Lecture Notes in Artificial Intelligence 5433

Edited by R. Goebel, J. Siekmann, and W. Wahlster

Subseries of Lecture Notes in Computer Science

Sanjay Chawla Takashi Washio
Shin-ichi Minato Shusaku Tsumoto
Takashi Onoda Seiji Yamada
Akihiro Inokuchi (Eds.)

New Frontiers in Applied Data Mining

PAKDD 2008 International Workshops
Osaka, Japan, May 20-23, 2008
Revised Selected Papers

Springer

Series Editors

Randy Goebel, University of Alberta, Edmonton, Canada
Jörg Siekmann, University of Saarland, Saarbrücken, Germany
Wolfgang Wahlster, DFKI and University of Saarland, Saarbrücken, Germany

Volume Editors

Sanjay Chawla
University of Sydney, NSW, Australia
E-mail: chawla@it.usyd.edu

Takashi Washio
Akihiro Inokuchi
Osaka University, Osaka, Japan
E-mail: {washio,inokuchi}@ar.sanken.osaka-u.ac.jp

Shin-ichi Minato
Hokkaido University, Sapporo, Japan
E-mail: minato@ist.hokudai.ac.jp

Shusaku Tsumoto
Shimane University, Izumo, Shimane, Japan
E-mail: tsumoto@computer.org

Takashi Onoda
Central Research Institute of Electric Power Industry, Tokyo, Japan
E-mail: onoda@criepi.denken.or.jp

Seiji Yamada
National Institute of Informatics, Tokyo, Japan
E-mail: seiji@nii.ac.jp

Library of Congress Control Number: 2009920983

CR Subject Classification (1998): I.2, H.2.7-8, H.3, H.5.1, G.3, J.3

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN 0302-9743
ISBN-10 3-642-00398-2 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-00398-1 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12613861 06/3180 5 4 3 2 1 0

Preface

As data mining techniques and tools mature, their application domains extend to previous unchartered territories. The common theme of the workshops organized along with the main 2008 Pacific Asia Conference on Knowledge Discovery and Data Mining (PAKDD) in Osaka, Japan was to extend the application of data mining techniques to new frontiers. Thus the title of the proceedings: "New Frontiers in Application of Data Mining."

For the 2008 program, three workshops were organized.

1. Algorithms for Large-Scale Information Processing (ALSIP). The focus of the workshop was novel algorithms and data structures to deal with processing of very large data sets.
2. Data Mining for Decision Making and Risk Management (DMDRM), which emphasized applications of risk information derived from data mining techniques on diverse applications ranging from medicine to marketing to chemistry.
3. Interactive Data Mining (IDM), which emphasized the relationship between techniques in data mining and human–computer interaction.

In total 38 papers were submitted to the workshops. After consultation with the workshop Chairs who were asked to rank their submissions, 18 were accepted for publication in this volume. We hope that the published papers propel further interest in the growing field of knowledge discovery in databases (KDD).

The paper selection of the industrial track and the workshops was made by the Program Committee of each organization. Upon the paper selection, the book was edited and managed by the volume editors.

December 2008

Sanjay Chawla
Takashi Washio

Workshop on Algorithms for Large-Scale Information Processing in Knowledge Discovery

The International Workshop on Algorithms for Large-Scale Information Processing in Knowledge Discovery (ALSIP 2008) was held on May 20, 2008 at Hotel Seagull Tempozan Osaka, Japan, in conjunction with the 12th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2008). This workshop was co-organized by MEXT Japan Grant-in-Aid for Science Research and Priority Area: Cyber Infrastructure for the Information-explosion Era, and the JSPS G-COE program of Hokkaido University: Next-Generation Knowledge Information Science for Future Science and Technology. The workshop aimed to exchange fresh ideas on large-scale data processing in problems such as data mining, clustering, machine learning, statistical analysis, and other computational aspect of knowledge discovery problems. Information created by people has increased rapidly since the year 2000, and now we are in a time that we could call the information-explosion era. To cope with such a large-scale information space, novel algorithms and data structures are desired for solving various problems in the area of knowledge discovery. This workshop aimed to exchange fresh ideas on large-scale data processing for various knowledge discovery problems. The topics of the workshop in the call for papers included:

- Machine learning, clustering and statistical methods
- Large-scale itemset mining and associate rule mining
- Graph-based data structures for knowledge representation
- Knowledge data compression and indexing
- Knowledge discovery from text and the Web
- Knowledge discovery from unstructured and multimedia data
- Knowledge discovery from data stream and spatial/temporal data
- Knowledge discovery in network and link data
- Biomedical knowledge discovery, analysis of microarray and gene deletion data
- Information extraction from scientific literature
- Active knowledge discovery
- Data and knowledge visualization
- Other computational aspects of knowledge discovery problems.

ALSIP 2008 was organized as a full-day workshop. All submitted papers were reviewed for quality and originality by the Program Committee. We accepted 11 technical papers out of 16 submissions. We also invited a special talk "The Challenge of Mining Billions of Transactions" by Osmar Zaiane (University of Alberta, Canada). By additional reviewing after the workshop, we selected 10 papers for this proceedings volume.

We are grateful for the great support from the ALSIP 2008 Program Committee members. We are also grateful to the PAKDD 2008 committee for their help,

especially Takashi Washio for his arrangements in conjunction with the PAKDD. We gratefully acknowledge Masaru Kitsuregawa, the leader of the MEXT project of the Information-explosion Era, for giving us an opportunity to organize this workshop, and Hiroki Arimura, the leader of the Hokkaido University GCOE program, for his support of the ALSIP workshop.

Workshop Chair

Shin-ichi Minato	Hokkaido University, Japan

Program Committee Members

Masaru Kitsuregawa	University of Tokyo, Japan
Hiroki Arimura	Hokkaido University, Japan
Jean-Francois Boulicaut	INSA Lyon, France
Koichi Hirata	Kyushu Institute of Technology, Japan
Hideki Isozaki	NTT Communication Science Labs., Japan
Kimihito Ito	Research Center for Zoonosis Control, Hokkaido University, Japan
Yoshitaka Kameya	Tokyo Institute of Technology, Japan
Hisashi Kashima	IBM Tokyo Research Lab., Japan
Roni Khardon	Tufts University, USA
Hiroyuki Kitagawa	University of Tsukuba, Japan
YongJin Kwon	Korea Aerospace University, Korea
Seishi Okamoto	Fujitsu Labs. Japan
Jan Poland	ABB Corporate Research Centers, Switzerland
Michele Sebag	LRI, University Paris-Sud, France
Masayuki Takeda	Kyushu University, Japan
Koji Tsuda	Max Planck Institute for Biological Cybernetics., Germany
Shusaku Tsumoto	Shimane University, Japan
Takeaki Uno	National Institute of Informatics, Japan
Takashi Washio	Osaka University, Japan
Akihiro Yamamoto	Kyoto University, Japan
Thomas Zeugmann	Hokkaido University, Japan

Workshop on Data Mining for Decision Making and Risk Management

The organizations and systems in our modern society have become large and complex to provide more advanced services due to the growing variety of social demands. Such organizations and systems are efficient but highly complex, and can cause various unexpected situations. According to this observation, the importance of decision making and risk management of these organizations and systems has been strongly emphasized in recent years. On the other hand, accumulation of a large amount of data on the operations of the organizations and systems has become easier with the introduction of information technology. These data can be used to support decision making or risk management in organizations and systems.

This workshop focused on both data mining and statistical techniques to detect and analyze the risks potentially existing in the organizations and systems and to utilize the risk information for their better management and decision support. The topics covered: (1) data mining and machine learning approaches, (2) statistical approaches, (3) chance discovery, (4) active mining, and (5) application of these techniques to medicine, marketing, security, decision support in business, social activities, human relationships, chemistry and sensor data.

The topics of the workshop in the call for papers included:

- Data mining for decision making/risk management
- Novel statistical approach for decision making/ risk management
- Chance discovery for decision making/risk management
- Active mining for decision making/risk management
- Exploratory data analysis for decision making/risk management
- Machine learning for decision making/risk management
- Other techniques for risk detection, analysis and utilization of risk information
- Applications in the fields (but not limited to) of:
  - Medicine
  - Marketing
  - Security
  - Decision support in business
  - Social activities
  - Human relationships
  - Chemistry
  - Sensor data

Workshop Chairs

Shusaku Tsumoto	Shimane University, Japan
Hiroe Tsubaki	Tsukuba University, Japan
Tzung-Pei Hong	National University of Kaohsiung, Taiwan

Program Committee

Shoji Hirano	Shimane University, Japan
Tony Hu	Drexel University, USA
Genshiro Kitagawa	Institute of Statistical Mathematics, Japan
T.Y. Lin	San Jose State University, USA
Yukio Ohsawa	The University of Tokyo, Japan
Vijay Raghavan	University of Lousiana, USA
Yong Shi	Chinese Academy of Science, China
Andrzej Skowron	Warsaw University, Poland
Guoyin Wang	Southwest Jiaotong University, China
Katsutoshi Yada	Kansai University, Japan
Dirk Van den Poel	Ghent University, Belgium
Ning Zhong	Maebashi Institute of Technology, Japan

Workshop on Interactive Data Mining Overview

The International Workshop on Interactive Data Mining 2008 (IDM08) was held in conjunction with the 12th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2008), Osaka, Japan, May 2008. The workshop aimed at sharing and comparing various fields of interactive data mining research such as interactive information retrieval, information gathering systems, personalization systems, recommendation systems, user interfaces and so on. In summary, the workshop provided a discussion forum for researchers working on interactive data mining where the attendees discussed various aspects of this field. The motivation for the workshop is to discuss recent progress in the research field of interactive data mining, which includes interactions between a human and computers through Web systems, agent systems, sensor systems, or robot systems. Various areas of interactive data mining research have been realized through related technologies including interactive information retrieval, information gathering systems, personalization systems, recommendation systems, user interfaces and so on. Each study and development has been done independently in different research fields such as the information retrieval research field, Web intelligence research field, electric power systems research field, user interfaces research field and so on. However, this situation might discourage us from studying interactive data mining from the unified view of computer–human interaction and making interactive data mining more friendly by applying computational intelligence. Hence, we held the workshop entitled "Interactive Data Mining" in the Pacific-Asia Conference on Knowledge Discovery and Data Mining 2008, to gather together a variety of researchers in diverse fields such as knowledge discovery, information retrieval, Web intelligence, electric power systems, user interfaces and data mining.

The topics of the workshop in the call for papers included, but were not limited to:

- Interactive information retrieval
- Interactive information gathering systems
- Interactive personalization systems
- Interactive Web systems
- Interactive recommendation systems
- Interactive knowledge discovery
- Interactive user interfaces
- Interactive risk mining for electric power systems

All submitted papers were carefully peer reviewed by the Program Chairs. We accepted five papers out of seven submissions. The acceptance rate

is approximately 70%. We would like to thank all the authors who submitted papers to the workshop and participated in the interesting discussions at the workshop.

Program Chairs

Takashi Onoda	Central Research Institute of Electric Power Industry, Japan
Seiji Yamada	National Institute of Informatics, Japan

Workshop on Interactive Data Mining Overview

The International Workshop on Interactive Data Mining 2008 (IDM08) was held in conjunction with The 12th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2008), Osaka, Japan, May 2008. The workshop aimed at sharing and comparing various interactive data mining researches such as interactive information retrieval, information gathering systems, personalization systems, recommendation systems, user interfaces and so on. In summary, the workshop gave a discussion forum for researchers working on interactive data mining where the attendees discussed various aspects on interactive data mining. The motivation of the workshop is to discuss recent progress in the research field of interactive data mining, which includes interactions between a human and computers through web systems, agent systems, sensor systems, or robot systems. Various interactive data mining researches have been realized through related technologies including interactive information retrieval, information gathering systems, personalization systems, recommendation systems, user interfaces and so on. Each study and development has been done independently in different research fields such as information retrieval research field, web intelligence research field, electric power systems research field, user interfaces research field and so on. However, this situation might discourage us from studying interactive data mining from unified view of computer-human interaction and making interactive data mining more friendly by applying computational intelligence. Hence, at this time, we will hold the workshop entitled "Interactive Data Mining" in the Pacific-Asia Conference on Knowledge Discovery and Data Mining 2008, to gather a variety of researchers in diverse fields like knowledge discovery, information retrieval, web intelligence, electric power systems, user interfaces and data mining.

The topics of the workshop in call for papers included, but are not limited to:

- Interactive Information Retrieval
- Interactive Information Gathering Systems
- Interactive Personalization Systems
- Interactive Web Systems
- Interactive Recommendation Systems
- Interactive Knowledge Discovery
- Interactive User Interfaces
- Interactive Risk Mining for Electric Power Systems, etc

All submitted papers were carefully peer reviewed by program chairs. We accepted 5 papers out of 7 submissions. The acceptance rate is approximately 70%.

We would like to thank all the authors who submitted papers to the workshop and participated in the interesting discussions at the workshop.

Program Chairs

Takashi Onoda — Central Research Institute of Electric Power Industry, JP
Seiji Yamada — National Institute of Informatics, JP

Table of Contents

Workshop of ALSIP 2008

Workshop of DMDRM 2008

Workshop of IDM 2008

Flexible Framework for Time-Series Pattern Matching over Multi-dimension Data Stream

Takuya Kida, Tomoya Saito, and Hiroki Arimura

Hokkaido University, Kita 14, Nishi 9, Kita-ku, Sapporo, 060-0814 Japan
{kida,saito,arim}@ist.hokudai.ac.jp

Abstract. In this paper, we study a complex time-series pattern matching problem over a multi-dimension continuous data stream. For each data stream, a pattern is given as a sequence of predicates, which specify a sequence of element sets on the stream. The pattern matching problem over such a multi-dimension data stream, is to find all occurrences where all predicates in the patterns are satisfied. We propose a flexible and extensible framework to solve the problem, which is based on bit-parallel pattern matching method that simulates NFAs for the pattern matching efficiently by a few logical bit operations. We consider four types of data streams especially: textual, categorical, ordered, and numeric, that is, those are a sequence of strings, concepts with taxonomic information, small integers, and real numbers (or large integers), respectively. We also present the time complexities to do pattern matching for those data types.

1 Introduction

Information retrieval is one of the basic means for extracting useful information from massive collections of text data. In particular, *pattern matching problem* studies the design and the analysis of efficient algorithms for textual problems arising in information retrieval and provides theoretical basis for implementing fast information retrieval systems with large text data.

On the other hand, by the rapid growth of network and sensor technologies, a new class of data-intensive applications such as sensor networks and network management emerged [1,2,3]. In these applications, the data are usually numerical, textual in some cases, or combinations of them. Moreover they are not static collections but transient streams, where a data sequence is an unbounded and rapid stream of individual data items that may arrive continuously. Then fast and flexible pattern matching algorithms for such data stream are desired.

Harada [4,5] and Sadri *et al.*[6] considered complex time-series pattern matching problem on streams consisting of single numerical values instead of characters, where a pattern is a sequence of numerical predicates on a set of inequations and equations as shown in Fig. 1. For a stream $S = (1, 5, 3, 5, 4, 2, 4, 1, 2, 2)$ of length 10, for example, the pattern P in Fig. 1 matches at two positions from the second to the 6-th and from the 5-th to the 9-th of S.

S. Chawla et al. (Eds.): PAKDD 2008 Workshops, LNAI 5433, pp. 1–12, 2009.

$P = \phi_1 \cdot \phi_2 \cdot \phi_3 \cdot \phi_4 \cdot \phi_5$ where
$\phi_1 = (X > 2)$, $\phi_2 = (X < 5)$, $\phi_3 = (X > 2 \wedge X < 7)$, $\phi_4 = (X < 5)$,
and $\phi_5 = (X < 3)$.

