Q_{predargs}$ and $S_{predargs}$ by using BIM to compute the binary attributes. It is therefore straightforward to compute the Jaccard Similarity Coefficient for the term-term similarity matrix. Intuitively, the best Jaccard Similarity Coefficient would be 1 provided all diagonal elements of the similarity matrix is filled with 1 (i.e. all ij elements must be identical to all ji elements).

$$Jaccard - Coefficient = \frac{M_{11}}{M_{01}+M_{10}+M_{11}} \quad (6)$$

where M_{11} denotes the total number of attributes where i and j both have a value of 1, M_{01} denotes the total number of attributes where the attribute of i is 0 and the attribute of j is 1, M_{10} denotes the total number of attributes where the attribute of i is 1 and the attribute of j is 0. Using equation 6, the Jaccard Similarity Coefficient for table 4 is equal to 1, which means query q and sentence s is semantically relevant to each other unlike Table 3 which has Jaccard Similarity Coefficient of 0 (i.e. all ij elements are not identical to all ji elements).

3.8 A Linear Relevance Model

The Transformed Term Similarity (TTS) approach that is described above is efficient provided $Q_{predargs}$ and $S_{predargs}$ are squared. That is, $Q_{predargs}$ and $S_{predargs}$ must have the same number of rows and columns. This is based on the fundamental principle of term-term similarity matrix which we adopted. However, in some cases, $Q_{predargs}$ and $S_{predargs}$ may not have the same number of rows and columns since there may be variations in the number of terms in each predicate-argument structure. For example, a given query may have just three terms in its predicate-argument

structure, while another sentence with much longer grammatical dependencies may have five or more terms in its predicate-argument structure. Thus, we use the relevance model proposed by Lavrenko and Croft [4], to estimate $P(\vec{\imath} \,|Q_{predargs} \cap S_{predargs})$. Again, this has shown outstanding performance in many information retrieval systems [25].

The relevance model $P(\vec{\imath} \,|Q_{predargs} \cap S_{predargs})$ is estimated in terms of the joint probability of observing a vector $\vec{\imath}$ together with vectors from $Q_{predargs} \cap S_{predargs}$ [4]. Consider, for example, the following scenario:

$\boldsymbol{q}$: important regulations that China proposed
$\boldsymbol{Q_{predargs}}$: proposed\important/regulations/China
$\boldsymbol{s_1}$: China proposed some important regulations on tobacco.
$\boldsymbol{S1_{predargs}}$: proposed\important/regulations/China/tobacco
$\boldsymbol{s_2}$: The proposed regulation on tobacco is a new and important development in China.
$\boldsymbol{S2_{predargs}}$: proposed\important/regulation/China/tobacco/development

Above, the predicate-argument structures $Q_{predargs}$, $S1_{predargs}$, and $S2_{predargs}$ are not squared. Each predicate-argument structure has different number of vectors. Thus, to solve this problem, we can make a *pairwise* independence assumption that $\vec{\imath}$ and all terms from $Q_{predargs} \cap S_{predargs}$ are sampled independently and identically to each other. By this way, the relevance model would be appropriate to determine when the predicate-argument structure $Q_{predargs}$ is independently present as part of the predicate-argument structure of some long range sentences such as shown in $S1_{predargs}$ and $S2_{predargs}$ above. The relevance model is computed as follows:

$$P(\vec{\imath}, \vec{j}_1 \ldots \vec{j}_k) = P(\vec{\imath}) \prod_{\mathrm{i}=1}^{\mathrm{k}} \sum_{M_\mathrm{i} \in Z} P(M_\mathrm{i}|\, \vec{\imath})\, P(\vec{j}_1|M_\mathrm{i}) \tag{7}$$

where M_i is a unigram distribution from which $\vec{\imath}$ and $\vec{j}$ are sampled identically and independently, Z denotes a universal set of unigram distributions according to [4].

Thus, for each blog document, our aim is to do a linear combination of the result of TTS in equation 6, the result of the relevant model in equation 7, and the absolute sum of the subjectivity score in equation 1. This offers a complete solution to compensate for squared and non-squared relationships between similar and subjective predicate-argument structures. By doing a linear combination, the predicate-argument structure of a given query can be checked against predicate-argument structures of subjective sentences with either short-range or long-range grammatical dependencies. While the subjectivity score ensures a document is subjective, we believe the TTS model (for squared similarity) and the Relevance Model (for non-squared similarity) would solve the short-range and long-range sentence dependencies problem efficiently and respectively. The *linear relevance scoring function* is computed as follows:

$$R_{\mathcal{L}}(q,d) = \frac{k}{N} \sum_{i=1}^{k} TTS_{score} \; + \; \frac{v}{N} \sum_{i=1}^{v} RM_{score} \; + \; \sum_{i=1}^{k+v} Score_{subj} \tag{8}$$

where $R_{\mathcal{L}}(q,d)$ is the linear relevance function that takes input as query q and document d satisfying a linear combination expression $C = aX + bY + Z$, where a and b can be empirical constants, and X, Y, and Z are the values to be linearly

combined. k is the number of sentences with short-range dependencies that satisfy TTS, N is the total number of sentences in document d, TTS_{score} is derived using equation 6, v is the number of sentences with long-range dependencies that satisfy RM_{score}, and RM_{score} is derived using equation 7.

4 Evaluation and Experiment

We have derived a linear relevance model (equation 8) that can retrieve context-dependent opinion using semantically related predicate-argument structures and a sentence-level subjective component. In this section we will evaluate the linear relevance model against TREC Blog 08 standard baselines using different evaluation measures. We will also describe the overview of the evaluations to be performed.

First, we retrieved 50,000 TREC Blog 08 documents from permalinks by developing a heuristic rule-based algorithm (BlogTEX)[8] to retrieve only English blog documents. Documents written in foreign languages are removed by comparing English function words with each blog document. For accuracy, we set frequency threshold for English function words within a blog document. Blog documents that have less than the frequency threshold are considered non-English and discarded. In order to prepare each blog document for sentence-level syntactic parsing, we use *LingPipe*[9] *Sentence Model* to identify sentence boundaries in blog documents. We modified the sentence model to detect omitted sentences and sentences without boundary punctuation marks.

Second, for each blog document, we transform each sentence to its equivalent syntactic parse tree using the log-linear CCG parsing model described in [20] as part of the popular C&C tools[10].

We extracted the title and description fields of the TREC 2008 new query topics *without* stemming or stop words removal. Description fields of 50 query topics (TREC 1001-1050) are transformed to their equivalent syntactic parse trees. 49 queries and approximately 50,000 blog documents were successfully transformed to their equivalent syntactic parse trees.

Third, in section 4.2, we perform evaluation based on cross-entropy by comparing the cross-entropy between an "ideal" relevance model (based on human judgments), our proposed *linear relevance model*, and one of the TREC Blog 08 best runs that also used title and description fields for opinion finding task (KLEDocOpinTD)[28]. The idea of using cross-entropy is to observe the best approximation model that has minimum cut in terms of cross-entropy by simply measuring the uncertainty of query topics shared by a given collection of blog documents.

Finally, to further show the performance of our opinion retrieval model in a more practical scenario, we perform experiment for opinion retrieval task using TREC Blog 08 dataset. The evaluation metrics used are based on Mean Average Precision (MAP), R-Precision (R-prec), and precision at ten (P@10). All the 50 TREC topics were used and more than 20 topics received significant improvement in terms of MAP.

---

[8] http://sourceforge.net/projects/blogtex

[9] http://alias-i.com/lingpipe/demos/tutorial/sentences/read-me.html

[10] http://svn.ask.it.usyd.edu.au/trac/candc/wiki

4.1 Evaluation Using Cross-Entropy

The cross-entropy method or Kullback-Leibler divergence uses an information-theoretic approach that measures the distance between two distributions. The cross-entropy is measured in bits, and a minimized cross-entropy indicates improved performance. We plot the graph of the measured cross-entropy as a function of the number of top-ranked blog documents, first from the "ideal" relevant documents, and then from the retrieved documents by the proposed linear relevance model and Blog 08's KLEDocOpinTD. For the ideal relevant documents, we manually labeled 100 documents to represent the relevant results of the ideal model.

Thereafter, for each approximation model, we limit the number of retrieved top-ranked documents from 1000 to 100 since we are more concerned about context-dependent opinionated documents at affordable browsing level. We then compute their respective cross-entropy or KL-divergence by measuring the uncertainty of TREC query topics shared by the top-ranked document collection. Note that we considered the synonyms and hyponyms of terms in comparing predicate-argument structures in our TTS technique (section 3.6.1). Thus, we ensured the frequency count for calculating the probabilities for cross-entropy also include either synonyms or hyponyms of terms in query topics and the retrieved blog documents.

$$CE = \sum_{w \in V} P(t|Q) \log \frac{P(t|Q)}{P_c(t)} \tag{9}$$

where V is the entire vocabulary of the top-ranked collection, $P(t|Q)$ is the relative frequency of term t in each TREC query topic, and $P_c(t)$ is the relative frequency of term t in the entire top-ranked collection. Note that since $P_c(t)$ has a fixed reference distribution, it perfectly makes sense to refer to cross-entropy and KL-divergence as identical in our case. Both cross-entropy and KL-divergence would have minimal values when $P(t|Q) = P_c(t)$.

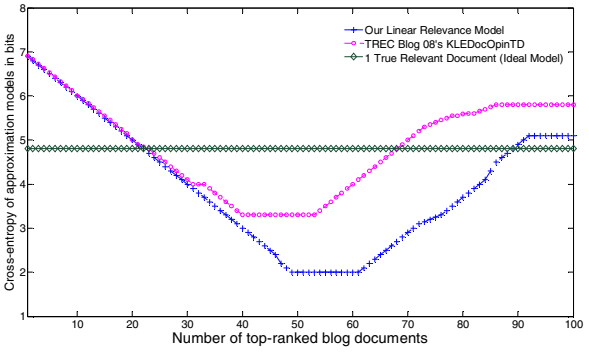


Fig. 2. Minimum cross-entropy between three approximation models and one ideal topic model

With the cross-entropy of a true relevant document shown as the horizontal line marked with "Δ", the two approximation models show minimum cross-entropy between 35 and 60 documents. Initially, the models compete between the top10 documents. This implies that all the models have the tendencies to retrieve relevant opinionated documents but with slight performance variations at top-ranked

documents. The variations can also be observed upon calculating the Precision@K and the MAP. However, the linear relevance model has more pronounced cross-entropy values at the top-ranked documents. Moreover, it shows the absolute minimum cross-entropy compared to other approximation models. It could also be observed that the cross-entropy of the linear relevance model flattens relatively slower than the other model at around 70 to 90 documents. Obviously, this is an indication that the linear relevance model approximation has more weight and robust performance than the other approximation model.

The evaluation shown above is query-dependent but it is sufficient enough to show the general relationship shared by a blog document collection towards a given query topic. However, to further demonstrate the effectiveness of our opinion retrieval model, we need to show *relevance clarity* C_R which describes a *strong relevance relationship* between the blog documents retrieved by the two approximation models such as used in [29]. Thus, using a document-dependent evaluation, we can assume that $P(t|Q)$ and $P(t|D)$ are the estimated query and document distributions respectively (assuming $P(t|D)$ has a linearly smoothed distribution). The relevance value of document D with respect to query Q can then be measured as the *negative* KL-divergence between the approximation models.

$$C_R = \sum_{w \in V} P(t|Q) \log P(t|D) + (-\sum_{w \in V} P(t|Q) \log P(t|Q)) \tag{10}$$

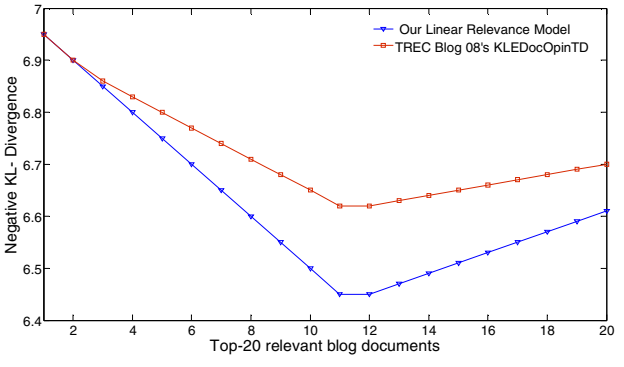


Fig. 3. Negative KL-Divergence between two approximation models. Proposed model shows robust performance at top-20 documents compared to the other model.

In order to show a strong relevance relationship among the approximation models, we select the top-20 relevant results of each model and compute their respective negative KL-divergence to indicate *relevance clarity*. The idea is to show if each retrieved document (as per the rank) contains the same opinion target across the two models. Interestingly, the evaluation shows the top-2 documents are likely to contain the same opinion target. It also shows that our linear relevance model has more relevant documents at top-ranked. Although the proposed linear relevance model has a more desired result compared to the other model, we learned that document-dependent evaluation using negative KL-divergence gives more consistent results.

4.2 Opinion Retrieval Task

To complement our cross-entropy or KL-divergence evaluations, we perform experiments on TREC Blog 08 and compared our results with Blog 08 best run and Blog 08's KLEDocOpinTD. Inspired by the evaluation results shown in Figure 3, we now focus our attention to produce context-dependent opinionated documents with high precision at top-ranked documents. For this purpose, we use a re-ranking technique by initially retrieving top 20 documents using BM25 popular ranking model. We use empirical parameters k=1.2 and b=0.75 for BM25 as they have been reported to give acceptable retrieval results [15]. These top 20 documents are then re-ranked by our model. We use the general evaluation metrics for IR measures which include MAP, R-Prec, and P@10. A comparison of opinion MAPs with the increased number of top-K blog documents is shown in Figure 4. Our linear relevance model shows improved performance over Blog 08 best run by more than 15% and significantly outperforms Blog 08's KLEDocOpinTD. Performance improvement in precision and recall curves upon the re-ranking technique is shown in Figure 5. The Linear Relevance Model show improved performance with our re-ranking technique. The result also complements our cross-entropy evaluation with persistent improvements in the same order. This shows the effectiveness of our model, hence suggests the importance of re-ranking technique. Our model also shows substantial performance without any re-ranking mechanism (i.e. when BM25 is not used at all). In Table 5 below, the best significant improvements over Blog 08 best run is indicated with *. Again, in terms of MAP, our model has significant performance more than Blog 08 best run and Blog 08's KLEDocOpinTD. Our model also shows significant improvement in terms of R-prec in all cases. This shows the effectiveness of our model in terms of retrieving context-dependent relevant opinions from blogs.

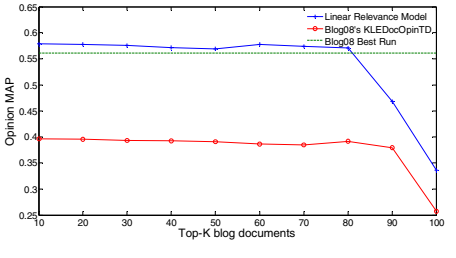


Fig. 4. MAP comparison for different models at Top-K documents

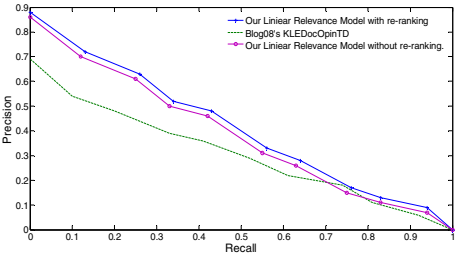


Fig. 5. Precision and Recall curves with and without re-ranking mechanism

Table 5. Performance and improvements on Opinion Retrieval task

Model	**MAP**	**R-Prec**	**P@10**
Best run at Blog 08	0.5610	0.6140	0.8980
Linear Relevance Model re-ranked	**0.6485** *	**0.7324** *	**0.9231**
Improvement over Best run at Blog 08	15.6%	19.3%	2.8%
Linear Relevance Model without re-ranking	**0.6369** *	**0.6894** *	**0.9317**
Improvement over Best run at Blog 08	13.5%	12.3%	3.8%
Blog 08's KLEDocOpinTD	0.3937	0.4231	0.6273
Improvement with Linear Relevance Model re-ranked	64.7%	73.1%	47.2%

5 Conclusions

We proposed context-dependent opinion retrieval by comparing semantically related predicate-argument structures of natural language queries and well formed subjective sentences. It is remarkable that our model outperforms the baselines in the results shown above. Specifically, our model outperforms Blog08 best run. With our re-ranking mechanism, we could observe that relevant documents to not only contain opinionated query words but opinionated sentences that lexically combine synonyms and hyponyms of query words that are largely used to refer to an opinion target. This implies that frequency or proximity of opinonated query words may not be helpful to context-dependent opinion retrieval. In future, we will integrate multi-sentence opinion dependencies into our model, whereby the subjectivity of a particular sentence may depend on other preceding sentences in the same documents. This could be intuitive for facet-based opinion retrieval. We will also compare the performance of our linear technique to other non-linear techniques. It will also worth identifying and studying the impact of ambiguous sentences (e.g. irony, sarcasm idioms, and metaphor) on opinion retrieval by using a grammar-based approach.