Fig. 1. An example of complex pattern over numerical streams

We addressed the problem and proposed in [7] a new approach to it. We presented a family of efficient algorithms, called *BPS*(Bit-parallel matching on Streams), that use bit-parallel matching method [8] for complex time-series patterns over a numerical stream. Assuming that registers of the bit width w, say 32 or 64 bit, we show by theoretical analysis that a variant *BPS_tab* algorithm for ordered values finds the set of all occurrences of a given pattern of length m in an input stream of length n in $O(\frac{1}{w}mn)$ time and *BPS_bin* algorithm for real values runs in $O(\frac{1}{w}mn \log m)$ time. This time complex properly improves the time complexity $O(mn)$ of the previous algorithms. The keys of the BPS algorithm is fast NFA (nondeterministic finite automata) simulation by bit-parallel processing as well as a mechanism called a predicate lookup table, which is essentially a lookup table for the indices of pattern predicates by a given key value. By these techniques, we obtain $O(w)$ times speed up in order to break $O(mn)$ barrier for complex pattern matching over single numerical streams.

Now our new goal and objective in the next stage is to deal with *multi-dimension data stream*. We should consider here that a combination of several types of data, such as numerical, textual, and so on, may arrive continuously. Consider that, for example, a textual data stream with its morphological analysis results and dependency parsing results which processed by some tools on the fly, is inputed into a search system continuously. The stream will consist of a sequence of records which may include parsed phrases (morphemes), word classes, dependency scores, and other features. For such data stream, we are eager to do pattern matching for a complex pattern like 'string `[Rr]iver` or `[Mm]ountain` after a noun (which is a superordinate concept of proper noun, common noun, collective noun, material noun, and abstract noun) which follows some adjective within 5 morphemes whose length is longer than 8,' to find a phrase "Seine River Cruise for a fantastic view."

Another important example is a network access log. A web server logs informations about user requests and the server statuses continuously and permanently. Each log entry may include the remote IP address, the access time and date, the request command, the status code, and the transfer data size as follows:

```
192.168.0.1 [01/Oct/2008:10:48:19 +0900] "GET / HTTP/1.1" 200 13288
192.168.0.2 [01/Oct/2008:10:49:30 +0900] "GET /a.gif HTTP/1.1" 200 765
```

To do time-series pattern matching on such log data will help in detecting intrusions and analyzing the log data.

In this paper, we propose a unified framework for time-series pattern matching over multi-dimension data stream, based on BPS algorithms. The framework can treat categorical data streams, which is a sequence of sets of *concepts* with a *concept hierarchy* (or *taxonomic information*) in addition to numerical and

textual data streams, and it can allow complex and flexible time-series pattern matching over them. In [9] we have already proposed a bit-parallel algorithm which solves the pattern matching problem on a concept sequence with a concept hierarchy. One of major contributions of this paper is to combine the algorithm of [9] into the BPS scheme.

The key of our framework is to separate the matching mechanism into two parts: one is to estimate each element of each data stream in order to decide which predicates are satisfied, and the other is to estimate each pattern for each stream. We use a suitable matching algorithm for each data type of the stream to the former, and use the BPS scheme to the latter.

The organization of this paper is as follows. In Section 2, we prepare basic definitions. In Section 3, we present our pattern matching framework for multi-dimension data streams, and we conclude in Section 4.

Related works: We presented in [9] a pattern matching algorithm with taxonomic information, where taxonomy is a partially ordered set of letters (*concepts*) describing an IS-A hierarchy, and the given pattern and text are both described as a sequence of concepts. Each concept on the pattern can match any lower concepts on the text. The algorithm is based on bit-parallel method, and it runs in $O(\frac{mn}{w})$ time with $O(m+\frac{mh}{w})$ preprocessing and $O(\frac{m\sigma}{w})$ extra space on a random access machine (RAM), where h and σ are the size of the taxonomic information and the cardinality of the taxonomy, respectively.

Harada [4,5] and Sadri *et al.*[6] presented efficient algorithms L2R [4,6] and R2L [5] by extending well-known string pattern matching algorithms KMP [10] and BM [11], respectively, for numerical streams, which use skipping of the input values based on static dependency analysis on numerical attributes. Both algorithms run in $O(mn)$ times.

So-called point sequence matching or (δ, γ)-matching, often discussed in music retrieval, are related to our pattern matching problem [12,13,14,15]. Such pattern matching deals with single numerical data streams and allows some approximation.

Hyrro *et al.*[16] presented an efficient pattern matching algorithm of multi-byte text based on bit-parallel method. They combine the outputs of Aho-Corasick pattern matching machine [17], which recognize all multi-byte codes and given patterns, into the inputs of bit-parallel pattern matching algorithms. Its architecture is similar to that of our framework.

Arikawa *et al.*[18] developed a search based textual database, called SIGMA. It is based on Aho-Corasik pattern matching machine, and provides flexible and powerful search environment. Interstage Shunsaku Data Manager[1] developed by FUJITSU corporation is a powerful successor of SIGMA. However, time-series pattern matching is not considered on those systems.

Matsumoto *et al.*[19] developed an annotated corpora management and search system, called ChaKi. It provides a powerful search environment on annotated corpora, such as word search, attribute search, and some kind of regular expression search for words and attributes, but it has neither combination search

[1] http://interstage.fujitsu.com/jp/shunsaku/

of numerical data and the other data type, nor a kind of approximate search. Our proposed framework may contribute to enhance the search ability of such systems.

2 Preliminaries

We denote by $\mathbb{N}$ and $\mathbb{R}$ the set of all nonnegative integers and all real numbers, respectively. Let Σ be an *alphabet*. Let binary relation $\succeq$ be an order on Σ. A *sorted alphabet* is an ordered set $(\Sigma, \succeq)$. We call Σ as *textual* and its element as a *symbol* or a *character* if Σ is a finite and $\succeq$ satisfies only reflective property. We call Σ as *categorical* and its element as a *concept* if Σ is a finite and $\succeq$ is a partial order. Moreover, for any concepts $c, d \in \Sigma$, we say that c *matches* d if $c \succeq d$. We call Σ as *numerical* and its element as a *(numerical) value* if Σ is a subset of real numbers or *large* integers, and $\succeq$ is a total order. We call Σ as *ordered* and its element as an *(ordered) value* if Σ is isomorphic with $\{1, \ldots, c\} \subseteq \mathbb{N}$ for a *small* positive integer c, and $\succeq$ is a total order. In this paper, we assume that c is a 16-bit integer, say, $0 \leq c \leq 2^{16} = 65,535$.

An element $T = a_1 \cdots a_n (n \geq 0)$ in Σ^* is especially called a *string* or a *word* for textual alphabet Σ, and a *concept sequence* for categorical alphabet Σ, respectively. Then, the length of T is denoted by $|T| = n$. For $1 \leq i \leq |T|$, the i-th symbol of a concept sequence T is denoted by $T[i] = a_i$, and for $1 \leq i \leq j \leq |T|$, the factor of T that begins at position i and ends at position j is denoted by $T[i:j] = a_i \cdots a_j$.

Relational k-dimension data streams: Let Σ_s be a textual alphabet, Σ_c be a categorical alphabet, Σ_o be an ordered alphabet. and Σ_n be a numerical alphabet. Let Δ be k-term class $\Delta = (\Sigma_1, \ldots, \Sigma_k) \in \{\Sigma_s{}^*, \Sigma_c, \Sigma_o, \Sigma_n\}^k$. For $s = (s[1], \ldots, s[k])$ and $\Delta = (\Sigma_1, \ldots, \Sigma_k)$, we denote $s \in \Delta$ if $s[i] \in \Sigma_i$ for every $1 \leq i \leq k$, and also denote a sequence $S = (s_1, \ldots, s_n)$ over Δ by $S \in \Delta^n$.

A *relational k-dimension data stream* (k-dimension data stream, for short) on Δ is a sequence $S = (s_1, \ldots, s_n) \in \Delta^n$ of length n, where $s_i = (s_i[1], \ldots, s_i[k]) \in \Delta$ for each $1 \leq i \leq n$. Note that a sequence $(s_1[j], s_2[j], \ldots, s_n[j])$ is an element of ${\Sigma_j}^n$ of length n for each $1 \leq j \leq k$.

Example 1. S=(('She', *pronoun*, 3), ('loves', *verb*, 5), ('her', *pronoun-objective*, 3), ('baby', *noun-common*, 4)) is a 3-dimension data stream of length 4 on $\Delta = (\Sigma_s^*, \Sigma_c, \Sigma_o)$, where Σ_s is the ASCII character set, Σ_c is a categorical alphabet of word class, Σ_o is an ordered alphabet.

Time-series patterns over predicates: For a integer $k \geq 1$, let $\mathcal{X}$ be a set of k variables $\mathcal{X} = \{X_i \mid i = 1, \ldots, k\}$ on elements of Δ, namely X_i is a variable on Σ_i. An *atomic formula* is the form of "X_i *op* C" for each i, where $C \in \Sigma_i$ is a constant and *op* is a relational operation on Σ_i. The constant and the operation vary as the type of alphabet Σ_i. Concretely,

- C is an expression for some kind of string pattern and $op \in \{=_s\}$, indicating some kind of pattern matching, for textual data,

– C is a concept of Σ_i and $op \in \{\succeq\}$ for categorical data, and
– C is a value of Σ_i and $op \in \{\leq, \geq, <, >, =, \neq\}$ for ordered or numerical data.

Then, we recursively define *predicates* on Σ_i for each i as follows: (i) An atomic formula is a predicate. (ii) If ϕ and ψ are predicates, $\neg\phi$, $\phi \wedge \psi$ and $\phi \vee \psi$ are predicates. (iii) Only expressions generated by the above rules are the predicates. The *size* of a predicate ϕ, denoted by $size(\phi)$, is defined as the total number of occurrences of operators, variables, and constant in ϕ. We define the *truth-value* $\phi(s) \in \{1, 0\}$ of predicate on an element $s \in \Sigma_i$ for each i in a similar way in the predicate logic. An atomic formula $\phi'(s) = 1$ if X_i matches C, otherwise 0, for textual or categorical data, and $\phi'(s) = 1$ if s satisfies the equality or the inequalities over $\mathbb{R}$ or $\mathbb{N}$, otherwise 0, for numerical or ordered data. We denote by $\mathcal{P}_i = \mathcal{P}(X_i)$ the set of all predicates on Σ_i.

A *serial pattern* (*pattern*, for short) of length $m \geq 0$ on Δ is a k-term set of $P = (P_1, P_2, \ldots, P_k)$, where each $P_j = \phi_j[1] \cdot \phi_j[2] \cdots \phi_j[m] \in {\mathcal{P}_j}^m$ for each $1 \leq j \leq k$. We call $\phi_j[i] \in \mathcal{P}_i$ the i-th predicate of j-th pattern, and also call i the *index*. We define the *index set* of P_j by $Idx(P_j) = \{1, \ldots, m\}$, the *length* of P_i by $|P_i| = m$, and the *size* of P_i by the total size $size(P_i) = size(\phi_i[1]) + \cdots + size(\phi_i[m])$. We also define the *length* of P by $|P| = m$, and the *size* of P by the sum of $size(P_i)$.

Let $S = (s_1, \ldots, s_n) \in \Delta^n$ be a stream of length n. We say that the pattern P *occurs at position* p *in* S if $\phi_i[j](s_{p+j-1}) = 1$ for every $1 \leq j \leq m$ and every $1 \leq i \leq k$. Then, p is called an *occurrence position* of P on S. We denote by $Occ(P, S) \subseteq \{1, \ldots, n\}$ the set of all occurrences of P on S.

The pattern matching problem: Now, we state our problem in this paper as follows.

The *pattern matching problem over k-dimension streams* is, given a data stream $S = (s_1, \ldots, s_n) \in \Delta^n$ of length n and a serial pattern $P = (P_1, \ldots, P_k) \in (\mathcal{P}_1, \ldots, \mathcal{P}_k)$ of length m, to find the set of all occurrence positions $Occ(P, S)$ of P on S.

Example 2. Let $P = (P_1, P_2, P_3)$ be a pattern on $\Delta = (\Sigma_s^*, \Sigma_c, \Sigma_o)$ for Example 1. When $P_1 = (X_1 =_s$ 'She' $\vee\, X_1 =_s$ 'He'$) \cdot (X_1 =_s$ *any-string*$) \cdot (X_1 =_s$ *any-string*$) \cdot (X_1 =_s$ *any-string*$)$, $P_2 = (X_2 \succeq noun) \cdot (X_2 \succeq verb) \cdot (X_2 \succeq noun) \cdot (X_2 \succeq any{-}words)$, and $P_3 = (X_3 > 0) \cdot (X_3 > 0) \cdot (X_3 \geq 1) \cdot (X_3 \geq 4 \wedge X_3 < 10)$, the pattern P matches the sequence S in Example 1.

Bit operations: In this paper, we assume as a computation model the RAM (random access machine) model with arithmetic and bit operations on integer registers of bit-width $w = 32 \sim 64$ [20,8]. For *bitmasks* $X = b_m \cdots b_1 \in \{0, 1\}^m$ of length $m \geq 0$, we assume bit-operations such as *bitwise-and* $\&$, *bitwise-or* $|$ and *bitwise-not* $\sim$, the n-bit *left shift* $<< n$. We denote by 0^m the bitmask consisting only of 0 bits and by $Bit(S) \in \{0, 1\}^m$ the bit-mask representation of a subset S of $\{0, \ldots, m-1\}$. For the bitmasks of width m, the arithmetic and bit operations takes $O(1)$ time if $m \leq w$ and takes $O(\lceil \frac{m}{w} \rceil)$ time if $m > w$.

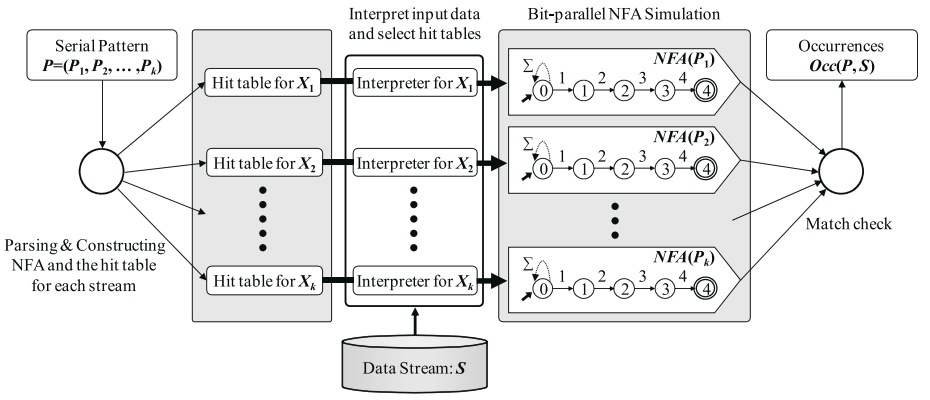


Fig. 2. The framework of our pattern matching algorithm over k-dimension data streams

3 The Framework Based on BPS

In this section, We present a framework for serial pattern matching over k-dimension data streams.

3.1 Outline of the Framework

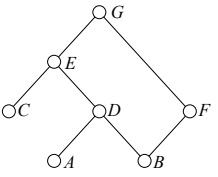
First of all, given a serial pattern $P = (P_1, \ldots, P_k)$, the matching machine parses it and constructs k-set of NFAs and the bitmask informations, which is used at the NFA simulation step and is stored as the predicate hit tables. The scanning processing phase consists of four steps: (1) For each i-th pattern, at j-th position, an element $s_i[j]$ $(1 \leq j \leq k)$ of each data stream is inputted to the machine, and it is processed on the fly by proper algorithms to interpret the element $s_i[j]$. (2) Each bitmask for $s_i[j]$ is decided for each serial pattern P_i. (3) The machine simulates k-BPS algorithms [7] in parallel by using the bitmasks. (4) Finally, it checks if all the NFAs reach their final states. Report if there is an occurrence at the position j, and then repeat these steps. In Fig. 2, we show the outline of our framework.

In what follows, we concentrate how to construct the predicate hit tables and how to simulate the NFAs by bit-parallel technique over data streams.

3.2 Predicate Hit Table

A key of BPS is a mechanism called the *predicate hit table* (*hit table*, for short), which is crucial to obtain the time complexity strictly below $O(mn)$ time. For simplicity, we will discuss here just a single data stream and its serial pattern matching, namely $\Delta = (\Sigma) \in \{\Sigma_s{}^*, \Sigma_c, \Sigma_o, \Sigma_n\}$.

Let $P = \phi[1] \cdots \phi[m] \in \mathcal{P}^*$ be a serial pattern with index set $Idx(P) = \{1, \ldots, m\}$. The predicate hit table for P implements a function $\eta : \Sigma \to 2^{Idx(P)}$ that receives an input value x and returns the set $\eta(x) = \{i \in Idx(P) \mid \phi[i](x) = 1\}$



$P =$	B	A	F	A	E	
x	$\eta(x)$					Bitmask
A		√		√	√	11010
B	√		√		√	10101
C					√	10000
D					√	10000
E					√	10000
F			√			00100
G						00000

Fig. 3. An example of a categorical alphabet and the predicate hit table for categorical serial pattern $P = (X \succeq \mathtt{B})(X \succeq \mathtt{A})(X \succeq \mathtt{F})(X \succeq \mathtt{A})(X \succeq \mathtt{E})$. The symbol $\surd$ indicates that $i \in \eta(x)$ for $x \in \{\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}, \mathtt{E}, \mathtt{F}, \mathtt{G}\}$.

of all indices such that the corresponding predicates are satisfied on x. In the algorithm, we encode the answer $\eta(x)$ by a bitmask $Bit(\eta(x)) \in \{0,1\}^m$ of width m for the succeeding processing. A straightforward method takes $O(||P||)$ time to compute $\eta(x)$ in running time for every $x \in \Delta$ by testing all the predicates sequentially. However, we can efficiently computes $\eta(x)$ using appropriate preprocessing as follows.

Hit table for categorical data stream: Recall that c matches d if $c \succeq d$ for any concepts $c, d \in \Sigma_c$. Then, the set $\eta(x) = \{i \in Idx(P) \mid \phi[i] \succeq x\}$ in this case. A categorical alphabet is represented as a *Hasse diagram*. A Hasse diagram for $(\Sigma_c, \succeq)$ can be viewed as a direct acyclic graph (DAG). For example, given a serial pattern $P = (X \succeq \mathtt{B})(X \succeq \mathtt{A})(X \succeq \mathtt{F})(X \succeq \mathtt{A})(X \succeq \mathtt{E})$ for $(\{\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}, \mathtt{E}, \mathtt{F}, \mathtt{G}\}, \succeq)$ and $\succeq$ is given as the Hasse diagram of the left side of Fig. 3, the predicate hit tables $\eta(x)$ becomes as the right table in Fig. 3. From the observation of relationships on the Hasse diagram, we have the following lemmas which are translations of the lemmas in [9].

Lemma 1 (modified version of Kida *et al.*[9]). *Let Σ_c be a categorical alphabet and P be a serial pattern on Σ_c. For any $x \in \Sigma_c$, we obtain the formula:*

$$\eta(x) = \bigcup_{y \in Upb(x)} \theta(y),$$

where $Upb(x) = \{z \in \Sigma_c \mid z \succeq x\}$ and $\theta(x) = \{i \in Idx(P) \mid \phi[i] = x\}$.

Lemma 2 (modified version of Kida *et al.*[9]). *Let Σ_c be a categorical alphabet and P be a serial pattern on Σ_c. For any $x \in \Sigma_c$, we obtain the formula:*

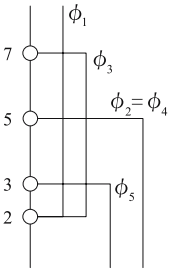
$$\eta(x) = \theta(x) \cup \bigcup_{y \in Par(x)} \eta(y),$$

where $Par(x)$ is the set of all parent nodes of x in the Hasse diagram.

We can compute $Bit(\eta(x))$ recursively from the above lemmas, by traversing the DAG representing $(\Sigma_c, \succeq)$ for every $x \in \Sigma_c$.

x	$\eta(x)$	Bitmask
8	1	00001
7	1	00001
6	13	00101
5	13	00101
4	1234	01111
3	1234	01111
2	245	11010
1	245	11010

(a) Ordered values



(b) Interval representation

x	$\eta(x)$	Bitmask
$x > 7$	1	00001
$x = 7$	1	00001
$5 < x < 7$	13	00101
$x = 5$	13	00101
$3 < x < 5$	1234	01111
$x = 3$	1234	01111
$2 < x < 3$	12345	11111
$x = 2$	245	11010
$x < 2$	245	11010

(c) Numeric values

Fig. 4. Predicate hit tables

Lemma 3 (modified version of Kida *et al.*[9]). *For a categorical alphabet Σ_c, let $\mathcal{H}$ be the DAG for $(\Sigma_c, \succeq)$ and $P = P[1:m] \in {\Sigma_c}^m$ be a categorical serial pattern. Then, the bitmasks $Bit(\eta(x))$ for all $x \in \Sigma_c$ can be computed in $O(m + \frac{m|\mathcal{H}|}{w})$ time instead of $O(m + m|\mathcal{H}|)$ time in naive algorithm, where $|\mathcal{H}|$ is the number of vertices plus the number of edges of $\mathcal{H}$.*

Hit table for ordered data stream: Recall that an ordered domain is a set $\Delta = \{1, \ldots, c\} \subseteq \mathbb{N}$ of small, say 16-bit, integers for some c. Suppose that the data range $c = |\Delta|$ is constant. In this case, we represent a predicate hit table simply by a table of its values as shown in (a) of Fig. 4. Then, we precompute the bitmasks $Bit(\eta(x))$ for all x in advance. In Fig. 5, we show the procedure HT_build for building the lookup table B. The time and space for the preprocessing is $O(c||P||)$ and $O(\frac{cm}{w})$, respectively. In running time, each table look-up by a value x is done by a random access to B by index x in $O(1)$ time.

Hit table for numerical data stream: Since a numeric domain $\Delta \subseteq \mathbb{R}$ is an infinite or large set, we cannot directly store the predicate hit table in main memory. In this case, however, we can show that the whole domain Δ can be

---

procedure HT_build(Σ_o, P)
global: An array $B = (B[1], \ldots, B[c])$ of $c = |\Sigma_o|$ bitmasks;
input: Ordered type $\Sigma_o = \{1, \cdots, c\}$ and a pattern $P = \phi[1] \cdots \phi[m]$ on Σ_o;
output: An array B;

```
1: for x ∈ {1, ··· , c} do
2:     B[x] ← 0^m;
3:     for i ∈ {1, ··· , m − 1} do
4:         if φ[i](x) = 1 then B[x] ← B[x] | 1 << i;
```

procedure HT_lookup(value x)≡**return** $B[v] \in \{0,1\}^m$;

Fig. 5. The procedure for preprocessing and looking up predicate hit tables for pattern P_1

partitioned into a collection $\Delta = \Delta_1 \cup \cdots \cup \Delta_k$ of equivalence classes such that two values $x, y \in \Delta$ are equivalent each other if and only if $\eta(x) = \eta(y)$ holds. Furthermore, each equivalence class can be represented uniquely as the union of open intervals and end points specified by the set of constants appearing in P (See (b) of Fig. 4).

From these observations, we encode a predicate hit table η by a table as in (c) of Fig. 4, whose keys are endpoints and the intervals $I \subseteq \Delta$ defined by the inequalities of pattern P and the values are $Bit(\eta(x))$ for the representative (or any) value $x \in I$. Since P contains at most $M = ||P||$ atomic formulas and so are the end points, the hit table contains $O(M)$ entries. Therefore, the time and space for the preprocessing is $O(\frac{mM^2}{w})$ and $O(\frac{mM}{w})$, respectively. In running time, each table look-up is done by a binary search on the keys in $O(\frac{m \log M}{w})$ time.

Hit table for textual data stream: In this case, $\Delta \subseteq \Sigma_s^*$. If $size(\phi[i]) = 1$ for every i and $=_s$ means exact matching, we can use the Aho-Corasick pattern matching machine [17] as the interpreter at the first and the second step of the scanning phase. That is, $\eta(x)$ is computed as the set of indices of words which appear in the predicates. Let $w_1, w_2, \ldots, w_m$ be strings on Δ, and $w_i = C_i$ for every i, where C_i is a constant of i-th predicate $\phi[i]$. Running the Aho-Corasick pattern matching machine for the set $\{w_1, \ldots, w_m\}$, we can know the set of indices for x as its outputs. On the other hand, If $size(\phi[i]) > 1$ for some i or $=_s$ means more complex matching like regular expression matching or approximate string matching, we might use other sophisticated algorithms. However, we can also know the set of indices in a similar way of the above case if we can know which expression is matched to x.

Although we omit the detail, in both cases, we can compute in advance the bitmask $Bit(\eta(x))$ for every $x \in \Delta$, and then store them in the array $B[1..|\Delta|]$ such that $B[x] = Bit(\eta(x))$ for every $x \in \Delta$, and attach them to the pattern matching machine as its outputs.

3.3 Simulation of NFA

NFA for serial patterns: We describe construction of NFA from a given pattern. In general, given a pattern P in the form of a regular expression, the NFA for pattern matching by P, denoted by $NFA(P)$, is the NFA accepting the language $\Delta^* L(P)$, where $L(P) \subseteq \Delta^*$ is the language accepted by P as regular expression.

For serial patterns $P = \phi[1] \cdots \phi[m] \in \mathcal{P}$, a straightforward approach is to build the NFA that accepts the language of substrings of values $\Delta^* \times \Gamma_1 \times \cdots \times \Gamma_m$ that match P, where $\Gamma_i = \{x \in \Delta \mid \phi[i](x) = 1\}$ is the set of all input values that satisfies the i-th predicate $\phi[i]$. However, since Δ and Γ_i's may be infinite or large, the straightforward implementation is not realistic.

Instead, we build the NFA $NFA(P)$ as follows. Firstly, we represent each set Γ_i of values simply by the index i for every $1 \leq i \leq m$. Thus, the alphabet of $NFA(P)$ is $\Sigma_P = Idx(P) \cup \{0\} = \{0, 1, \ldots, m\}$, where we assume that the index 0 corresponds to a special predicate **true** that is always satisfied by every value.

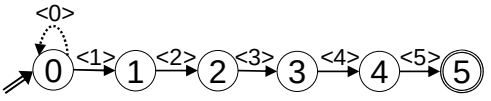


Fig. 6. NFA $NFA(P_1)$ for the serial pattern P_1

procedure NFA_run
1: $I \leftarrow 0^{m-1}1$; {The set of initial states}
2: $F \leftarrow 10^{m-1}$; {The set of accepting states}
3: $D \leftarrow I$;
4: **for** $i = 1, \ldots, n$ **do**
5: $D \leftarrow ((D << 1) \mid I) \& B[s_i]$;
$\{B[s_i]$ is the bit-mask for the next input value $s_i \in \Delta\}$
6: **if** $(D \& F) \neq 0^m$ **then**
7: Output occurrence $occ = i - m + 1$;

Fig. 7. Procedure NFA_run

For clarity, we often write $\langle i \rangle$ for index i if it is treated as a letter. Then, the target NFA is an NFA with a chain-like linear structure as the NFA in Fig. 6, which is the NFA for the pattern $P_1 = \phi[1] \cdots \phi[5]$.

More precisely, the target NFA is $NFA(P) = (Q_P, \Sigma_P, \delta_P, I_P, F_P)$, where (i) $\Sigma_P = Q_P = Idx(P) \cup \{0\} = \{0, 1, \ldots, m\}$ the alphabet and the set of states, (ii) $\delta_P = \{(i-1, \langle i \rangle, i) \mid 1 \leq i \leq m\} \cup \{(0, \langle 0 \rangle, 0)\} \subseteq Q_P \times \Sigma_P \times Q_P$ is the state transition relation consisting of the set of forward transitions $(i-1, \langle i \rangle, i) \in \delta_P$ from state $i-1$ to state i by letter $\langle i \rangle \in \Sigma_P$ for every $1 \leq i \leq m$ and a self loop $(0, \langle 0 \rangle, 0) \in \delta_P$ by letter $\langle 0 \rangle$, and (iii) $0 \in I_P$ and $m \in F_P$ are the unique initial state and the unique accepting state, respectively.

Fast simulation by bit-parallel method: In the running phase, BPS simulates NFA $NFP(P)$ by scanning an input data stream S in the left to right manner at once. Although a straightforward implementation of NFA simulation takes $O(mn)$ time in total to make the pattern matching, we can simulate in $O(\frac{mn}{w})$ time by using bit-parallel method as follows. For simplicity, we give a description in the case of an ordered domain. Let $\Delta = \{1, \ldots, c\}$ $(c \geq 0)$ be an ordered alphabet and $\eta(x)$ be the predicate lookup table built in preprocessing phase. Let $B[1..c]$ be the array of bitmasks obtained by η such that the α-th entry $B[\alpha] = Bit(\eta(\alpha))$ is the bitmask for every value $x \in \Delta$ $(1 \leq \alpha \leq c$. In Fig. 7, we show a procedure for simulating $NFA(P)$ for serial patterns by bit-parallel method.

In Fig. 8, we show the overall algorithm BPS_tab for the time-series pattern matching on an ordered alphabet. From the above discussion and Theorem 1 of [7], we have the following result:

Theorem 1. *Let Δ be a k-term class of alphabets, and assume that the interpretation of inputs can be done in $O(1)$ time for each. Then, the algorithm multi-dimensionBPS in Fig. 8 finds the set of all occurrences of a given pattern*

algorithm multi-dimensionBPS(Δ, P, S)
*i*nput: A k-term class of alphabets $\Delta = (\Sigma_1, \ldots, \Sigma_k)$, a pattern $P = (P_1, \ldots, P_k) \in ({\mathcal{P}_1}^m, \ldots, {\mathcal{P}_k}^m)$, and a stream $S = (s_1, \ldots, s_n) \in \Delta^n$. *o*utput: The set of occurrences $Occ(P, S)$ of P on S.

1: {Preprocessing phase:}
2: Build k arrays for the predicate hit table η_i of P_i by HT_build(Σ_i, P_i) for every i-th pattern P_i $(1 \leq i \leq k)$;
3: {Scanning phase:}
4: **for each** position j **do**
5: Make state transition of each NFA $NFA(P_i)$ for s_j, by using bit-parallel method with the array by HT_lookup and NFA_run;
6: Check if all NFAs reach its final state, and then report the occurrence.

Fig. 8. The overall algorithm of our framework

P of length m in an input stream $S \in \Delta^n$ of length n in $O(\frac{kmn}{w} \log m)$ time using preprocessing $O(km|\Sigma|)$ time and space $O(\frac{km|\Sigma|}{w})$ words, where w is the bit-width of registers.

This time complexity is properly superior than the time complexity $O(kmn \log m)$ of one based on the previous algorithms.

4 Conclusion

In this paper, we presented a flexible unified framework for time-series pattern matching over multi-dimension data streams. The proposed framework can deal with textual, categorical, ordered and numerical type data streams. Each pattern for each data stream can be given as a sequence of predicates which specify the set of elements on the stream. It enables us to search with a rather complex pattern over data streams. We omitted in this paper, it is easy to extend our framework into approximate matching (m of k-dimension match, Hamming distance, Levenshtein distance, and so on), by using techniques in [8].