References

1. Zhang, M., Ye, X.: A generation model to unify topic relevance and lexicon-based sentiment for opinion retrieval. In: Proc. of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Singapore (2008)
2. Lee, S.-W., Lee, J.-T., Song, Y.-I., Rim, H.-C.: High precision opinion retrieval using sentiment-relevance flows. In: Proc. of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, Switzerland (2010)
3. He, B., Macdonald, C., He, J., Ounis, I.: An effective statistical approach to blog post opinion retrieval. In: Proc. of the 17th ACM Conference on Information and Knowledge Management, USA (2008)
4. Lavrenko, V., Croft, W.B.: Relevance based language models. In: Proc. of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, USA (2001)
5. Macdonald, C., Santos, R.L.T., Ounis, I., Soboroff, I.: Blog track research at TREC. In: SIGIR Forum pp. 58-75 (2010)
6. Gildea, D., Hockenmaier, J.: Identifying semantic roles using Combinatory Categorial Grammar. In: Proc. of the 2003 Conference on Empirical Methods in Natural Language Processing, vol. 10 (2003)
7. Kanayama, H., Nasukawa, T.: Fully automatic lexicon expansion for domain-oriented sentiment analysis. In: Proc. of the 2006 Conference on Empirical Methods in Natural Language Processing, Australia (2006)
8. Surdeanu, M., Harabagiu, S., Williams, J., Aarseth, P.: Using predicate-argument structures for information extraction. In: Proc. of the 41st Annual Meeting on Association for Computational Linguistics, Japan, vol. 1 (2003)
9. Stefan Siersdorfer, S.C., Pedro, J.S.: How Useful are Your Comments? Analyzing and Predicting You Tube Comments and Comment Ratings. In: International World Wide Web Conference, USA (2010)

10. Luís Sarmento, P.C., Silva, M.J., de Oliveira, E.: Automatic creation of a reference corpus for political opinion mining in user-generated content. Paper presented at the Proceeding of the 1st International CIKM Workshop on Topic-Sentiment Analysis for Mass Opinion, China (2009)
11. Pang, B., Lee, L.: Opinion Mining and Sentiment Analysis. Found Trends Inf. Retr. 2, 1–135 (2008)
12. Ding, X., Liu, B., Yu, P.S.: A holistic lexicon-based approach to opinion mining. In: Proc. of the International Conference on Web Search and Web Data Mining, USA (2008)
13. Hiemstra, D.: Using language models for information retrieval. PhD Thesis, Centre for Telematics and Information Technology, The Netherlands (2000)
14. Amati, G., Amodeo, G., Bianchi, M., Gaibisso, C., Gambosi, G.: A Uniform Theoretic Approach to Opinion and Information Retrieval. SCI, pp. 83–108. Springer, Heidelberg (2010)
15. Robertson, S., Zaragoza, H.: The Probabilistic Relevance Framework: BM25 and Beyond. Foundations and Trends in Information Retrieval 3, 333–389 (2009)
16. Gerani, S., Carman, M.J., Crestani, F.: Proximity-Based Opinion Retrieval. SIGIR ACM, 978 (2010)
17. Javanmardi, S., Gao, J., Wang, K.: Optimizing two stage bigram language models for IR. In: Proc. of the 19th International Conference on World Wide Web, USA (2010)
18. Steedman, M.: The Syntactic Process (Language, Speech, and Communication). The MIT Press (2000)
19. Clark, S., Curran, J.R.: Wide-coverage efficient statistical parsing with ccg and log-linear models. Comput Linguist 33, 493–552 (2007)
20. Curran, J.R., Clark, S., Bos, J.: Linguistically motivated large-scale NLP with C & C and boxer. In: Proc. of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions, Czech Republic (2007)
21. Esuli, A.: Automatic generation of lexical resources for opinion mining: models, algorithms and applications. SIGIR Forum 42, 105–106 (2008)
22. Wiebe, J., Wilson, T., Cardie, C.: Annotating Expressions of Opinions and Emotions in Language. Language Resources and Evaluation 39, 165–210 (2005)
23. Dumais, S.T., Furnas, G.W., Landauer, T.K., Deerwester, S., Harshman, R.: Using latent semantic analysis to improve access to textual information. In: Proc. of the SIGCHI Conference on Human Factors in Computing Systems, USA (1988)
24. Hofmann, T.: Probabilistic latent semantic indexing. In: Proc. of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, USA (1999)
25. Lv, Y., Zhai, C.: A comparative study of methods for estimating query language models with pseudo feedback. In: Proc. of the 18th ACM Conference on Information and Knowledge Management, China (2009)
26. Rijsbergen, C.J.V.: A Theoretical Basis for the use of Co-Occurrence Data in Information Retrieval. Journal of Documentation 33, 106–119 (1977)
27. Müller, C., Gurevych, I.: Using Wikipedia and Wiktionary in Domain-Specific Information Retrieval. In: Peters, C., Deselaers, T., Ferro, N., Gonzalo, J., Jones, G.J.F., Kurimo, M., Mandl, T., Peñas, A., Petras, V. (eds.) CLEF 2008. LNCS, vol. 5706, pp. 219–226. Springer, Heidelberg (2009)
28. Yeha Lee, S.-H.N., Kim, J., Nam, S.-H., Jung, H.-Y., Lee, J.-H.: KLE at TREC 2008 Blog Track: Blog Post and Feed Retrieval. In: TREC (2008)
29. Huang, X., Croft, W.B.: A unified relevance model for opinion retrieval. In: Proc. of the 18th ACM Conference on Information and Knowledge Management, China (2009)

XML Document Clustering Using Structure-Preserving Flat Representation of XML Content and Structure

Fedja Hadzic[1], Michael Hecker[1], and Andrea Tagarelli[2]

[1] Digital Ecosystems and Business Intelligence Institute, Curtin University, Australia
{fedja.hadzic,michael.hecker}@curtin.edu.au
[2] Dept. of Electronics, Computer and Systems Sciences, University of Calabria, Italy
tagarelli@deis.unical.it

Abstract. With the increasing use of XML in many domains, XML document clustering has been a central research topic in semistructured data management and mining. Due to the semistructured nature of XML data, the clustering problem becomes particularly challenging, mainly because structural similarity measures specifically designed to deal with tree/graph-shaped data can be quite expensive. Specialized clustering techniques are being developed to account for this difficulty, however most of them still assume that XML documents are represented using a semistructured data model. In this paper we take a simpler approach whereby XML structural aspects are extracted from the documents to generate a flat data format to which well-established clustering methods can be directly applied. Hence, the expensive process of tree/graph data mining is avoided, while the structural properties are still preserved. Our experimental evaluation using a number of real world datasets and comparing with existing structural clustering methods, has demonstrated the significance of our approach.

1 Introduction

XML has become extremely popular in management and mining of semistructured/hierarchical text data due to its abilities in representing information in a well-defined, extensible and machine readable format. This is evidenced by the existence of many domain-specific XML based markup languages [17].