References

1. Morikawa, H., Asai, T., Arimura, H.: Efficient xpath processing for semi-structured data streams. In: Proc. DBWS 2003, IPSJ-IEICE (2003) 28 (in Japanese)
2. Motwani, R., Widom, J., et al.: Query processing, approximation, and resource management in a data stream management system. In: Proc. CIDR 2003 (2003)
3. Stonebraker, M., Cetintemel, U.: One size fits all: An idea whose time has come and gone. In: Proc. ICDE 2005, pp. 2–11 (2005)
4. Harada, L.: Complex temporal patterns detection over continuous data streams. In: Manolopoulos, Y., Návrat, P. (eds.) ADBIS 2002. LNCS, vol. 2435, pp. 401–414. Springer, Heidelberg (2002)

5. Harada, L.: Pattern matching over multi-attribute data streams. In: Laender, A.H.F., Oliveira, A.L. (eds.) SPIRE 2002. LNCS, vol. 2476, pp. 187–193. Springer, Heidelberg (2002)
6. Sadri, R., Zaniolo, C., Zarkesh, A.M., Adibi, J.: Optimization of sequence queries in database systems. In: Proc. PODS 2001. ACM, New York (2001)
7. Saito, T., Kida, T., Arimura, H.: An efficient algorithm for complex pattern matching over continuous data streams based on bit-parallel method. In: Proc. of The Third IEEE International Workshop on Databases for Next-Generation Researchers (SWOD 2007), pp. 13–18 (April 2007)
8. Navarro, G., Raffinot, M.: Flexible Pattern Matching in Strings: Practical on-line search algorithms for texts and biological sequences. Cambridge University Press, Cambridge (2002)
9. Kida, T., Arimura, H.: Pattern matching with taxonomic information. In: Myaeng, S.-H., Zhou, M., Wong, K.-F., Zhang, H.-J. (eds.) AIRS 2004. LNCS, vol. 3411, pp. 265–268. Springer, Heidelberg (2005)
10. Knuth, D.E., Morris, J.H., Pratt, V.R.: Fast pattern matching in strings. SIAM J. Comput. 6(2), 323–350 (1977)
11. Boyer, R.S., Moore, J.S.: A fast string searching algorithm. Comm. ACM 20(10), 62–72 (1977)
12. Mäkinen, V., Navarro, G., Ukkonen, E.: Matching numeric strings under noise. In: Proceedings of the 8th Prague Stringology Conference (PSC 2003), pp. 99–110 (2003)
13. Crochemore, M., Iliopoulos, C.S., Navarro, G., Pinzon, Y.J.: A bit-parallel suffix automaton approach for (δ, γ)-matching in music retrieval. In: Nascimento, M.A., de Moura, E.S., Oliveira, A.L. (eds.) SPIRE 2003. LNCS, vol. 2857, pp. 211–223. Springer, Heidelberg (2003)
14. Crochemorea, M., Iliopoulos, C.S., Navarro, G., Pinzon, Y.J., Salingerc, A.: Bit-parallel (δ,γ)-matching and suffix automata. Journal of Discrete Algorithms 3(2-4), 198–214 (June 2005); (Combinatorial Pattern Matching (CPM) Special Issue)
15. Fredriksson, K., Mäinen, V., Navarro, G.: Flexible music retrieval in sublinear time. International Journal of Foundations of Computer Science (IJFCS) 17(6), 1345–1364 (2006)
16. Hyrro, H., Takaba, J., Shinohara, A., Takeda, M.: On bit-parallel processing of multi-byte text. In: Myaeng, S.-H., Zhou, M., Wong, K.-F., Zhang, H.-J. (eds.) AIRS 2004. LNCS, vol. 3411, pp. 289–300. Springer, Heidelberg (2005)
17. Aho, A.V., Corasick, M.J.: Efficient string matching: An aid to bibliographic search. Comm. ACM 18(6), 333–340 (1975)
18. Arikawa, S., Shinohara, T., Takeya, S.: Sigma: A text database management system. Berliners Informatik Tag 5, 72–81 (1989)
19. Matsumoto, Y., Asahara, M., Kawabe, K., Takahashi, Y., Tono, Y., Ohtani, A., Morita, T.: Chaki: An annotated corpora management and search system. In: Proceedings from the Corpus Linguistics Conference Series, vol. 1, Corpus Linguistics (July 2005), `http://chasen.naist.jp/hiki/ChaKi/`
20. Aho, A.V., Hopcroft, J.E., Ullman, J.D.: The Design and Analysis of Computer Algorithms (1974)

An Adaptive Algorithm for Splitting Large Sets of Strings and Its Application to Efficient External Sorting

Tatsuya Asai[1], Seishi Okamoto[1], and Hiroki Arimura[2]

[1] Fujitsu Laboratories Ltd, Kawasaki 211–8588, Japan
asai.tatsuya@jp.fujitsu.com, seishi@labs.fujitsu.com
[2] Hokkaido University, Sapporo 060–0814, Japan
arim@ist.hokudai.ac.jp

Abstract. In this paper, we study the problem of sorting a large collection of strings in external memory. Based on adaptive construction of a summary data structure, called *adaptive synopsis trie*, we present a practical string sorting algorithm DistStrSort, which is suitable for sorting string collections of large size in external memory, and also suitable for more complex string processing problems in text and semi-structured databases such as counting, aggregation, and statistics. Case analyses of the algorithm and experiments on real datasets show the efficiency of our algorithm in realistic setting.

1 Introduction

Sorting strings is a fundamental task of string processing, which is not only sorting the objects in associated ordering, but also used as a basis of more sophisticated processing such as grouping, counting, and aggregation. With a recent emergence of massive amount of *semi-structured data* [2], such as plain texts, Web pages, or XML documents, it is often the case that we would like to sort a large collection of strings that do not fit into main memory. Moreover, there are potential demands for flexible methods tailored for semi-structured data beyond basic sorting facilities such as statistics computation, counting, and aggregation over various objects including texts, data, codes, and streams [1,15].

In this paper, we study efficient methods for sorting strings in external memory. We present a new approach to sorting a large collection of strings in external memory. The main idea of our method is to generalize distribution sort algorithms for collections of strings by the extensive use of *trie* data structure. To implement this idea, we introduce a synopsis data structure for strings, called a *synopsis trie*, for approximate distribution, and develop an adaptive construction algorithm for synopsis tries for a large collection of strings with the idea of adaptive growth.

Based on this technique, we present an external string sorting algorithm DistStrSort. In Fig. 1, we show an architecture of our external sorting. Let $S = \{s_1, \ldots, s_m\} \subseteq \Sigma^*$ be an input string list over an alphabet Σ. This algorithm first adaptively constructs a small synopsis trie as a model of string

S. Chawla et al. (Eds.): PAKDD 2008 Workshops, LNAI 5433, pp. 13–25, 2009.

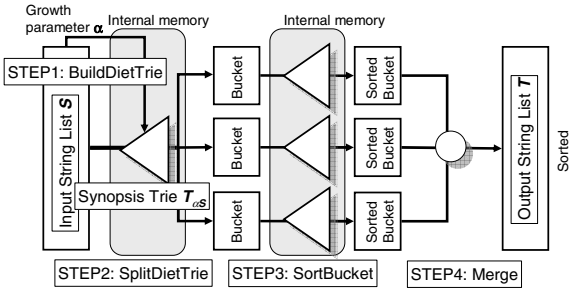


Fig. 1. The architecture of an external sorting algorithm DistStrSort using adaptive construction of a synopsis trie

distribution by a single scan of the input string list S, splits input strings into a collection of buckets using the synopsis trie as a splitter, then sorts the strings in each bucket in main memory, and finally merges the sorted buckets into an output collection.

By an analysis, we show that if both of the size of the synopsis trie and the maximum size of the buckets are small enough to fit into main memory of size M, then our algorithm DistStrSort correctly solves the external string sorting problem in $O(N)$ time within main memory M using $3N$ sequential read, N random write, and N sequential write in the external memory, where N is the total size of an input string list S. Thus, this algorithm of $O(N)$ time complexity is attractive compared with a popular merge sort-based external string sort algorithm of $O(N \log N)$ time complexity. Experiments on real datasets show the effectiveness of our algorithm.

As an advantage, this algorithm is suitable to implement complex data manipulation operations [11] such as group-by aggregation, statistics calculation, and functional join for texts.

1.1 Related Works

In what follows, we denote by $K = |S|$ and $N = ||S||$ be the cardinality and the total size of an input list $S = \{s_1, \ldots, s_m\}$.

On incore string sorting, a straightforward extension of quick sort algorithm solves the string sorting problem in $O(LK \log K)$ time in main memory, where L is the maximum length of the input strings and $N = O(LK)$. This algorithm is efficient if $L = O(1)$, but inefficient if L is large. Sinha and Zobel presented a trie-based string sorting algorithm, called the *burst sort*, and show that the algorithm outperforms many of previous string sorting algorithms on real datasets [14]. However, their algorithm is an internal sorting algorithm, and furthermore, the main aim of this paper is proposal of an adaptive strategy for realizing an efficient external string sorting. Although there are a number of studies on sorting the suffixes of a single input string [4,8,9,13], they are rather irrelevant to this work

since they are mainly incore algorithms and utilize the repetition structure of suffixes of a string.

On external string sorting, Arge *et al.* presented an algorithm that solves the string sorting problem in $O(K \log K + N)$ time in main memory [3]. The String B-tree is an efficient dynamic indexing data structure for strings [5]. This data structure also has almost optimal performance for external string sorting in theory, while the performance degenerates in practice because of many random access to external memory.

From the view of stream processing algorithms, the adaptive construction of a trie for a stream of letters has been studied since 90's in time-series modeling [7], data compression [10], and bioinformatics [12]. However, the application of adaptive trie construction to external sorting seems new and an interesting direction in text and semi-structured data processing. Recently, [15] showed the usefulness of distributed stream processing as a massive data application of new type.

1.2 Organization

This paper is organized as follows. In Section 2, we prepare basic notions and definitions on string sorting and related string processing problems. In Section 3, we present our external string sorting algorithm based on adaptive construction of a synopsis trie. In Section 4, we report experiments on real datasets, and finally, we conclude in Section 5.

2 Preliminaries

In this section, we give basic notions and definitions on external string sorting.

2.1 Strings

Let $\Sigma = \{A, B, A_1, A_2, \ldots\}$ be an *alphabet* of letters, with which a *total order* $\leq_\Sigma$ on Σ is associated. In particular, we assume that our alphabet is a set of integers in a small range, i.e., $\Sigma = \{0, \ldots, C-1\}$ for some integer $C \geq 0$, which is a constant or at most $O(N)$ for the total input size N. This is the case for ASCII alphabet $\{0, \ldots, 255\}$ or DNA alphabet $\{A, T, C, G\}$.

We denote by Σ^* the set of all finite strings on Σ and by ε the empty string. Let $s = a_1 \cdots a_n \in \Sigma^*$ be a string on Σ. For $1 \leq i \leq j \leq n$, we denote by $|s| = n$ the *length* of s, by $s[i] = a_i$ the *i-th letter*, and by $s[i..j] = a_i \cdots a_j$ the *substring* from i-th to j-th letters. If $s = pq$ for some $p, q \in \Sigma^*$ then we say that p is a *prefix* of s and denote $p \sqsubseteq s$. We define the lexicographic order $\leq_{\mathrm{lex}}$ on strings of Σ^* by extending the total order $\leq_\Sigma$ on letters in the standard way.

2.2 String Sorting Problem

For strings $s_1, \ldots, s_m \in \Sigma^*$, we distinguish an ordered list $(s_1, \ldots, s_m)$, an unordered list (or multiset) $\{\!\{s_1, \ldots, s_m\}\!\}$, and a set $\{s_1, \ldots, s_m\}$ (of unique strings) each other. The notations $\in$ and $=$ are defined accordingly. We use the intentional

notation $(s \mid P(s))$ or $\{s \mid P(s)\}$ for ordered lists and unordered sets, respectively. For lists of strings S_1, S_2, we denote by S_1S_2 ($S_1 \oplus S_2$, resp.) the concatenation of S_1, S_2 as *ordered lists* (as *unordered lists*, resp.).

An input to our algorithm is a *string list* of size $K \geq 0$, that is, just an ordered list $S = (s_1, \ldots, s_K) \in (\Sigma^*)^*$ of possibly identical strings, where $s_i \in \Sigma^*$ is a strings on Σ for every $1 \leq i \leq K$. We denote by $|S| = K$ the *cardinality* of S and $||S|| = N = \sum_{i=1}^{K} |s_i|$ the *total size* of S. We denote the set of all unique strings in S by $uniq(S) = \{\, s_i \mid i = 1, \ldots, K \,\}$. A string list $S = (s_1, \ldots, s_n)$ is *sorted* if $s_i \leq_{\text{lex}} s_{i+1}$ holds for every $i = 1, \ldots, n-1$.

Definition 1. The *string sorting problem* (STR-SORT) is, given a string list $S = (s_1, \ldots, s_K) \in (\Sigma^*)^*$, to compute the sorted list $\pi(S) = (s_{\pi(1)}, \ldots, s_{\pi(K)}) \in (\Sigma^*)^*$ for some permutation $\pi : \{1, \ldots, K\} \to \{1, \ldots, K\}$ of its indices such that $S_{\pi(1)} <_\Sigma \cdots <_\Sigma S_{\pi(K)}$.

We can extend our framework for more complicated string processing problems as follows.

Definition 2. The *string counting problem* (STR-COUNT) is, given a string list $S = (s_1, \ldots, s_K) \in (\Sigma^*)^*$, to compute the histogram $h : uniq(S) \to \mathsf{N}$ such that $h(s)$ is the number of occurrences of the unique string $s \in uniq(S)$.

Let $(\Delta, \circ)$ be a pair of a collection Δ of *values* and a corresponding associative operation $\circ : \Delta^2 \to \Delta$. For example, $(\mathsf{N}, +)$ and $(\mathsf{R}, \max)$ are examples of associative operation $(\Delta, \circ)$. A *string database* of size $K \geq 0$ over $dom(D) = \Delta$ is a list $D = (\,(s_i, v_i) \mid i = 1, \ldots, K\,) \in (\Sigma^* \times \Delta)^*$ of pairs of a string and a value. We denote by $uniq(D) = \{\, s_i \mid (s_i, v_i) \in D \,\}$ the set of unique strings in S. For $(\Delta, \circ)$, the *aggregate* on s w.r.t. $\circ$, denoted by $\mathtt{aggr}_\circ(s)$, is defined as $\mathtt{aggr}_\circ(s) = \circ\,(\, v \mid (s, v) \in D\,) = v_1 \circ \cdots \circ v_m$, where $(s_1, v_1), \ldots, (s_m, v_m) \in D$ is the list of all pairs in D with $s_i = s$.

Definition 3. The *string aggregation problem* w.r.t. associative operation $(\Delta, \circ)$ (STR-AGGREGATE$(\Delta, \circ)$) is, given a string database $S{=}(\,(s_i, v_i) \mid i{=}1, \ldots, K\,) \in (\Sigma^* \times \Delta)^*$, to compute the pair $(s, \mathtt{aggr}_\circ(s))$ for every unique string $s \in uniq(D)$.

The above string counting and string aggregation problems as well as string sorting problem can be efficiently solved by a modification of trie-based sorter. In this paper, we extend such a trie-based sorter for an external memory environment.

2.3 Model of Computation

In what follows, we denote by $K = |S|$ and $N = ||S||$ the cardinality and the total size of the input string list $S = (s_1, \ldots, s_m)$. We assume a naive model of computation such that a CPU has random access to a main memory of size $M \geq 0$ and sequential and random access to an external memory of unbounded size, where memory space is measured in the number of integers to be stored. Though the external memory can be partitioned into blocks of size $0 \leq B \leq M$,

we do not study a detailed analysis of I/O complexity. In this paper, we are particularly interested in the case that the input size does not fit into the main memory but not so large, namely, $N = O(M^2)$.

3 Our Algorithm

3.1 Outline of the Main Algorithm

Fig. 1 shows the architecture of our external string sorting algorithm DistStrSort using an adaptive construction of synopsis trie for splitting, which is a variant of distribution sort for integer lists. In Fig. 2, we show an outline of DistStrSort.

The key of the algorithm is Step 2 that splits the input list S into buckets $S_1, \ldots, S_d$ of at most B by computing an ordered partition defined as follows. For string lists $S_1, S_2 \in (\Sigma^*)^*$, S_1 *precedes* S_2, denoted by $S_1 \prec S_2$, if $s_1 <_{\text{lex}} s_2$ for every combination of s_1 in S_1 and s_2 in S_2.

Then, an *ordered partition* of a string list S is a sequence of string lists $S_1, \ldots, S_d$ such that

(i) the sequence is a partition of S as unordered list, i.e., $S = S_1 \oplus \cdots \oplus S_d$,
(ii) S_i precedes S_{i+1}, i.e., $S_i \preceq S_{i+1}$, for every $i = 1, \ldots, n-1$, and

Furthermore, an ordered partition of a string list S satisfies *maximum bucket size* B if

(iii) $|S_i| \leq B$ holds for every $i = 1, \ldots, n$.

It is easy to see that if we can compute an ordered partition of an input string list S of maximum size $B \leq M$ in main memory of size M and if we

---

Algorithm: DistStrSort
Input: A string list $S = (s_1, \ldots, s_K) \in (\Sigma^*)^*$ and a maximal bucket size $0 \leq B \leq M$.
Output: The sorted string list $T \in (\Sigma^*)^*$.

1. Determine a growth parameter $\alpha \geq 0$. Build an adaptive synopsis trie $\mathcal{T}_{\alpha,S}$ for S in main memory with the growth parameter α and a bucket size B. (BuildDietTrie in Fig. 4)
2. Split the input list S in external memory into partition of buckets $S_1, \ldots, S_d$ of size at most B ($d \geq 0$) by using $\mathcal{T}_{\alpha,S}$, where $S_i \preceq S_{i+1}$ for every $i = 1, \ldots, n-1$. (BuildDietTrie in Fig. 5)
3. For every $i = 1, \ldots, K$, sort the bucket B_i by arbitrary internal sorting algorithm and write back to B_i in external memory. (BuildDietTrie in Fig. 6)
4. Return the concatenation $T = B_1 \cdots B_K$ of sorted buckets.

Fig. 2. An outline of the external string sorting algorithm with adaptive splitting DistStrSort

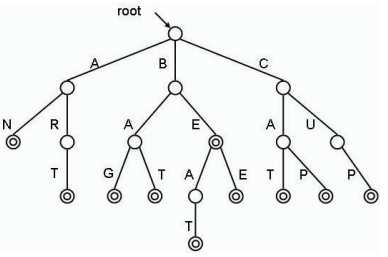


Fig. 3. A trie

can sort bucket of size B in $O(B)$ main memory then the algorithm DistStr-Sort in Fig. 2 correctly solves the string sorting problem in main memory M (Theorem 7).

However, it is not easy to compute a good partition with maximum block size B using only a limited main memory for a large input data S that does not fit into main memory. Hence, we take an approach of computing an approximate answer using adaptive computation in the following sections.

3.2 A Synopsis Trie

A *synopsis trie* for a string list $S = (s_1, \ldots, s_K) \in (\Sigma^*)^*$ is the trie data structure $\mathcal{T}$ for a subset of prefixes of strings in $uniq(S)$ defined as follows.

A *trie* for a set S of strings on Σ [6] is a rooted tree $\mathcal{T}$ whose edges are labeled with a letter in Σ. In Fig. 3, we show an example of a trie for a set $S = \{\texttt{AN}, \texttt{ART}, \texttt{BAG}, \texttt{BAT}, \texttt{BEAT}, \texttt{BEE}, \texttt{CAT}, \texttt{CAP}, \texttt{CUP}\}$ of strings.

We denote by $\mathcal{T}$ the set of all vertices of $\mathcal{T}$ and by $L(\mathcal{T}) \subseteq \mathcal{T}$ the set of its leaves. Each vertex $v \in \mathcal{T}$ represents the string $str(v) \in \Sigma^*$, called the *path string*, obtained by concatenating the labels on the unique path from the root `root` to v in $\mathcal{T}$. Let $Str(\mathcal{T}) = \{\, str(\ell) \mid \ell \in L(\mathcal{T}) \,\}$ be the set of strings represented by the leaves of $\mathcal{T}$. The vertices of a trie $\mathcal{T}$ are often labeled with some additional information.

If a trie $\mathcal{T}$ satisfies that $Str(\mathcal{T}) = S$ then $\mathcal{T}$ is said to be a *trie for* S. The trie $\mathcal{T}$ for a string list S can be constructed in $O(N)$ time and $O(N)$ space in the total input size $N = ||S||$ for constant Σ [6].

Then, a *synopsis trie* for S is a trie $\mathcal{T}$ for S which satisfies that for every string $s \in uniq(S)$, some leaf $\ell \in \mathcal{T}$ represents a prefix of s, i.e., $str(\ell) \sqsubseteq s$. Note that a synopsis trie for S is not unique and the empty trie $\mathcal{T} = \{\texttt{root}\}$ is a trivial one for any set S.

Given a string list S, a synopsis trie $\mathcal{T}$ for S can store a subset of strings in S. For every leaf $\ell \in L(\mathcal{T})$, we define the sublist of strings in S belongs to ℓ by $\texttt{list}(\ell, S) = (\, s \in S \mid str(\ell) \sqsubseteq s \,)$ and the count of the vertex by $\texttt{count}(\ell, S) = |\texttt{list}(\ell, S)|$. The total space required to implement a trie for S is $O(||S||)$ even if we include the information on `count` and `list`. On the incore computation by a trie, we have the following lemma.

Lemma 1. *We can solve the string sorting, the string counting, and the string aggregation problems for an input string list S in $O(N)$ time using main memory of size $O(N)$, where $N = ||S||$.*

We will extensively use this trie data structure in the following subsections in order to implement the computation at Step 1, Step 2, and Step 3 of the main sorting algorithm in Fig. 2.

3.3 STEP1: Adaptive Construction of a Synopsis Trie

In Fig. 4, we show the algorithm BuildDietTrie for constructing an adaptive synopsis trie.

Although the empty trie $\mathcal{T} = \{\texttt{root}\}$ obviously satisfies the above condition, it is useless for computing a good ordered partition. Instead, the goal is adaptive construction of a synopsis trie $\mathcal{T}$ for S satisfying the following conditions:

(i) $\mathcal{T}$ fits into the main memory of size M, i.e., $\texttt{size}(\mathcal{T}) \leq M$.
(ii) For an input string list S, $\texttt{count}(\ell, S) \leq B$ for a parameter $B \leq M$

The adaptive construction is done as follows. For an alphabet Σ=$\{c_0, \ldots, c_{s-1}\}$, each state v of the automaton is implemented as the list $(\texttt{goto}(v, 0), \ldots, \texttt{goto}(v, s-1))$ of s pointers. For an alphabet of constant size, this is implemented as an array of pointers. At the creation of a new state v, these pointers are set to $NULL$ and a counter $\texttt{count}(v)$ is set to 1.

When we construct a synopsis trie $\mathcal{T}$, the algorithm uses a positive integer $\alpha > 0$, called the *threshold for growing*. This threshold represents a minimum frequency for extending a new state to the current edge.

Algorithm: BuildDietTrie
Input: A string list $S = (s_1, \ldots, s_K)$, a maximum bucket size $M \geq B \geq 0$, and a growth parameter $\alpha > 0$.
Output: A synopsis trie $\mathcal{T}_{\alpha,S}$ for S.

1. $\mathcal{T} := \{\texttt{root}\}$;
2. For each $i := 1, \ldots, K$, do the following:
   (a) Read the next string $s = s_i$ from the external disk; Let $x := \texttt{root}$;
   (b) For each $j := 1, \ldots, |s|$, do:
   - $y := \texttt{goto}(x, s[j])$; (Trace the edge c_j.)
   - If $y = \perp$ holds, then:
     - If $\texttt{count}(x) \geq \alpha$ holds, then:
       Create a new state y; $T := T \cup \{y\}$;
       $\texttt{count}(y) := 1$; $\texttt{goto}(x, c_j) := y$;
     - Else:
       break the inner for-loop and **goto** the step 2(a);
   - Else, $x := y$ and
     $\texttt{count}(y) := \texttt{count}(y) + 1$;
3. Return $\mathcal{T}$;

Fig. 4. An algorithm for constructing a synopsis trie

The algorithm firstly initializes a trie $\mathcal{T}$ as the trie $\mathcal{T} = \{\texttt{root}\}$ consisting of the root node only. When the algorithm inserts a string $s \in S$ into $\mathcal{T}$, it traces the corresponding edge to s starting from $x = \texttt{root}$ by `goto` pointers. Each state x in the synopsis trie $\mathcal{T}$ has an integer $\texttt{count}(x)$ incremented when a string s reaches to x. These counters represent approximate occurrences of suffixes of input strings.

When there are no states the current state x transfers to by the next symbol $s[j]$, the synopsis trie tries to generate a new state. However, the algorithm does not permit the trie to extend a new edge unless the counter $\texttt{count}(x)$ of x is more than the given threshold α. If $\texttt{count}(x)$ exceeds α, the trie generates a new state, and attach it to the current state with a new edge.

Lemma 2. *Let S be an input string list of total size N. The algorithm* BuildDietTrie *in Fig. 4 computes a synopsis trie for S in $O(N)$ time and $O(|\mathcal{T}|)$ space.*

At present, we have no theoretical upper bounds of $|\mathcal{T}|$ and the maximum value of $\texttt{count}(\ell)$ according to the value of growth parameter $\alpha \geq 0$. Tentatively, setting $\alpha = cB$ for some constant $0 \leq c \leq 1$ works well in practice.

3.4 STEP2: Splitting a String List into Buckets

Let $BID = \{1, \ldots, b\}$ be a set of *bucket-id*'s for some $b \geq 0$. Then, a *bucket-id assignment* for $\mathcal{T}$ is a mapping $\beta : L(\mathcal{T}) \to BID$ that assigns bucket id's to the leaves. Let $(\mathcal{T}, BID)$ be a synopsis trie $\mathcal{T}$ whose leaves are labeled with the bucket-id assignment β.

A bucket-id assignment β for $\mathcal{T}$ is said to be *order-preserving* if it satisfies the following condition: for every leaves ℓ_1, ℓ_2, if $str(\ell_1) \leq_{\text{lex}} str(\ell_2)$ then $\beta(\ell_1) \leq \beta(\ell_2)$.

By the definition of a synopsis trie above, we know that for every string $s \in uniq(S)$, there is the unique leaf $\ell \in L(\mathcal{T})$ such that $str(\ell) \sqsubseteq s$. We denote this leaf by $\texttt{vertex}(s, \mathcal{T}) = \ell$. Then, the bucket-id assigned to s is defined by $\texttt{bucket}(\mathcal{T}, \beta, s) = \beta(\texttt{vertex}(s, \mathcal{T}))$. Given an input string list $S \in (\Sigma^*)^*$, $(\mathcal{T}, \beta)$ defines the partition

$$PART(S, \mathcal{T}, \beta) = S_1 \oplus \ldots \oplus S_b$$

where $S_k = (\ s \in S \mid bid(\mathcal{T}, \beta, s) = k\ )$ holds for every bid $k \in BID$. Then, the *maximum bucket size* of $(\mathcal{T}, BID)$ on input S is defined by $\max\{\ S_k \mid k = 1, \ldots, b\ \}$. Clearly, $S_1 \oplus \ldots \oplus S_b = S$ holds.

Lemma 3. *If β is order-preserving then $PART(S, \mathcal{T}, \beta)$ is an ordered partition.*

In Fig. 5, we show an algorithm SplitDietTrie for splitting an input string list S into buckets $PART(S, \mathcal{T}, \beta) = B_1 \oplus \ldots \oplus B_b$. We can easily see that the bucket-id assignment computed by algorithm SplitDietTrie is always order-preserving. Furthermore, we have the following lemma.

Algorithm: SplitDietTrie
Input: A string list $S = (s_1, \ldots, s_K)$ (in external memory), a synopsis trie $\mathcal{T}_{\alpha,S}$ for S (in internal memory), and a positive integer $M \geq B \geq 0$.
Output: An ordered partition $B_1 \oplus \cdots \oplus B_b$ for S.
1. //Compute a bucket-id assignment β
   - Let $N' = \sum_{\ell \in L(\mathcal{T}_{\alpha,S})} \texttt{count}(\ell)$; $B' = BN'/N$; $k = 1$; $A = 0$;
   - For each leaf $\ell \in L(\mathcal{T}_{\alpha,S})$ in the lexicographic order of $str(\ell)$, do:
     - If $(A + \texttt{count}(\ell) \leq B')$, then $\texttt{bucket}(\ell) := k$ and $A := A + \texttt{count}(\ell)$;
     - Else, $k := k + 1$ and $A := 0$;
   - $b = k$;
2. //Distribute all strings in S into the corresponding buckets.
   - For each $k = 1, \ldots, b$, do: $B_i := \emptyset$;
   - For each string $s \in S$, do:
     - Find the leaf $\ell = \texttt{vertex}(s, \mathcal{T}_{\alpha,S})$ reachable from the root by s;
     - Put s to the k-th bucket B_k in external memory for $k = \beta(\ell)$.

Fig. 5. An algorithm SplitDietTrie for splitting an input string list

Lemma 4. *Let S be an input list and $\mathcal{T}$ be a synopsis trie with bucket-id assignment for S. Suppose that $|\mathcal{T}| \leq M$. Then, the algorithm* SplitDietTrie *of in Fig. 5, given S and $\mathcal{T}$, computes an ordered partition of S in $O(N)$ time and $O(|\mathcal{T}|)$ space in main memory, where $N = ||S||$.*

We have the following lemma on the accuracy of the approximation.

Lemma 5. *Let $B' \geq 0$ $(0 \leq B' \leq M)$ and $k \geq 1$ be any integers. Let $\mathcal{T}$ be the synopsis trie for S. If the trie $\mathcal{T}$ on S satisfies $0 \leq \texttt{count}(v) \leq \frac{1}{k}B'$ for any vertex $\ell \in L(\mathcal{T})$, then the resulting ordered partition $S = B_1 \oplus \cdots \oplus B_b$ $(b \geq 0)$ satisfies that $\frac{k-1}{k}B \leq B_i \leq B$ holds for any bucket B_i $(1 \leq i \leq m)$.*

3.5 STEP3: Internal Sorting of Buckets

Once an input string list S of total size N has been split into an ordered partition $S_1 \oplus \ldots \oplus S_b = S$ of maximum bucket size $B \leq M$, we can sort each bucket

Algorithm: SortBucket
Input: A bucket $B \in (\Sigma^*)^*$.
Output: A bucket $C_i \in (\Sigma^*)^*$ obtained from B by sorted in $\leq_{\text{lex}}$.
1. Build a trie $\mathcal{T}_B$ for the bucket B in main memory;
2. Traverse all vertices v of $\mathcal{T}_B$ in the lexicographic order of path strings $str(v)$ and output $str(v)$.

Fig. 6. An algorithm SortBucket for sorting each bucket

B_i in main memory by using any internal memory sorting algorithm. For this purpose, we use a *internal trie sort* which is described in Fig. 6.

Lemma 6. *The algorithm* SortBucket *of Fig. 6 sorts a bucket B of strings in $O(N)$ time using main memory of size $O(N)$, where $N = ||B||$ and Σ is a constant alphabet.*

3.6 STEP4: Final Concatenation

Step 4 is just a concatenation of already sorted buckets $B_1, \ldots, B_b$. Therefore, this step requires no extra cost.

3.7 Analysis of the Algorithm

In this subsection, we give a case analysis of a practical performance of the proposed external sorting algorithm assuming a practical situations for such an algorithm. For the purpose, we suppose the following condition:

Condition 1. *The synopsis trie computed by* BuildDietTrie *and* SplitDietTrie *on the input string list S and the choice of a growth parameter α satisfies the following conditions:*

- *The synopsis trie fits into main memory, that is, $\texttt{size}(\mathcal{T}) \leq M$.*
- *The maximum bucket size of the ordered partition computed by* SplitDietTrie *at Step 2 does not exceed M.*

For the above condition to hold, we know that at least the input S satisfies that $N = O(M^2)$. Note that at this time, we only have a heuristic algorithm for computing a good synopsis trie without any theoretical guarantee for its performance. Finally, we have the following result.