Clustering of XML documents has important applications in many domains that need management and processing of real-life complex objects that are represented as semistructured data. It is a more challenging task than the standard data clustering problem since the structural aspects in the data need to be taken into account. Defining a distance/similarity function over semistructured data that can be effectively and efficiently utilized for the XML clustering problem is known to be a difficult research problem. Initially proposed techniques were based on tree edit distance [13,2]. Since calculating tree edit distance is known to be a computationally expensive problem [2], different approaches for measuring structural similarity have been proposed. For example, in [8] structural

J. Tang et al. (Eds.): ADMA 2011, Part II, LNAI 7121, pp. 403–416, 2011.

information of XML documents is represented in form of paths contained in the underlying tree model on which specialized similarity measures are defined. The S-GRACE approach [12] converts an XML document into a structure graph and the similarity is based on the number of common element-subelement relationships. In [3], the XRep method is developed focusing on a notion of structural XML cluster representative. This representative is built to capture the most relevant structural features of the documents within a cluster, and it is used to assign XML documents to structurally homogeneous groups in a cluster hierarchy. Another tree-based, summary-aware framework for clustering XML documents by structure is described in [4]. XML documents represented as rooted ordered labeled trees are individually summarized to reduce nesting and repetition of elements which are compared using a tree edit distance based on a variant of the Chawathe's algorithm [2]. The clustering scheme is based on a traditional graph-theoretic divisive approach. Given a collection of XML documents, a fully connected, weighted graph is built over the structural summaries of the input documents, where the edge weights correspond to the structural distance between summaries. A minimum spanning tree (MST) of the graph is computed, then the MST edges with a weight above a user-specified threshold are deleted. The connected components of the remaining graph are the single-link clusters.

More recently, one can witness an increasing number of XML document clustering approaches that take into account both the structure and content of the documents [18,5,10,16,11]. For instance, XProj [1] utilizes frequent substructures in a segment of data to measure the similarity, whereas HCX [10] determines the structural similarity based on frequent subtrees to extract (constrained) content and represent it in a Vector Space Model. Recently, the use of a Tensor Space Model to capture the content and structural information has been proposed in [11]. The structure features for the model are generated based on the length-constrained closed induced subtrees and these are used to constrain the content included in the model. In [16], subtrees that are cohesive according to the semantics of the original XML document trees are modeled into a transactional domain in which each item embeds a distinct combination of semantically-enriched structure and content XML features.

However, regardless of the techniques developed to use XML structural information at different refinement levels (i.e., node labels, edges, paths, twigs), the large majority of existing methods for structure-based XML clustering rely on tree/graph-shaped representation models. Consequently, the similarity/distance functions that drive the proximity search and detection during the clustering, are required to be suitable for the comparison of semistructured data, which is known to be a computationally difficult problem.

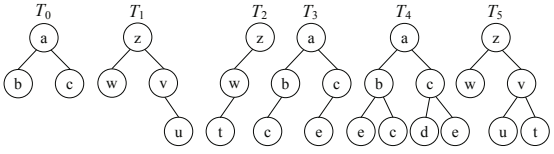
In this work we propose an alternative approach for XML document clustering, based on our recently proposed [6] structure preserving flat data representation for tree-structured data (henceforth referred to as FDT). The conversion process is based on the extraction of a document structure model within which each document instance can be matched to generate the flat data representation that captures the structural properties. Given such a representation,

well-established clustering methods originally designed for vectorial data can be directly applied. CLUTO, a well-known toolkit for efficiently clustering large, high-dimensional document collections [9,20] is taken as the case in point and is applied directly on the tree-structured data converted in the flat data format. The complexity of incorporating structural information in a distance/similarity measure is avoided, while the results can still be accompanied by the structural information captured by the document structure model. The implications of the approach are that the exact position of nodes within a general structure encompassing structural properties of all documents is taken into account during clustering. This is somewhat different from XML clustering methods based on subtree mining and matching, like XProj, which extract substructures that can occur anywhere within the documents structure of a data segment. In our approach, the user can specify a minimum frequency threshold during the document structure model generation, so that the general structure extracted only encompasses the frequently occurring structures among the documents being clustered. Furthermore, given that many more clustering techniques exist for flat data representation, the conversion approach in itself can potentially enable a wider range of techniques to be applied for the XML clustering problem.

XML content can be quite broad encompassing elements, values, attributes and attribute values. Although XML documents can be naturally modeled as trees, it is not clear how this information should be organized, and it may well be that it is dependent on the particular mining task as well as the application at hand. For example, in frequent subtree mining the values associated with elements are often assigned to a single node. This is both more efficient as we avoid building large trees, and practical because from the frequent subtrees one can discover interesting associations among the element values of an XML document from a single domain. On the other hand in XML document clustering, assigning the element name and value to a single node may not be desired, as one would like to detect similarity among element names, even when the values are different. This is even more so when the aim is to form clusters from heterogeneous collections of XML documents. In this work we also empirically study the impact of including/excluding partial content from XML documents for the clustering task. Our experimental evaluation was performed using several sets of XML documents from a variety of domains. The comparisons with existing XML clustering methods demonstrate the significance of our approach, in terms of both efficiency in performing the clustering task and of quality of the output clustering solutions.

2 Problem Background

The XML document clustering problem considered in this work can be defined in the homogeneous and heterogeneous context, in which the goal is: given a set of XML document instances from one domain, to group similar instances together (homogeneous context), and given a set of XML documents from different domains, to group the documents arising from the same domain together (heterogeneous context). Note however that even in the heterogeneous context



0	a b -1 c -1
1	z w -1 v u -1 -1
2	z w t -1 -1
3	a b c -1 -1 c e -1 -1
4	a b e -1 c -1 -1 c d -1 e -1 -1
5	z w -1 v u -1 t -1 -1

Fig. 1. Example of a tree-structured database consisting of 6 transactions

one can subdivide the clusters so as to further cluster the instances of each independent domain.

XML documents are conveniently modeled using a rooted ordered labeled tree (e.g., [1,13]), which can be denoted as $T = (v_0, V, L, E)$, where $v_0 \in V$ is the root vertex; V is the set of vertices or nodes; L is a labeling function that assigns a label $L(v)$ to every vertex $v \in V$; $E = \{(v_1, v_2) | v_1, v_2 \in V \wedge v_1 \neq v_2\}$ is the set of edges in the tree, and for each internal node the children are ordered from left to right. Note that in relation to the particular mining task being considered, each vertex $v \in V$ can be chosen to correspond to a different aspect of XML (e.g., in frequent subtree mining [7,19], the common choice is that $L(v)$ corresponds to an element name or a combination of element name and value). We will consider variation of XML content inclusion, and in Section 4 we explain how these will be captured in the tree representation of XML documents.

In order to represent trees in our approach, we used the pre-order (depth-first) string encoding (φ) [19]. This pre-order string encoding can be generated by adding vertex labels in the pre-order traversal of a tree $T = (v_0, V, L, E)$ and appending a backtrack symbol (e.g., '-1', '-1' $\notin L$) whenever we backtrack from a child node to its parent node. Figure 1 shows a tree-database (T_{db}) consisting of 6 tree instances (transactions). On the right of Fig. 1, the string encoding format representation commonly used in frequent subtree mining [7,19] is shown for T_{db}. The first column stores transaction (tree) identifiers while the second column contains the string encoding φ of each tree. Note that the backtrack symbols can be omitted after the last node in the string encoding (e.g., $\varphi(T_3)$ ='a b c -1 -1 c e').

3 Method Description

3.1 Conversion of Tree-Structured Database to Flat Representation

The first row of a (relational) table consists of attribute names, which in a tree database T_{db} are scattered through independent tree instances (transactions). One way to approach this problem is to first assume a structure/model according to which all the instances are organized. Each of the instances in a T_{db} should be a valid subtree of this document structure model, denoted as DSM.

The process of extracting a DSM from a tree database consists of traversing the tree database and expanding the current DSM as necessary so that every tree instance can be matched against DSM. This process was formally described

Algorithm 1. Database Structure Model (DSM) extraction from a tree database

Input: a tree database T_{db}
Output: the string encoding $\varphi(DSM)$ of the extracted DSM

```
 1: inputNodeLevel = 0                                   {current level of φ(tid_i)_k}
 2: DSMNodeLevel = 0                                     {current level of φ(T(h_max, d_max))_k}
 3: φ(DSM) = φ(tid_0)                                    {set default DSM (use x instead of labels)}
 4: for i = 1 → n − 1                                    {n = |T_db|}
 5:   for all φ(tid_i)_k ∈ φ(tid_i)
 6:     for p = 0 → (|φ(DSM)| − 1)
 7:       if φ(tid_i)_k = −1 then dec(inputNodeLevel)
 8:       else inc(inputNodeLevel)
 9:       if φ(DSM)_p = 'b' then dec(DSMNodeLevel)
10:       else inc(DSMNodeLevel)
11:       if inputNodeLevel ≠ DSMNodeLevel
12:         if φ(tid_i)_k = −1
13:           while inputNodeLevel ≠ DSMNodeLevel do
14:             inc(p)
15:             if φ(DSM)_p = −1 then dec(DSMNodeLevel)
16:             else inc(DSMNodeLevel)
17:         else
18:           while inputNodeLevel ≠ DSMNodeLevel do
19:             if φ(tid_i)_k ≠ −1 then φ(DSM)_{p+1} = 'x'
20:             else φ(DSM)_{p+1} = 'b'
21:             inc(k), inc(p)
22:             if φ(tid_i)_k = −1 then dec(inputNodeLevel)
23:             else inc(inputNodeLevel)
24: return φ(DSM)
```

in [6], and we replicate it here for clarity purposes. Let the tree database consisting of n transactions be denoted as $T_{db} = \{tid_0, tid_1, \ldots, tid_{n-1}\}$, and let the string encoding of the tree instance at transaction tid_i be denoted as $\varphi(tid_i)$. Further, let $|\varphi(tid_i)|$ denote the number of elements in $\varphi(tid_i)$, and $\varphi(tid_i)_k$ ($k = \{0, 1, \ldots, |\varphi(tid_i)| - 1\}$) denote the k-th element (a label or a backtrack) of $\varphi(tid_i)$. The same notation for the string encoding of the (current) DSM is used, i.e., $\varphi(DSM)$. However, rather than storing the actual labels in $\varphi(DSM)$, **x** is always stored to represent a node in general, and **b** to represent a backtrack. The process of extracting the DSM from T_{db} is depicted in Algorithm 1.

Note that we also store the number of times that a tree instance has matched a node or backtrack in the progressively built DSM. Occurrence of each node attribute (except the root node) within the DSM implies the existence of a specific backtrack attribute to ensure structural validity of the pre-order string encoding. Hence, after the whole DSM is extracted, it is safe to simply remove any nodes and backtracks that do not satisfy the minimum support set by the user. This will result in DSM reflecting the structure that was exhibited by as many instances as the user specified minimum support threshold. This relates to the next stage of populating the table, since the existence/non-existence of backtracks does not need to be stored in the generated table. These are stored

Table 1. Flat representation of T_{db} in Fig. 1 (optional columns in grey)

$\mathbf{x}_0$	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{b}_0$	$\mathbf{x}_3$	$\mathbf{b}_1$	$\mathbf{b}_2$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{b}_3$	$\mathbf{x}_6$	$\mathbf{b}_4$	$\mathbf{b}_5$
a	b	0	0	0	0	1	c	0	0	0	0	1
z	w	0	0	0	0	1	v	u	1	0	0	1
z	w	t	1	0	0	1	0	0	0	0	0	0
a	b	c	1	0	0	1	c	e	1	0	0	1
a	b	e	1	c	1	1	c	d	1	e	1	1
z	w	0	0	0	0	1	v	u	1	t	1	1

in the DSM and can be used for mapping structural information onto detected cluster constraints.

To illustrate the conversion process using DSM please refer back to Fig. 1. In this example the DSM is reflected in the structure of T_4 in Fig. 1 and it becomes the first row in Table 1. Since the order of the nodes (and backtracks) is important, the nodes and backtracks are labeled sequentially according to their occurrence in DSM. For nodes (labels in the pre-order string encoding of DSM), $\mathbf{x}_i$ is used as the attribute name, where i corresponds to the pre-order position of the node in DSM, while for backtracks, $\mathbf{b}_j$ is used as the attribute name, where j corresponds to the backtrack number in DSM. Hence, from our example in Fig. 1, $DSM = \mathbf{x}_0\ \mathbf{x}_1\ \mathbf{x}_2\ \mathbf{b}_0\ \mathbf{x}_3\ \mathbf{b}_1\ \mathbf{b}_2\ \mathbf{x}_4\ \mathbf{x}_5\ \mathbf{b}_3\ \mathbf{x}_6\ \mathbf{b}_4\ \mathbf{b}_5$. To fill in the remaining rows, every transaction from T_{db} is scanned and when a node or a backtrack is encountered, a '1' is placed to the matching column (i.e., under the matching node ($\mathbf{x}_i$) or backtrack ($\mathbf{b}_j$) in DSM), while the remaining entries are assigned values of '0' (non-existence). Table 1 shows the resulting FDT (structure preserving Flat Data representation for Tree-structured data).

During the FDT generation even if parts of a tree instance cannot all be matched to the DSM because of low occurrence, we still need to capture the partial information from the tree instance that structurally conforms to DSM. Each instance is matched against the DSM, and the current levels of the tree instances and DSM are tracked. Whenever the levels match, the label or '1' is stored in the corresponding column of the table, and when levels do not match, either the DSM or the tree instance encoding is traversed until the levels match. As an example, if minimum support threshold was set to 3 and the T_{db} in Fig. 1 was converted, the resulting DSM would be $DSM = \mathbf{x}_0\ \mathbf{x}_1\ \mathbf{x}_2\ \mathbf{b}_0\ \mathbf{b}_1\ \mathbf{x}_3\ \mathbf{x}_4\ \mathbf{b}_2\ \mathbf{b}_3$, while the resulting flat representation would be equivalent to the one in Table 1 when columns with attributes $\mathbf{x}_3, \mathbf{b}_1, \mathbf{x}_6, \mathbf{b}_4$ are removed, and the remaining attributes renumbered sequentially to reflect the new DSM. Note that all the grey-coloured columns from the tables can be safely removed, as any results obtained can be mapped back to the extracted DSM, and the cluster constraints enriched with structural information. Hence, no information is lost, while the complex process of incorporating structural information during clustering is avoided. The minimum support threshold should be set to reflect the percentage of documents (instances) that are expected to form the smallest group/cluster. For example, in the experiments presented in Section 4, since the number of instances from each class was known, the support was set not far below the percentage of instances

of minority class. The choice is thus dependent on some apriori expectation of the number of documents to be considered as part of one group/cluster. This comprises the tree-structured to flat data conversion step in our method, which enables a clustering approach to be directly applied without using specialized distance measures for tree structures.

3.2 Characteristics/Implications of the Proposed Approach

The DSM governs what a valid instance of a particular tree characteristic is, and the exact positions of the node within the DSM are taken into account. As an example, consider the subtree with encoding 'a b c' from the T_{db} in Fig. 1. Within the current frequent subtree mining framework (upon which XProj [1] and HCX [10] XML clustering approaches are based) this subtree would be considered to occur in two transactions T_3 and T_4. Using the proposed approach the occurrence of 'a b c' in T_3 would be considered different than in T_4, because node 'c' occurs at different (pre-order) position, and hence the instance is represented differently in the table. This can be seen from Table 1, since the occurrence of node 'c' in T_3 is matched against attribute node $\mathbf{x}_2$, while it is matched against attribute node $\mathbf{x}_3$ for its occurrence in T_4. This illustrates the key difference in our approach, and it implies that two document instances will be considered similar only if their substructures occur at the same/similar positions. Different occurrences of a substructure in the DSM may indicate that it is used in a different context, and hence a different domain, and in these cases such property is useful. This is especially the case if all the instances in a T_{db} follow the same structure and node layout as the general document model. This property could be less useful in scenarios, where the instances of a T_{db} may not follow the same order or not have all the elements of the general document structure (e.g., XML schema) available. In spite of this difference the proposed method is an alternative approach to clustering, and can arrive at high quality clusters as will be demonstrated in the experimental results given in the next section.