Theorem 7. *Suppose that the algorithms* BuildDietTrie *and* SplitDietTrie *satisfy Condition 1 on the input S and $\alpha \geq 0$. The algorithm* DistStrSort *correctly solves the external string sorting problem in $O(N)$ time within main memory M using $3N$ sequential read (Step 1–Step 3), N random write (Step 2), and N sequential write (Step 3) in the external memory.*

From Theorem 7 and Lemma 1, we can show the following corollary using modified version of DistStrSort algorithm.

Corollary 8. *Let $(\Delta, \circ)$ be any associative binary operator, where operation $\circ$ can be computed in $O(1)$ time. Suppose that the algorithms* BuildDietTrie *and* SplitDietTrie *satisfy Condition 1 on the input S and $\alpha \geq 0$. There exists an algorithm that solves the string counting (STR-COUNT), and the string aggregation problems (STR-AGGREGATE($\Delta, \circ$)) for an input string list S in $O(N)$ time using main memory of size M in external memory.*

4 Experimental Results

In this section, we present experimental results on real datasets to evaluate the performance of our algorithms. We implemented our algorithms in C. The experiments were run on a PC (Xeon 3.6GHz, Windows XP) with 3.25 gigabytes of main memory.

We prepared two datasets *data A* and *data B* consisting of cookie values from Web access logs. The parameters of the datasets are shown in Table 1.

Table 1. Parameters of the datasets

Dataset	*data A*	*data B*
File size	67MB	1.1GB
Number of records (with duplications)	140×10^4	2290×10^4
Number of records (no duplications)	22×10^4	109×10^4
Average of record length	45.0	44.9
Variance of record length	45.1	19.7
Maximum record length	58	58
Minimum record length	15	15

4.1 Size of Synopsis Trie

First, we studied sizes of synopsis tries constructed by the algorithm BUILDDIETTRIE by varying the growth parameter $\alpha = 100, 1000$, or 10000. Table 2 shows the number of states of the constructed synopsis trie for each α on the *data A*. We can see that the bigger α is, the smaller the size of the constructed synopsis trie becomes.

Table 2. Sizes of the constructed synopsis tries

Growth parameter α	100	1,000	10,000
Number of states	426,209	92,768	5,030

4.2 Running Time

Next, we measured the running times of the subprocedures BUILDDIETTRIE and SPLITDIETTRIE on the *data A*, by varying the number of buckets and the growth parameter. Fig. 7 show the results. The horizontal axis in the figure represents pairs of the number of buckets and the growth parameter. The vertical axis represents the running time.

The results indicate that the running time is smaller as the growth parameter α increases. Thus, a smaller synopsis trie can be constructed in a reasonable computational time by adjusting α.

Finally, we studied the running time of the algorithm DISTSTRSORT. Fig. 8 shows the running times on the *data B* of DISTSTRSORT and the GNU sort 5.97. From the figure, we observe that the running time of our algorithm scales linearly with the size of data and is faster than the GNU sort for larger data sizes.

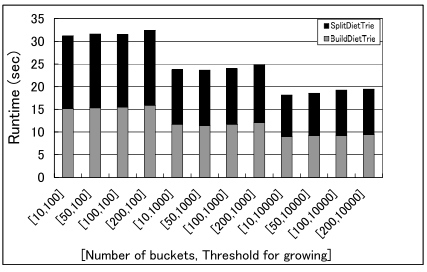


Fig. 7. Running time of BUILDDIETTRIE and SPLITDIETTRIE

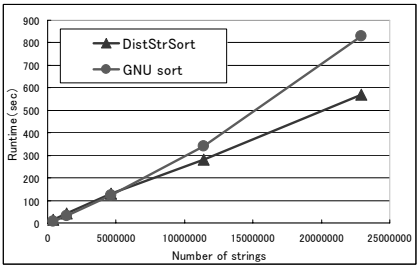


Fig. 8. Running time of DISTSTRSORT and GNU sort

5 Conclusion

In this paper, we presented a new algorithm for sorting large collections of strings in external memory. The main idea of our method is first splitting a set of strings to some buckets by adaptive construction of a synopsis trie for input strings, then sort the strings in each bucket, and finally merge the sorted buckets into an output collection. Experiments on real datasets show the effectiveness of our algorithm.

In this paper, we only made empirical studies on the performance of adaptive strategy for construction of a synopsis trie for data splitting. The probabilistic analysis of the upper bounds of $|\mathcal{T}|$ and the maximum value of $\texttt{count}(\ell)$ concerning to the value of growth parameter $\alpha \geq 0$ is an interesting future research.

We considered only the case that the input size is at most the square of the memory size, and then presented a two-level external sorting algorithm. It is a future research to develop a hierarchical version of our algorithm for inputs of unbounded size. A stream-based distributed version of distribution string sort algorithms is another possible direction.

References

1. Abiteboul, S., Agrawal, R., Bernstein, P.A., Carey, M.J., Ceri, S., Croft, W.B., DeWitt, D.J., et al.: The Lowell database research self-assessment. C. ACM 48(5), 111–118 (2005)
2. Abiteboul, S., Buneman, P., Suciu, D.: Data on the Web. Morgan Kaufmann, San Francisco (2000)
3. Arge, L., Ferragina, P., Grossi, R., Vitter, J.S.: On Sorting Strings in External Memory. In: Proc. the 29th Annual ACM Symposium on Theory of Computing (STOC 1997), pp. 540–548 (1997)
4. Bentley, J., Sedgewick, R.: Fast Algorithms for Sorting and Searching Strings. In: Proc. the 8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1997), pp. 360–369 (1997)
5. Ferragina, P., Grossi, R.: The String B-tree: A New Data Structure for String Search in External Memory and Its Applications. J. ACM 46(2), 236–280 (1999)
6. Fredkin, E.: Trie Memory. C. ACM 3(9), 490–499 (1960)

7. Laird, P., Saul, R.: Discrete sequence prediction and its applications. Machine Learning 15(1), 43–68 (1994)
8. Manber, U., Myers, E.W.: Suffix Arrays: A New Method for On-Line String Searches. SIAM J. Comput. 22(5), 935–948 (1993)
9. Manzini, G., Ferragina, P.: Engineering a lightweight suffix array construction algorithm (Extended abstract). In: Möhring, R.H., Raman, R. (eds.) ESA 2002. LNCS, vol. 2461, pp. 698–710. Springer, Heidelberg (2002)
10. Moffat, A.: Implementing the PPM data compression scheme. IEEE Trans. Communications COM-38(11), 1917–1921 (1990)
11. Ramakrishnan, R., Gehrke, J.: Database Management Systems. McGraw-Hill Professional, New York (2000)
12. Ron, D., Singer, Y., Tishby, N.: The power of amnesia: learning probabilistic automata with variable memory length. Machine Learning 25(2-3), 117–149 (1996)
13. Sadakane, K.: A Fast Algorithms for Making Suffix Arrays and for Burrows-Wheeler Transformation. In: Proc. the 8th Data Compression Conference (DCC 1998), pp. 129–138 (1999)
14. Sinha, R., Zobel, J.: Efficient Trie-Based Sorting of Large Sets of Strings. In: Proc. the 26th Australasian Computer Science Conference (ACSC 2003) (2003)
15. Stonebraker, M., Cetintemel, U.: One Size Fits All: An Idea Whose Time Has Come and Gone. In: Proc. the IEEE 21st International Conference on Data Engineering (ICDE 2005), pp. 2–11, keynote (2005)

Incrementally Mining Recently Repeating Patterns over Data Streams*

Jia-Ling Koh and Pei-Min Chou

Department of Computer Science and Information Engineering
National Taiwan Normal University
Taipei, Taiwan 106, R.O.C
jlkoh@csie.ntnu.edu.tw

Abstract. Repeating patterns represent temporal relations among data items, which could be used for data summarization and data prediction. More and more data of various applications is generated as a data stream. Based on time sensitive concern, mining repeating patterns from the whole history data sequence of a data stream does not extract the current trend of patterns over the stream. Therefore, the traditional strategies for mining repeating patterns on static database are not applicable to data streams. For this reason, an algorithm, named appearing-bit-sequence-based incremental mining algorithm, for efficiently discovering recently repeating patterns over a data stream is proposed in this paper. The appearing bit sequences are used to monitor the occurrences of patterns within a sliding window. Two versions of algorithms are proposed by maintaining the appearing bit sequences of maximum repeating patterns and closed repeating patterns, respectively. Accordingly, the cost of re-mining repeating patterns over a sliding window is reduced to that of monitoring frequency changes of the maintained patterns. The experimental results show that the incremental mining methods perform much better than the re-miming approach.

Keywords: repeating patterns, incremental mining, data streams.

1 Introduction

Repeating patterns represent temporal relations among data items in a data sequence, which could be used for data summarization and data prediction. Therefore, repeating patterns discovery is of great interest for applications with sequential data representations. There have been many approaches proposed for mining repeating patterns [5][6][7][10]. The concept of repeating patterns was used in [5] to represent the significant content of a music object. It proposed a data structure called *correlative matrix* to aid the process for extracting repeating patterns. Moreover, for providing more efficient processing, the *String-Join* algorithm was developed to extract *non-trivial* repeating patterns instead of repeating patterns in a music object. In [6], we

* This work was partially supported by the R.O.C. N.S.C. under Contract No. 96-2221-E-003-018 and 96-2524-S-003-001.

S. Chawla et al. (Eds.): PAKDD 2008 Workshops, LNAI 5433, pp. 26–37, 2009.

designed *bit index sequences* to characterize note sequences of music objects. In the mining process, the frequency of a candidate pattern was obtained by performing **shift** and **and** operations on bit sequences and counting the number of 1s in the resultant bit sequence. Therefore, frequency checking of a pattern could be performed quickly. In addition, this approach avoided scanning the data sequences repeatedly. Fault-tolerant data mining would discover more general and useful information for real-world dirty data. By extending the idea of appearing bit sequences, fault-tolerant appearing bit sequences were defined in [7] to represent the locations where candidate patterns appear in a data sequence with insertion/deletion errors being allowed. Accordingly, the fault-tolerant frequency of each candidate pattern could be obtained quickly.

Recently, the data stream, which is an unbounded sequence of data elements generated at a rapid rate, provides a dynamic environment for collecting data. Knowledge discovery on streaming data is a research topic of growing interest especially for learning concept drifts [4][12]. The lossy-counting algorithm is a representative approach for mining frequent itemsets over data streams [11]. For reducing the required memory usage, the lossy-counting algorithm only maintained patterns with supports being no less than an error tolerance parameter ε. Consequently, the frequency of a pattern was estimated by compensating the maximum number of times that the pattern could have occurred before being monitored. Time sensitivity is another important issue when mining frequent itemsets over data streams. It is likely that the embedded knowledge in a data stream will change quickly as time goes by. In order to catch recent trend of data, the *estDec* algorithm [2] decayed the old occurrences of each itemset to diminish the effect of old transactions on the mining result of frequent itemsets. However, in particular applications, it is interested only the frequent patterns mined from the recently arriving data within a fixed time period. A time-sensitive sliding window approach was proposed in [9] for mining recently frequent itemsets within the current sliding window in a data stream. Accordingly, the recently frequent itemsets were discovered from the most recent *w* blocks of transactions. For discovering closed frequent itemsets in a sliding window, [3] proposed a compact data structure, named a closed enumeration tree (CET), to monitor the itemsets which consist of the boundary between closed frequent itemsets and the rest of the itemsets. Accordingly, the closed frequent itemsets in a sliding window are discovered by maintaining the CET structure incrementally. For achieving the similar purpose, a data structure called summary frequent itemset forest was introduced in [8] for incremental maintaining the essential information about maximal frequent itemsets embedded in a data stream.

Although the problem of mining frequent itemsets over data streams has been investigated in the above literatures [2][3][8][9][11], the temporal relations among data items were not considered in these studies. Accordingly, it is essential to provide a data structure for maintaining sequential information of items within a sliding window to discover recently repeating patterns appearing in the window. Moreover, an incremental pattern mining strategy is necessary to avoid performing re-mining when the window slides. In this paper, an algorithm, named appearing-bit-sequence-based incremental mining algorithm, for efficiently mining recently repeating patterns over a data stream is proposed. The appearing bit sequences are used to monitor the occurrences of patterns within a sliding window. Two versions of algorithms are proposed by maintaining the appearing bit sequences of maximum repeating patterns and closed repeating patterns, respectively. Accordingly, the cost of mining repeating patterns in

a sliding window is reduced to that of mining frequency changes of the maintained patterns. The experimental results show that the incremental mining methods perform much better than the re-miming approach.

This paper is organized as follows. The related terms used in this paper are defined in Section 2 first. In Section 3, the proposed algorithm for discovering recently repeating patterns incrementally from the maintained structure is introduced. The performance evaluation on the proposed algorithms is reported in Section 4. Finally, in Section 5, we conclude this paper.

2 Preliminaries

2.1 Problem Definition

Let $I = \{I_1, I_2, I_3 \ldots\ldots I_m\}$ denote the set of data items in the specific application domain. Suppose an item in I is inputted at each time point. Accordingly, a data stream $DS = [S_1, S_2, S_3 \ldots)$ is formed, where each S_i denoting a data item in I is associated with an time identifier i. Under a predefined sliding window size W, the ***sequence window*** at time t, denoted as SW_t, represents the sequence of items $[S_{t-W+1}, S_{t-W+2}, \ldots, S_t]$. The ith item in SW_t is denoted as $SW_t[i]$. A pattern $P= p_1 p_2 \ldots p_k$ ($k \geq 1$) is a data sequence consisting of one or more items in I. The number of items in pattern P is called the length of P, denoted as $|P|$.

Suppose a sequence window SW_t and a pattern $P=p_1p_2\ldots p_m$ are given. If there exists an positive integer i such that $SW_t[i\text{-}m+1]SW_t[i\text{-}m+2]\ldots SW_t[i] = p_1p_2\ldots p_m$, it is called that p ***appears in*** SW_t and SW_t ***contains*** p on position i. The number of positions in SW_t, where SW_t contains p, is named the ***recent frequency*** of p in DS at time t, denoted as $RF_t^{DS}(p)$. Given a user specified minimum frequency, denoted as F_{min}, a pattern p is called a ***recently repeating patterns*** (RRPs) in DS at time t if $RF_t^{DS}(p) \geq F_{min}$. A recently repeating pattern with length k is named a *k-RRP*.

2.2 Appearing Bit Sequences

In our previous work [6], the ***appearing bit sequences*** were proposed to speed up the mining of repeating patterns in a data sequence with fixed length. For each kind of data item N in a data sequence, N has a corresponding *appearing bit sequence* (denoted as $Appear_N$). The length of an appearing bit sequence equals the length of the data sequence. The leftmost bit is numbered as bit 1. If a data item appears on the ith position of the data sequence, bit i in the appearance bit sequence of this data item is set to be 1; otherwise, it is set to be 0. A bit index table is used to store the appearing bit sequences for all the data items in the data sequence. Therefore, the frequency of a data item is obtained according to the number of bits with value 1 in its appearing bit sequence, without needing to scan the data sequence repeatedly.

Example 1. The bit index table of "BCBCABCABC" is given as shown in Figure 1.

1) Suppose we would like to get $Appear_{AB}$. A position i where "AB" appears implies "A" must appear on position i and "B" appears on the next position (i+1).

Step1. Get $Appear_A$=0000100100 and $Appear_B$=1010010010 from Table 1.
Step2. Perform one **right shift** operation on $Appear_A$ (shift bit i to bit(i+1), where 1≤i≤9, and set bit 1 to be 0), $R\_shift(Appear_A,1)$ = 0000010010.
Step3. $Appear_{AB} = R\_shift(Appear_A, 1) \wedge Appear_B$ = 0000010010.

	B	C	B	C	A	B	C	A	B	C
$Appear_A$	0	0	0	0	1	0	0	1	0	0
$Appear_B$	1	0	1	0	0	1	0	0	1	0
$Appear_C$	0	1	0	1	0	0	1	0	0	1

Fig. 1. The bit index table for data sequence "BCBCABCABC."

2) Suppose we would like to get $Appear_{ABC}$ after getting $Appear_{AB}$. A position i where "ABC" appears implies "AB" must appears on position i and "C" appears on position i+1.
Step1. Obtain $Appear_C$=0101001001 from Table 1.
Step2. Perform one **right shift** operation on $Appear_{AB}$, $R\_shift(Appear_{AB},1)$ = 0000001001.
Step3. $Appear_{ABC} = R\_shift(Appear_{AB}, 1) \wedge Appear_C$ = 0000001001.

Accordingly, the frequency of "ABC" in the sample data sequence equals the number of bits with value 1 in $Appear_{ABC}$ (that is 2 in this case).

Suppose pattern $P=P_1P_{2...}P_m$, where P_i (i=1, …,m) is a data item and m>1. Let $P'=P_1P_{2...}P_{m-1}$ and $X=P_m$. In general, $Appear_P$ is deduced from $Appear_{P'}$ and $Appear_X$ according to the following recursive formula:

$$Appear_P = R\_shift(Appear_{P'},1) \wedge Appear_X .$$

3 Incremental Repeating Patterns Mining

The processing of the incremental repeating patterns mining approach is characterized into two phases: window initialization phase and window sliding phase. The window initialization phase is activated when the first time the sequence window becomes full occurs. In this phase, the appearing information of repeating patterns in the *initial sequence window* is discovered and maintained. After that, the window sliding phase is activated. A newly coming data item is inserted and the oldest data item has to be removed from the sequence window. Without needing to re-perform repeating patterns mining on the new sequence window, the RRPs are discovered incrementally from the maintained information efficiently.

3.1 Window Initialization Phase

Definition 1. Prefix-subpattern and suffix-subpattern
Suppose a pattern $P=x_1x_2…x_k$ is given. For any pattern $P_1= x_1x_2…x_i$, where $i<k$, P_1 is named the *i-prefix-subpattern* of P. Pattern $x_1x_2…x_{k-1}$ is called the *maximum prefix-subpattern* of P. For any pattern $P_2= x_jx_{j+1}…x_k$, where $1<j$, P_2 is named the (k-j+1)-*suffix-*

subpattern of *P*. Pattern $x_2x_3\ldots x_k$ is called the *maximum suffix-subpattern* of *P*. All the prefix and suffix-subpatterns of *P* are named *sub-patterns* of *P*, and *P* is a *super-pattern* of its sub-patterns.

Definition 2. Maximum repeating pattern
A *maximum repeating pattern* is defined as a RRP for which none of its super-patterns are RRPs at the same time.

In the window initialization phase, a bit index table, named a *recent bit index table*, is constructed to represent the initial sequence window. An Apriori-like algorithm is developed to extract RRPs from the initial sequence window, in which appearing bit sequences of patterns are used to count the recent frequencies of candidate patterns efficiently. For providing incremental mining in the window sliding phase, the discovered repeating patterns and their corresponding appearing bit sequences are maintained. In order to reduce required storage space of the maintained patterns, only the maximum repeating patterns are remained.

First, the initial sequence window is scanned once to create the recent bit index table. The 1-*RRP*s are those data items with recent frequencies being no less than F_{min}. A pair of (*k*-1)-*RRP*s are merged to produce a candidate *k*-pattern. To avoid generating duplicate candidates, a pattern P_1 is merged with another pattern P_2 only if they have the same (*k*-2)-suffix-subpattern. The resulting candidate pattern P_3 is a pattern with length (*k*+1), which is obtained by concatenating pattern P_1 with the last data item of P_2. The appearing bit sequence of P_3 is derived by performing a right shift operation on $\boldsymbol{Appear}_{P_1}$ followed by an **and** operation with $\boldsymbol{Appear}_{P_2}$. Then the recent frequencies of the candidate (*k*+1)-patterns are counted from their appearing bit sequences efficiently.

Whenever a (*k*+1)-*RRP P* is discovered, the pattern *P* and its appearing bit sequence $\boldsymbol{Appear}_{P}$ is inserted into a table for maintaining the recent maximum repeating patterns. Moreover, all the sub-patterns of *P* are removed from the table. The above process repeats until no more (*k*+1)-*RRP* is discovered.

3.2 Window Sliding Phase

In the window sliding phase, a new data item is appended into and the first data item is removed from the sequence window. However, the middle part of SW_{t-1} is remained in SW_t. In our method, the RRPs in the new current sequence window are discovered incrementally from the maintained information of previous window to prevent from generating candidate patterns iteratively.

First, the recent bit index table is updated. Each appearing bit sequence in the table is performed by a left shift operation to remove the first bit which corresponds to the out-of-date time point. Besides, for the newly coming data item *x*, the last bit of $Appear_x$ is set to be 1.

Let *y* denote the removed data item. There are two cases that the recent frequencies of patterns will be changed. The first case is for those patterns that are beginning with *y* and have an occurrence at the removed data item. Then the recent frequencies of these repeating patterns are reduced by one. Similarly, it is possible that the recent frequency of a pattern ending with *x* is increased such that the pattern becomes a

RRP. In the following, the process of incremental mining RRPs will be introduced according to these two cases.

Step 1: remove the out-of-date data item *y*

For each pattern maintained in the maximum repeating pattern table, its appearing bit index sequence is updated by performing a left shift operation. According to the new appearing bit index sequence of a pattern *P*, let bit *i* denote the first non-zero bit in $Appear_P$. It is implied that *P* has an occurrence located out of the new sequence window if $i < |P|$. Therefore, bit *i* of $Appear_P$ is set to be 0 and the recent frequency of *P* is reduced by one.

If the recent frequency of *P* after updating is less than F_{min}, *P* is not a RRP any more. However, all the suffix-subpatterns of *P* remain to be RRPs. Let *P* be denoted by $yp_1p_2\ldots p_k$, where each p_i (i=1,..k) is a data item. Therefore, the maximum suffix-subpattern of *P*, denoted as *P'*, becomes a new maximum repeating pattern. $Appear_{P'}$ is computed according to the information in the recent bit index table. The pattern *P'* is inserted into the maximum repeating pattern table to replace pattern *P*.

Moreover, it is possible that part of the prefix-subpatterns of *P* remains to be RRPs. Initially, *P"* is set to be *y* and $RF_t^{DS}(P")$ is obtained according to $Appear_y$. If *P"* is a RRP, it is appended with p_i (i=1,..k) one by one until a maximum repeating pattern is discovered.

Step 2: append the newly coming data item *x*

Let *z* denote the last data item in the previous sequence window. If $RF_t^{DS}(x)$ is less than F_{min}, there is not any RRP newly generated. Otherwise, it is possible to form longer repeating patterns for the repeating patterns ending with *z*.

According to the maintained maximum repeating patterns, there are two situations to get the repeating patterns ending with *z*. The first one is to retrieve the suffix-subpatterns of a maximum pattern ending with *z*. Let *P* denote a maximum repeating pattern ending with *z*. The pattern *P* is appended with the newly coming data item *x* to generate a longer pattern *Px*, where $Appear_{Px}$ is computed according to the maintained information of $Appear_P$ and $Appear_x$. If *Px* is certified to be a RRP, it is a new maximum repeating pattern. Accordingly, *Px* is inserted into the maximum repeating pattern table to replace pattern *P*. In the case that *Px* is not a RRP, it is still possible to find new RRPs by appending *x* to the suffix-subpatterns of *P*. Let *P* be denoted by $p_1p_2\ldots p_mz$, where each individual p_i (i=1,..m) is a data item. Each suffix-subpattern of *P*, denoted by *P'*, is composed of $p_kp_{k+1}\ldots p_mz$ for $k \geq 2$. Initially, *P'* is set to be the 1-suffix-subpattern of *P*, that is *z* in this case. Then, $Appear_{P'x}$ is computed according to $Appear_z$ and $Appear_x$. If *P'x* is a RRP, *P'* is extended to be p_mz, which is the 2-suffix-subpattern of *P*. The appearing bit sequence of p_mzx is obtained by performed two left shift operations on $Appear_{p_m}$ followed by an AND operation with $Appear_{zx}$. The above process repeats until finding a *k* such that $P'=p_{k-1}p_k\ldots p_mz$ and $RF_t^{DS}(P'x)$ is less than F_{min}. Finally, pattern $p_k\ldots p_mzx$ is a maximum repeating pattern, which is inserted into the maximum repeating pattern table.

In addition to the suffix-subpatterns of a maximum repeating pattern ending with *z*, it is possible that a repeating pattern ending with *z* is contained in the middle of a maximum repeating pattern. Let *Q* denote a maximum repeating pattern consisting of $q_1q_2\ldots q_mzq_{m+1\ldots}q_n$, where each individual q_i (i=1,..n) is a data item. Let *Q'* denote the maximum prefix-subpattern of *Q* ending with *z*, which is composed of $q_1q_2\ldots q_mz$. Then all the suffix-subpatterns of *Q'* are repeating patterns ending with z. First, *Q"* is

set to be the 1-suffix-subpattern of *Q'*, that is *z* in this case. If *Q"x* is a RRP, *Q"* is extended to be the 2-suffix of *Q'*. The above process repeats until finding a *k* such that $Q"x=q_kq_{k+1}\ldots q_mzx$ is a RRP but $q_{k-1}q_k\ldots q_mzx$ is not or *Q"* has been identical to *Q'*. Finally, pattern *Q"x* is a maximum repeating pattern, which is inserted into the maximum repeating pattern table.

SW_{20} A B C B C A B B C A B C B A B C B A B C

Data Item	Appearing Bit Sequence	Frequency
A	1 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0	5
B	0 1 0 1 0 0 1 1 0 0 1 0 1 0 1 0 1 0 1 0	9
C	0 0 1 0 1 0 0 0 1 0 0 1 0 0 0 1 0 0 0 1	6

(a)

Pattern	Appearing Bit Sequence	Frequency
ABCB	0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0	3

(b)

SW_{21} B C B C A B B C A B C B A B C B A B C A

Data Item	Appearing Bit Sequence	Frequency
A	0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 1	5
B	1 0 1 0 0 1 1 0 0 1 0 1 0 1 0 1 0 1 0 0	9
C	0 1 0 1 0 0 0 1 0 0 1 0 0 0 1 0 0 0 1 0	6

(c)

Pattern	Appearing Bit Sequence	Frequency
ABCB	0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0	3

(d)

Pattern	Appearing Bit Sequence	Frequency
ABC	0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0	3
BCB	0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0	3

(e)

Pattern	Appearing Bit Sequence	Frequency
ABC	0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0	3
BCB	0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0	3
BCA	0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1	3

(f)

Fig. 2. The recent bit index table and maintained patterns of the example

According to the updated information in the maximum repeating pattern table, all the RRPs could be enumerated. Moreover, it provides the hints for mining RRPs incrementally in the next sequence window.

Example 2. Suppose a data stream DS=[ABCBCABBCABCBABCBABCA…) is given, the size of the sliding window is 20, and Fmin is set to be 3. Accordingly, the initial sequence window SW20=[ABCBCABBCABCBABCBABC]. The corresponding recent bit index table is constructed as shown in Figure 2(a). After window initialization phase is performed, the maximum repeating pattern ABCB is discovered and maintained in the maximum repeating pattern table as shown in Figure 2(b).

When the sequence window slides to be SW_{21}=[BCBCABBCABCBABCBABCA], a data item "A" is newly inputted and the first "A" is removed. All the appearing bit

sequences are performed by a left shift operation, individually. Besides, the last bit of $Appear_A$ is set to be 1. The recent bit index table is updated to be the one shown in Figure 2(c). To eliminate the occurrence of the removed item "A", the appearing bit sequence of the maximum repeating pattern "ABCB" is performed by a left shift operation to get the one shown in Figure 2(d). It is implied that the first occurrence of "ABCB" is out-of-date because the first non-zero bit in $Appear_{\mathrm{ABCB}}$, located on bit 3, is less than the length of pattern "ABCB". Accordingly, the bit is reset to be 0 and the recent frequency of "ABCB" becomes to be 2. Since pattern "ABCB" is not a RRP in SW_{21}, the maximum suffix-subpattern of "ABCB", "BCB" in this case, becomes a maximum repeating pattern. The appearing bit sequence of "BCB" is obtained from a series of computing on $Appear_{\mathrm{B}}$ and $Appear_{\mathrm{C}}$. Furthermore, the prefix-subpatterns of "ABCB" are enumerated to find a longest prefix-subpattern of "ABCB" which is a RRP. Consequently, "ABC" is discovered to be a maximum repeating pattern in SW_{21}. The maintained maximum repeating table is updated to be the one shown in Figure 2(e).

The last data item in the previous sequence window, SW_{20}, is "C". It is possible to generate a longer repeating pattern by appending the inputted item "A" to a repeating pattern ending with "C". First, the new data item "A" is appended to the maximum repeating pattern ending with "C", "ABC" in this case, to generate a longer pattern "ABCA". Since "ABCA" is not a RRP, the other RRPs ending with "C" are generated by enumerating the suffix-subpatterns of "ABC": "C" and "BC" in sequence. Finally, "BCA" is discovered to be a newly generated maximum repeating pattern. After the process of window sliding phase is performed, the maintained patterns in the maximum repeating table are shown as Figure 2(f).

3.3 Maintaining Closed Repeating Patterns

Definition 3. Closed repeating pattern
Given patterns P_1 and P_2, it is named that P_1 is *closed contained* in P_2 if P_1 is a sub-pattern of P_2 and $RF_{\mathrm{t}}^{DS}(\mathrm{P}_1) = RF_{\mathrm{t}}^{DS}(\mathrm{P}_2)$. A *closed repeating pattern* is defined as a RRP which is not closed contained in any of its super-patterns.

Theorem 1. If pattern P is closed contained in pattern Px, the sub-patterns of P, which are closed contained in P, are also closed contained in Px.

For providing a lossless compression of the whole collection of RRPs, the closed repeating patterns in a sequence window are maintained instead of keeping the maximum repeating patterns. The processing steps described in the previous section are performed similarly on the maintained closed repeating patterns. Besides, the following properties are used to improve the processing efficiency.

Property 1. Let $P=yp_1p_2\ldots p_k$ denote a closed repeating pattern in SW_t. Besides, the maximum suffix-subpattern of P, $P'=p_1p_2\ldots p_k$, is closed contained in P. In the case that P becomes a non-repeating pattern in SW_{t+1} due to the removing of the out-of-date data item y, P' becomes a new closed repeating pattern in SW_{t+1} to replace P and $Appear_{P'}=Appear_P$.

Property 2. Let $P=yp_1p_2\ldots p_k$ denote a closed repeating pattern in SW_t. Besides, a prefix-subpattern of P, $P''=yp_1p_2\ldots p_j$ ($j<k$), is closed contained in P. In the case that P becomes a non-repeating pattern in WS_{t+1} due to the removing of the out-of-date data

item *y*, *P"* is also a non-repeating pattern in SW_{t+1}. Accordingly, *P* is removed from the list of maintained patterns without needing to check the prefix-subpatterns of *P*.

Property 3. Let *z* denote the last data item in SW_t and *P* denote a closed repeating pattern ending with *z*. In the case that *Px* becomes a repeating pattern in SW_{t+1} due to the inserting of the new data item *x*, according to [Theorem 1], all the sub-patterns of *P* which are closed contained in *P* are closed contained in *Px* if *P* is closed contained in *Px*. Therefore, *Px* becomes a new closed repeating pattern in SW_{t+1} to replace *P* in this case.