4 Experimental Evaluation

4.1 Experimental Methodology

Our approach is composed of 5 steps, covering data preparation, document structure model extraction, conversion to a flat representation, document clustering, and clustering evaluation.

Step 1. Our approach starts by creating a single XML document that contains all the instances of all XML documents considered. They are organized in such a way that all instances are siblings on the root level of the document for further processing. The second part of the data preparation phase is the creation of multiple variations of the input document. We have chosen the following variations and show it on the following example (with labels applied):

<[N]dataset [A]subject=[W]''astronomy''>[V]''Stars''</[N]dataset>

(1) tag names only, which are labeled as N (dataset in our example), (2) tag names and #PCDATA elements (values), which are labeled as N and V (`dataset` and `‘‘Stars’’` in our example), (3) tag names and attribute names, which are labeled as N and A (`dataset` and `subject` in our example), (4) tag names, #PCDATA elements (values) and attribute names, which are labeled as N, V and A (`dataset`, `‘‘Stars’’`, and `subject` in our example), and (5), tag names, #PCDATA elements (values), attribute names and attribute values, which are labeled as N, V, A and W (`dataset`, `‘‘Stars’’`, `subject`, and `‘‘astronomy’’` in our example). For each of these variations, we output a string representation file and a mapping file that contains the mappings between the original document and the string representation. In each of the variations, we consider each item as a separate node for further processing. For instance, variation NVAW of our example would look as follows:

```
<[N]dataset><[A]subject><[W]astronomy/></[A]subject><[V]Stars/></[N]dataset>
```

Each variation is stored in a commonly used string representation format [7,19], example of which was given in Fig. 1.

Step 2. This step performs the document structure model (DSM) extraction. The string encoding representation of each of the 5 variations from the previous step (N, NV, NA, NVA, NVAW) is traversed to generate the output that contains the string encoding of DSM (i.e., $\varphi(DSM)$) for each variation. As explained in Section 3, the user can specify the minimum support threshold so that the DSM only captures the structural characteristics if they have occurred in specified percentage of document instances. In our experiments, we choose the support value to be not far below the percentage of instances from the minority class.

Step 3. In the third step, the $\varphi(DSM)$ from the previous step and the respective output from step 1 (string encoding format representation of the tree database) are used to create the flat representation. This is done for two variations, one with the backtracks (-1 or $\mathbf{b}_i$) as attributes and second without, and each instance is labeled with a class value (if available). The output of each of these variations is the generated flat representation, example of which was given in Table 1.

Step 4. The fourth step converts these files into CLUTO dense matrix format, where we perform a number of CLUTO runs. Three runs are performed for each of the following similarity measures: *correlation coefficient* (*corr*), *Euclidean* (*eucl*) and *extended Jaccard* (*jacc*). The number of clusters is set with respect to the amount of different classes from the original input.

Step 5. Once CLUTO has finished generating all output files, clustering validation is performed to evaluate the clustering solution in terms of both internal and external clustering validity criteria. The internal validity criteria are based on the average of internal and external cluster similarity, respectively denoted as $ISim$ and $ESim$ as produced by CLUTO output, which essentially assess the cluster compactness and cluster separation, respectively, of a given clustering solution. For the exact formula of how such measures are computed, the reader is

referred to [20]. The external criteria aim to evaluate how well a clustering solution fits a known organization of the data into predefined classes; we hereinafter refer to this organization as reference classification. In this work we resorted to commonly used external criteria for document clustering, namely Entropy (E), Purity (Pty), micro-averaged F-measure (F^m), and macro-averaged F-measure (F^M) (with relating overall precision, P, and recall, R), definitions of which can be found in [15,20,16].

4.2 Data

The proposed method is evaluated on three sets of data. The first dataset (henceforth *Real*) was previously used in [3], and it contains collections of documents from 5 different domains, namely 217 astronomy documents (http://adc.gsfc.nasa.gov), 264 documents representing messages from a Web forum (http://userland.com), 64 press news documents (http://www.prweb.com/rss.php), 51 documents containing issues of SIGMOD record (http://www.dia.uniroma3.it/Araneus/Sigmod/) and 53 documents representing wrapper programs for Web sites. Thus, it is a representative of the problem in heterogeneous context (cf. Section 2).

The second dataset (henceforth *DEBII_N*) was generated by taking Apache2 (v2.2.3) web server log files from the DEBI Institute website (http://debii.curtin.edu.au) for a 4-month period (i.e., homogeneous context). The order of web pages accessed is organized as a tree where the navigational pattern is reflected through the pre-order traversal of the tree. This way of representing web logs is based on the LOGML representation [14], which allows for a more detailed and informative representation of web logs using an XML template. The difference is that we are only storing the web pages accessed as nodes in a tree, and ignore other information (e.g., time stamps). Hence, only option N (elements only) in tree generation is used by all the approaches compared. The instances were labeled according to 3 classes, namely, external access, within-university access, and within-DEBI-institute access. Hence, there is some overlap between the classes, as it is well possible that similar usage patterns occur within each class.

The third dataset (henceforth *Process*) was generated from publicly available process log datasets (http://prom.win.tue.nl/tools/prom/). These datasets describe examples of business process logs in MXML, a business process management XML based template. Several sets of process logs were taken, each of which was assigned a unique class (8 classes in total) depending which category/system/format it came from. The majority of process logs were quite similar in structure, as the same MXML format was used. This data is therefore a good example of content-based XML document clustering application, where it is important to consider more extensive content information in addition to element names. While this dataset is rather homogeneous by nature, due to different categories and/or source systems used for data generation, it can be seen as a representative of the problem in mixed heterogeneous/homogeneous context.

As previously mentioned, in our data preparation phase all of the documents are merged into a single tree database, and varied content is included, variations

Table 2. Dataset characteristics

	$\|Tr\|$	$\|L\|$	$Avg\|T\|$	$Avg\|D\|$	$Avg\|F\|$	$Max\|T\|$	$Max\|D\|$	$Max\|F\|$
*Real*_N	649	127	94.85	4.43	3.18	1440	9	330
*Real*_NA	649	158	11.94	4.59	3.19	1474	10	330
*Real*_NV	649	15572	150.92	5.35	1.67	2307	9	330
*Real*_NVA	649	15603	173.006	5.43	1.79	2341	10	330
*Real*_NVAW	649	18022	194.426	5.59	1.67	2375	11	330
*DEBII*_N	10996	16946	6.93	4.13	1.52	30	29	26
*Proc*_N	20968	8	80.33	2.2	3.63	1256	4	258
*Proc*_NA	20968	12	89.56	2.8	3.41	1975	4	258
*Proc*_NV	20968	22185	138.67	3.2	1.69	2260	4	256
*Proc*_NVA	20968	22189	147.92	3.2	1.79	3019	4	258
*Proc*_NVAW	20968	23883	156.36	3.8	1.73	3833	5	258

of which we will denote as *dataset*_N, *dataset*_NA, *dataset*_NV, *dataset*_NVA and *dataset*_NVAW. Table 2 summarizes the structural characteristics of each dataset and its content variation, and the following notation is used: $|Tr|$ - # of transactions (independent tree instances), $|L|$ - # of unique node labels; $|T|$ - # of nodes (size) in a transaction; $|D|$ - depth; $|F|$ - fan-out or degree.

4.3 Results

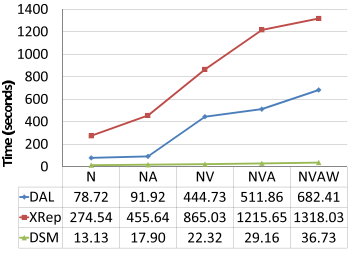
We assessed the clustering performance of our approach in terms of accuracy as well as efficiency, and also compared the results with those obtained by two existing XML clustering methods, namely the approach presented in [4] (hereinafter denoted as DAL) and XRep [3].

Unlike XRep (which is a stand-alone XML clustering method), DAL can enable the direct application of the CLUTO software, by providing it with a similarity matrix that stores the (normalized) structural distance values converted into similarity values (each distance value is subtracted to 1 and the difference is stored into the matrix). Hence, both our DSM and DAL are compared using CLUTO to cluster the data uniquely prepared by the different approaches. Note that all of the approaches have been enhanced in their preprocessing modules to deal with the variations of XML document as explained earlier (i.e., N, NA, NV, NVA, NVAW). Each algorithm name is appended with the symbol indicating the variation of content considered. In addition, our approach is also optionally appended with the similarity function it used within the CLUTO toolkit (e.g., DSM_corr_NA, DSM_eucl_NA, DSM_jacc_NA). All of our experiments were carried out on the same machine sequentially. The server in question is a quad socket quad core Xeon E7330 (2.4 GHz) machine, which is dedicated for running our experiments in a Windows Server 2008 64 bit environment. This machine has 128GB of RAM and we have allocated 50GB of memory to every experiment.

In Fig. 2(a) we summarize the clustering results for *Real* data, with input number of clusters set to 5. Note that for XRep we could not obtain the information necessary to determine the average internal and external similarity of the formed clusters, represented in column 8 and 9, respectively. The support used for DSM extraction phase was set to 5%. As can be seen, when *corr* similarity measure is used within our approach, better results were generally obtained in

	Pty	E	P	R	F^M	F^m	$ISim$	$ESim$
DAL_N	0.851	0.502	0.721	0.783	0.751	0.733	0.609	0.14
DAL_NA	0.901	0.505	0.8	0.94	0.864	0.875	0.597	0.09
DAL_NV	1	0	1	1	1	1	0.569	0.081
DAL_NVA	1	0	1	1	1	1	0.578	0.089
DAL_NVAW	1	0	1	1	1	1	0.561	0.081
XRep_N	1	0	1	1	1	1	-	-
XRep_NA	1	0	1	0.996	0.998	0.999	-	-
XRep_NV	1	0	1	1	1	1	-	-
XRep_NVA	1	0	1	0.996	0.998	0.999	-	-
XRep_NVAW	1	0	1	1	1	1	-	-
DSM_corr_N	1	0	1	1	1	1	0.777	0.014
DSM_corr_NA	0.998	0.408	0.999	0.996	0.998	0.998	0.782	-0.02
DSM_corr_NV	0.998	0.408	0.999	0.996	0.998	0.998	0.757	0.067
DSM_corr_NVA	0.998	0.408	0.999	0.996	0.998	0.998	0.767	0.004
DSM_corr_NVAW	0.998	0.408	0.999	0.996	0.998	0.998	0.764	0.004
DSM_eucl_N	0.966	0.136	1	0.593	0.745	0.626	0.605	0.003
DSM_eucl_NA	0.965	0.162	1	0.587	0.740	0.571	0.512	0.007
DSM_eucl_NV	0.995	0.066	1	0.627	0.771	0.624	0.389	0.002
DSM_eucl_NVA	0.994	0.043	1	0.641	0.781	0.642	0.427	0.002
DSM_eucl_NVAW	0.994	0.059	1	0.641	0.781	0.679	0.357	0.001
DSM_jacc_N	0.968	0.142	1	0.598	0.748	0.598	0.458	0.005
DSM_jacc_NA	0.965	0.162	1	0.571	0.727	0.592	0.494	0.005
DSM_jacc_NV	0.998	0.005	1	0.725	0.840	0.681	0.406	0.003
DSM_jacc_NVA	0.997	0.006	1	0.708	0.829	0.691	0.424	0.003
DSM_jacc_NVAW	0.997	0.006	1	0.714	0.833	0.683	0.422	0.002
DSM_corr_N(s=10)	0.919	0.075	0.799	0.913	0.852	0.858	0.735	0.008
DSM_corr_N(s=20)	0.919	0.075	0.799	0.941	0.864	0.892	0.794	-0.029

(a)



(b)

Fig. 2. *Real* data: (a) cluster quality comparison and (b) total runtime taken for different *Real* dataset variations

terms of Pty, F^M, and F^m; moreover, in case of F^M, recall scores were consistently higher than *eucl* and *jacc* over the variations of XML document (from about +0.42 to +0.22). Focusing on our *corr* based approach, we can see that the external-criteria-based quality of clusters was recognized as optimal for variation N (element names only); as far as internal evaluation, $ISim$ for variation N was slightly worse than variation NA only, nevertheless DSM $ISim$ values were always higher than the corresponding $ISim$ values obtained by the competing clustering methods. In general, DSM behaved as comparable to the competing methods with some minor differences. For example, our approach performed best for option N while the DAL approach did not achieve optimal results for this option. We also observed that, for this data, there was no difference in the results obtained by DSM when backtrack attributes were included. The last two columns of the table of Fig. 2(a) show how the different support thresholds (i.e., 10% and 20%) affect the results for DSM_corr_N option. Note that the percentage of instances from the minority class in this dataset, cover less than 10% of the database, and hence the structural characteristics unique to those instances will not be considered in DSM. Hence, one can expect less optimal values for all of the values except for internal and external cluster similarity, which can be confirmed by comparing these cases with the results of row 12. At $s = 20\%$, more optimal values for internal and external cluster similarity were achieved, but this could be mainly due to the fact that the unique structural characteristics of the instances of minority class were not captured by any formed cluster.

The time performance comparison between the different approaches, for the different variations of the data, is shown in Fig. 2(b). This is the total time

Table 3. Cluster quality comparison for *DEBII* data

	Pty	E	P	R	F^M	F^m	$ISim$	$ESim$
DAL_N	0.801	0.860	0.349	0.513	0.415	0.655	0.277	0.187
XRep_N	0.897	0.188	1	0.001	0.002	0.273	-	-
DSM_corr_N	0.801	0.806	0.384	0.504	0.436	0.504	0.470	0.191
DSM_corr_N_B	0.801	0.797	0.387	0.511	0.441	0.5	0.505	0.237
DSM_eucl_N	0.802	0.622	0.639	0.146	0.237	0.332	0.537	0
DSM_eucl_N_B	0.802	0.622	0.639	0.146	0.237	0.332	0.537	0
DSM_jacc_N	0.801	0.601	0.546	0.092	0.157	0.316	0.475	0
DSM_jacc_N_B	0.801	0.647	0.541	0.162	0.249	0.459	0.475	0
DSM_corr_N(s = 10)	0.801	0.546	0.358	0.446	0.397	0.501	0.468	0.102
DSM_corr_N(s = 20)	0.801	0.513	0.389	0.446	0.416	0.57	0.507	0.003

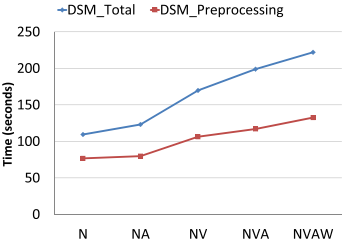
taken including preprocessing and clustering of the data. As more content from XML data was considered, the task became more complex (especially when element values were considered), but such a higher complexity negatively affected mainly the performance of the competing methods; precisely, total runtimes (in seconds) were 2744.14 (DAL), 115885.16 (XRep), and 91.30 (DSM), where in all cases the preprocessing stage took about from 88% (DAL) to 99% (DSM) of the total runtime. Overall, the proposed approach dramatically outperformed the competing methods in terms of runtime (from 2 up to 4 orders of magnitude) and the improvements were particularly evident for the content-oriented variations.

Table 3 shows the clustering results for *DEBII* data, with input number of clusters set to 3. The support used for DSM extraction phase was set to 3%. Note that here we have also included the results when the backtrack attributes were considered, but as one can see there was only a slight difference for *corr* based similarity measure. Again, DSM performed as good as or better than DAL in terms of all criteria except F^m, whereas XRep was not able to detect the 3-class organization of the data as it ended to identify many small clusters. The total runtimes (seconds) were as follows: 91.3 (90.4 for preprocessing) for DSM, 2744.14 (2421.65 for preprocessing) for DAL, and 115885.16 for XRep. The last two columns of Table 3 show how the different support thresholds (i.e., 10% and 20%) affect the results for DSM_corr_N option. Comparing with the results of row 4, one can see that increasing the support has no effect on purity, better results for entropy and external cluster similarity, while there is a small difference in remaining criteria. In contrast to the previous experiment, in this case increasing the support can positively affect entropy, which is because in *DEBII* data there is a higher possibility of overlapping classes.

When we ran the different approaches on *Process* data (input number of clusters set to 8), the XRep approach exhausted the available memory and the DAL approach took nearly 3 days to complete. The support used for DSM extraction phase was set to 0.4%. In Fig. 3(a) we show clustering results for our approach (no backtracks considered) and DAL approach. In contrast to the other two data, higher external-criteria-based quality was achieved by our approach when more content variations were included, leading to improvements up to about 0.22 (F^m) on NVAW variation. Higher intra- and extra-cluster similarity were instead obtained on N variation. However, this was not the case for the DAL approach, as F^m only increased by 0.004 for NVAW option and $ISim$ and $ESim$ were higher for

	Pty	E	P	R	F^M	F^m	$ISim$	$ESim$
DAL_N	0.396	0.999	0.125	0.998	0.222	0.363	0.488	0.246
DAL_NA	0.396	0.999	0.228	0.874	0.361	0.362	0.404	0.224
DAL_NV	0.397	0.999	0.35	0.751	0.478	0.363	0.492	0.273
DAL_NVA	0.397	0.999	0.35	0.752	0.478	0.363	0.516	0.253
DAL_NVAW	0.4	0.996	0.248	0.886	0.387	0.367	0.493	0.189
DSM_corr_N	0.583	0.73	0.341	0.443	0.386	0.4	0.838	0.464
DSM_corr_NA	0.601	0.667	0.461	0.501	0.480	0.431	0.829	0.431
DSM_corr_NV	0.752	0.446	0.43	0.549	0.483	0.548	0.753	0.243
DSM_corr_NVA	0.756	0.423	0.445	0.583	0.505	0.543	0.746	0.245
DSM_corr_NVAW	0.787	0.385	0.556	0.604	0.579	0.618	0.749	0.206

(a)



(b)

Fig. 3. *Process* data: (a) cluster quality comparison and (b) preprocessing and total runtime taken by DSM for different variations of the dataset

NVA and NVAW options. Overall, one can see DSM performed better than DAL in terms of all criteria except R and $ESim$; however, these two criteria should be considered w.r.t. P and $ISim$, respectively, and hence the results were still comparable according to all four criteria as a whole. The DAL approach took approximately 21.5h to complete the clustering for NVAW variation, 15.5h for NV, 18.5h for NVA, 3h for NA and 2.5h for N. Due to these large runtimes, in Fig. 3(b) we show the total and preprocessing time taken by our approach for the variations of *Process* data. As expected, the presence of element values impacted more on the preprocessing as well as the clustering step than non-V variations.

5 Conclusions

In this paper we have presented an alternative approach to XML document clustering. The structural aspects from XML data are first extracted from the document to generate a structure-preserving flat data format, to which well-established traditional clustering approaches, such as the partitional one, can be directly applied. Within this view, we have taken the CLUTO clustering toolkit as a case in point and have compared the results with existing structural clustering methods. There is strong empirical evidence that the complexity associated with XML document clustering due to the structural aspects that need to be considered, can be significantly reduced with the proposed approach. At the same time high quality clustering results can be obtained that are comparable or better than those obtained by XML clustering methods that deal with complex structural similarity determination. Furthermore, given that many more clustering techniques exist for flat data representation, the conversion approach in itself can potentially enable a wider range of techniques to be applied for XML document clustering problem.

Acknowledgments. The authors wish to acknowledge Theodore Dalamagas for providing his XML clustering tool [4].

References

1. Aggarwal, C.C., Ta, N., Wang, J., Feng, J., Zaki, M.: XProj: a framework for projected structural clustering of XML documents. In: Proc. ACM KDD Conf., pp. 46–55 (2007)

2. Bille, P.: A survey on tree edit distance and related problems. Theoretical Computer Science 337(1–3), 217–239 (2005)
3. Costa, G., Manco, G., Ortale, R., Tagarelli, A.: A Tree-Based Approach to Clustering XML Documents by Structure. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) PKDD 2004. LNCS (LNAI), vol. 3202, pp. 137–148. Springer, Heidelberg (2004)
4. Dalamagas, T., Cheng, T., Winkel, K.-J., Sellis, T.K.: A methodology for clustering XML documents by structure. Information Systems 31(3) (2006)
5. Doucet, A., Lehtonen, M.: Unsupervised Classification of Text-Centric XML Document Collections. In: Fuhr, N., Lalmas, M., Trotman, A. (eds.) INEX 2006. LNCS, vol. 4518, pp. 497–509. Springer, Heidelberg (2007)
6. Hadzic, F.: A Structure Preserving Flat Data Format Representation for Tree-Structured Data. In: Proc. PAKDD Workshops (QIME 2011), Springer, Heidelberg (2011)
7. Hadzic, F., Tan, H., Dillon, T.S.: Mining of Data with Complex Structures, 1st edn. SCI, vol. 333. Springer, Heidelberg (2011)
8. Joshi, S., Agrawal, N., Krishnapuram, R., Negi, S.: A bag of paths model for measuring structural similarity in Web documents. In: Proc. ACM KDD Conf., pp. 577–582 (2003)
9. Karypis, G.: CLUTO - Software for Clustering High-Dimensional Datasets (2002/2007), http://glaros.dtc.umn.edu/gkhome/cluto/cluto/download
10. Kutty, S., Nayak, R., Li, Y.: HCX: an efficient hybrid clustering approach for XML documents. In: Proc. ACM Symposium on Document Engineering, pp. 94–97 (2009)
11. Kutty, S., Nayak, R., Li, Y.: XML Documents Clustering using a Tensor Space Model. In: Huang, J.Z., Cao, L., Srivastava, J. (eds.) PAKDD 2011, Part I. LNCS, vol. 6634, pp. 488–499. Springer, Heidelberg (2011)
12. Lian, W., Cheung, D.W.-L., Mamoulis, N., Yiu, S.-M.: An Efficient and Scalable Algorithm for Clustering XML Documents by Structure. IEEE Transactions on Knowledge Data Engineering 16(1), 82–96 (2004)
13. Nierman, A., Jagadish, H.V.: Evaluating Structural Similarity in XML Documents. In: Proc. WebDB Workshop, pp. 61–66 (2002)
14. Punin, J.R., Krishnamoorthy, M.S., Zaki, M.J.: LOGML: Log Markup Language for Web Usage Mining. In: Kohavi, R., Masand, B., Spiliopoulou, M., Srivastava, J. (eds.) WebKDD 2001. LNCS (LNAI), vol. 2356, pp. 88–112. Springer, Heidelberg (2002)
15. Steinbach, M., Karypis, G., Kumar, V.: A comparison of document clustering techniques. In: Proc. KDD Workshop on Text Mining (2000)
16. Tagarelli, A., Greco, S.: Semantic clustering of XML documents. ACM Transactions on Information Systems 28(1) (2010)
17. Yao, J.T., Varde, A., Rundensteiner, E., Fahrenholz, S.: XML Based Markup Languages for Specific Domains. In: Web-based Support Systems. Advanced Information and Knowledge Processing, pp. 215–238. Springer, London (2010)
18. Yoon, J.P., Raghavan, V., Chakilam, V., Kerschberg, L.: BitCube: A Three-Dimensional Bitmap Indexing for XML Documents. Journal of Intelligent Information Systems 17(2–3), 241–254 (2001)
19. Zaki, M.J.: Efficiently Mining Frequent Trees in a Forest: Algorithms and Applications. IEEE Transactions on Knowledge and Data Engineering 17(8), 1021–1035 (2005)
20. Zhao, Y., Karypis, G.: Empirical and Theoretical Comparison of Selected Criterion Functions for Document Clustering. Machine Learning 55(3), 311–331 (2004)

Author Index