Property 4. Let *z* denote the last data item in SW_t and *P* denote a closed repeating pattern ending with *z*. In the case that *Px* is not a RRP in SW_{t+1} due to the inserting of the new data item *x*, for any suffix-subpattern of *P*, *P'*, being closed contained in *P*, *P'x* is not a RRP in SW_{t+1}. Therefore, it is not necessary to enumerate the suffix-subpatterns of *P* for generating longer patterns.

Property 5. Let *z* denote the last data item in SW_t and *Q* denote a closed repeating pattern consisting of $q_1q_2\ldots q_mzq_{m+1\ldots}q_n$. For any repeating pattern *Q'*, which is ending with *z* and closed contained in the middle of *Q*, it only appears when *Q* ever appears. Therefore, it is impossible that *Q'* will become a repeating pattern in SW_{t+1} by inserting the new data item *x*. Therefore, it is not necessary to enumerate the sub-patterns of *Q* ending with *z* for generating longer patterns.

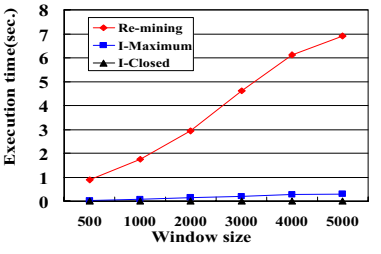
4 Performance Study

The proposed appearing-bit-sequence-based incremental mining algorithms were implemented using Visual C++ 6.0. In the following experiments, according to the properties of the maintained patterns, the two versions of algorithms are notated as ***I-Maximum*** and ***I-Closed***, respectively. Moreover, the appearing bit sequence approach [6] is applied to discover RRPs within each sequence window for comparison, which is notated as ***Re-mining*** algorithm. The experiments have been performed on a personal computer with 3.4GHz Intel Pentium IV CPU and 1 GB main memory.

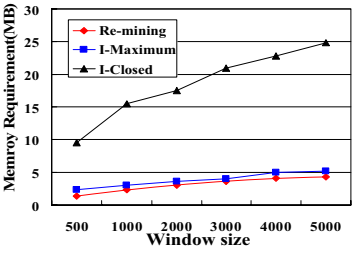
The data sets are generated by the IBM data generator [1], where the generated sequential transaction data are concatenated together to simulate a sequence data stream with hidden patterns. Two input parameters are varied when running the data generator, where *I* is used to specify the number of various data items and *P* denotes the number of potential patterns in the generated data sequences.

In the following four experiments, the data sets are generated by concatenating 50,000 sequential data with I100P100 setting. The scalabilities of I-Maximum and I-Closed algorithms on execution time and memory usage are compared with the ones of Re-mining algorithm under various parameters setting. According to theses experimental results, the effectiveness of incremental mining of RRPs is observed.

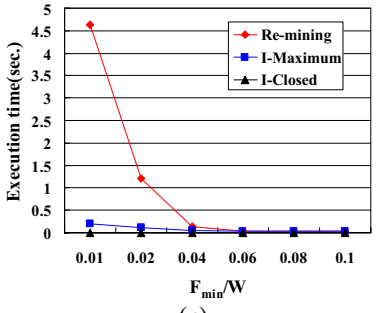
Experiment 1. Changing the sliding window size **(*W*)**
In this experiment, F_{min} is set to be *W* *0.01. Figure 3(a) shows the average running time of I-Maximum, I-Closed, and Re-mining over a sliding window under different *W* setting. It is reported that the execution efficiency of I-Maximum and I-Closed outperforms



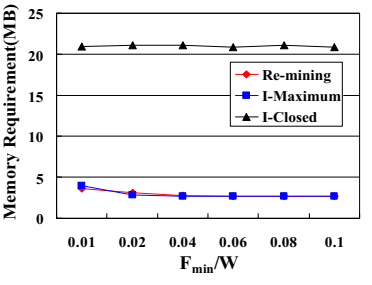
(a)



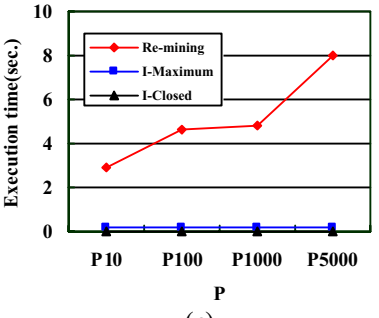
(b)



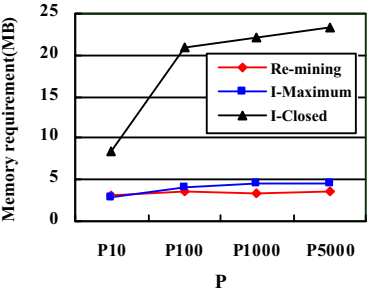
(c)



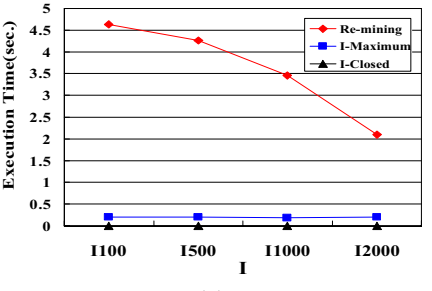
(d)



(e)



(f)



(g)

Fig. 3. Results of Experiments

the one of Re-Mining algorithm significantly. When W increases, the execution efficiency of Re-mining is much slower than the other two algorithms. Both I-Maximum and I-Closed are linear scalable, but I-Closed has better scalability. The total space needed for three algorithms are analyzed as shown in Figure 3(b). As W increases, due to the increasing of the length of an appearing bit sequence, the memory requirements of all the three algorithms grow linearly with the window size W. For I-Closed and I-Maximum, the major memory space needed is to hold the maintained patterns for incremental mining. Since the number of closed repeating patterns is much larger than the one of maximum repeating patterns, the memory requirement of I-Maximum is larger than the other two. Although the memory requirement of I-Closed is 5 times of the one of Re-mining, the execution efficiency of I-Closed is near 1000 times faster than Re-mining. Moreover, with slightly larger memory requirement than Re-mining, 20 times of execution efficiency is achieved by I-Maximum.

Experiment 2. Changing the minimum frequency threshold value **(F_{min}).**
In this experiment, W is set to be 3000. Figure 3(c) shows the execution time of I-Maximum, I-Closed, and Re-mining under different F_{min} setting. As seen in the figure, the execution time of Re-mining grows exponentially as F_{min} decreases because the number of satisfying patterns increases dramatically. However, the execution of I-Maximum is faster than that of Re-mining nearly an order of magnitude. The execution of I-closed is even faster than that of I-Maximum. In the situations when F_{min} is larger than W*0.04, much fewer repeating patterns are discovered. Therefore, the execution time of all the three algorithms is approximately the same. Because of the reason described above, it is shown in Figure 3(d) that the memory requirements of both I-Maximum and Re-mining increase as Fmin decreases. Although I-Closed has the largest memory requirement among the three algorithms, the amount of its required memory keeps nearly stable.

Experiment 3. Changing the setting of P for generating data sets
In this experiment, parameter I is set to be 100. Besides, W and F_{min} are fixed to be 3000 and W *0.01(=30), respectively. The execution times and memory requirements of I-Maximum, I-Closed, and Re-mining under different P setting are shown in Figure 3(e) and 3(f), respectively. As the number of potential patterns increases, more processing cost is required by Re-mining for generating candidate patterns. Since the main processing costs of I-Maximum, I-Closed are spent for mining frequency changes caused by window sliding, the execution time of these two algorithms keeps more stable than the one of Re-mining. The memory requirement of I-Closed is proportional to the number of repeating patterns. Therefore, as shown in Figure 3(f), the memory requirement of I-Closed grows as the number of various data items increases. However, the observed space-time trade-off between I-Closed and Re-mining is coincident with the one shown in [Experiment 1].

Experiment 4. Changing the setting of parameter I for generating data sets
In this experiment, parameter P is set to be 100. Besides, W and F_{min} are set to be 3000 and W *0.01(=30), respectively. The execution time of I-Maximum, I-Closed, and Re-mining under different I setting is shown in Figure 3(g). As the number of various data items increases, the distribution of patterns in a sliding window becomes sparser such that the number of satisfying patterns decreases. Accordingly, the execution time of Re-mining goes down as the number of various data items increases.

However, for the same reason stated in [Experiment 3], the running time of both I-Maximum and I-Closed remains stable.

5 Conclusion and Future Works

In this paper, the appearing bit sequences are used to monitor the occurrences of repeating patterns within a sliding window. Two versions of incremental algorithms are proposed to maintain the maximum and closed repeating patterns in a sliding window, respectively. Accordingly, the cost of re-mining repeating patterns over a sliding window is reduced to that of mining frequency changes of the maintained patterns. It is reported from the experimental results that the incremental mining methods perform much better than the re-miming approach.

References

[1] Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules. In: Proc. of Int. Conf. on Very Large Data Bases (1994)
[2] Chang, J.H., Lee, W.S.: Finding Recent Frequent Itemsets Adaptively over Online Data Streams. In: Proc. the 9th ACM International Conference on Knowledge Discovery and Data Ming (2003)
[3] Chi, Y., Wang, H., Yu, P.S., Muntz, R.R.: Moment: Maintaining Closed Frequent Itemsets over a Stream Sliding Window. In: Proc. Int. Conf. on Data Mining (ICDM 2004) (2004)
[4] Domingos, P., Hulten, G.: Mining high-speed data streams. In: Proc. The 6th ACM International Conference on Knowledge Discovery and Data Ming (2000)
[5] Hsu, J.L., Liu, C.C., Chen, A.L.P.: Discovering Nontrivial Repeating Patterns in Music Data. IEEE Transactions on Multimedia (2001)
[6] Koh, J.L., Yu, W.D.C.: Efficient Feature Mining in Music Objects. In: Mayr, H.C., Lazanský, J., Quirchmayr, G., Vogel, P. (eds.) DEXA 2001. LNCS, vol. 2113, p. 221. Springer, Heidelberg (2001)
[7] Koh, J.L., Kung, Y.T.: An Efficient Approach for Mining Top-K Fault-Tolerant Repeating Patterns. In: Li Lee, M., Tan, K.-L., Wuwongse, V. (eds.) DASFAA 2006. LNCS, vol. 3882, pp. 95–110. Springer, Heidelberg (2006)
[8] Li, H., Lee, S., Shan, M.K.: Online Mining (Recently) Maximal Frequent Itemsets over Data Streams. In: Proc. of RIDE-SDMA 2005 (2005)
[9] Lin, C.H., Chiu, D.Y., Wu, Y.H., Chen, A.L.P.: Mining Frequent Itemsets from Data Streams with a Time-Sensitive Sliding Window. In: Proc. SIAM International Conference on Data Mining (2005)
[10] Liu, N.-H., Wu, Y.-H., Chen, A.L.P.: An Efficient Approach to Extracting Approximate Repeating Patterns in Music Databases. In: Zhou, L.-z., Ooi, B.-C., Meng, X. (eds.) DASFAA 2005. LNCS, vol. 3453, pp. 240–252. Springer, Heidelberg (2005)
[11] Manku, G.S., Chen Motwani, R.: Approximate Frequent Counts over Data Streams. In: Proc. of the 28th International Conference on Very Large Database (2002)
[12] Wand, H., Fan, W., Yu, P.S., Han, J.: Mining Concept Drifting Data Streams using Ensemble Classifiers. In: Proc. the 9th ACM International Conference on Knowledge Discovery and Data Ming (2003)

A Graph-Based Approach for Sentiment Sentence Extraction

Kazutaka Shimada, Daigo Hashimoto, and Tsutomu Endo

Department of Artificial Intelligence, Kyushu Institute of Technology
680-4 Iizuka Fukuoka 820-8502 Japan
{shimada,d_hashimoto,endo}@pluto.ai.kyutech.ac.jp

Abstract. As the World Wide Web rapidly grows, a huge number of online documents are easily accessible on the Web. We obtain a huge number of review documents that include user's opinions for products. To classify the opinions is one of the hottest topics in natural language processing. In general, we need a large amount of training data for the classification process. However, construction of training data by hand is costly. In this paper, we examine a method of sentiment sentence extraction. This task is to classify sentences in documents into opinions and non-opinions. For the task, we use the Hierarchical Directed Acyclic Graph (HDAG) proposed by Suzuki et al. We obtained high accuracy in the sentiment sentence extraction task. The experimental result shows the effectiveness of the method based on the HDAG.

Keywords: Sentiment Analysis, Sentiment Sentence Extraction, Graph-based Approach, Hierarchical Directed Acyclic Graph, Similarity.

1 Introduction

As the World Wide Web rapidly grows, a huge number of online documents are easily accessible on the Web. Finding information relevant to user needs has become increasingly important. The most important information on the Web is usually contained in the text. We obtain a huge number of review documents that include user's opinions for products. When buying products, users usually survey the product reviews. More precise and effective methods for evaluating the products are useful for users. To classify the opinions is one of the hottest topics in natural language processing. Many researchers have recently studied extraction and classification of opinions [8, 11, 12, 16, 17].

There are many research areas for sentiment analysis; extraction of sentiment expressions, identification of sentiment polarity of sentences, classification of review documents and so on. In this paper, we focus on sentiment sentence extraction. Extraction of sentiment expressions or sentiment sentences is one of the most important tasks in the sentiment analysis because classification tasks usually need a large amount of training data to generate a high accuracy classifier. There are several reports for classification of sentences [9, 11]. However, the purpose of these studies is to classify sentences into positive and negative opinions. Our purpose in this paper is to classify sentences into opinions and

S. Chawla et al. (Eds.): PAKDD 2008 Workshops, LNAI 5433, pp. 38–48, 2009.

non-opinions. Touge et al. [15] and Kawaguchi et al. [7] have proposed methods for opinion extraction. However, these approaches essentially need a large amount of training data for the process. Construction of training data by hand is costly. Kaji and Kitsuregara have reported a method of acquisition of sentiment sentences in HTML documents [5]. The method required only several rules by hand and obtained high accuracy. Also they have proposed a method for building lexicon for sentiment analysis [6]. The knowledge extracted from the Web by using the proposed methods contains the huge quantities of words and sentences. Takamura et al. also have reported a method for extracting polarity of words [14]. These dictionaries are versatile and valuable for users because they do not depend on a specific domain. Here, assume that we need to construct a system for a domain. In that case, we often desire domain-specific knowledge for the system. Therefore, we need to efficiently extract sentiment sentences, which depend on a particular domain or topic. For the process, we need not only pattern-rules or grammatical information to identify sentiment sentences but also surface information of sentences of target domains.

In this paper, we propose a method of sentiment sentence extraction. The method can deal with grammatical information and surface information of sentences. Also our method does not require dictionaries of sentiment expressions. It uses several sample sentences for the extraction process. In the process, we compute a similarity between the sample sentences and target sentences. For the similarity calculation, we employ the graph-based approach, called Hierarchical Directed Acyclic Graph (HDAG), which has been proposed by Suzuki et al [13].

In Section 2, we explain the HDAG data structure and layers. In Section 3, we describe a sentiment sentence extraction process with similarity calculation based on the HDAG. In Section 4, we evaluate the performance of the method and conclude this paper in Section 5.

2 A Graph-Based Data Structure

In this section, we explain a graph-based data structure to compute a similarity.

2.1 Hierarchical Directed Acyclic Graph

In natural language processing, bag-of-words representation is the most general way to express features of a sentence for the similarity calculation. However, it is insufficient to represent the features of a sentence because of lack of relations between words. To solve this problem, many researchers have proposed new approaches: a string kernel [10], a word-sequence kernel [1] , an extended string subsequence kernel [3] and a tree kernel [2]. These kernels are usually more effective as compared with bag-of-words based methods. However, they are not the best representation for deep and complex features, such as semantic or grammatical information, in a sentence because they are somewhat of a simple representation.

To solve the problems, Suzuki et al. have reported a new graph-based approach, called Hierarchical Directed Acyclic Graph kernels (HDAG) [13]. The

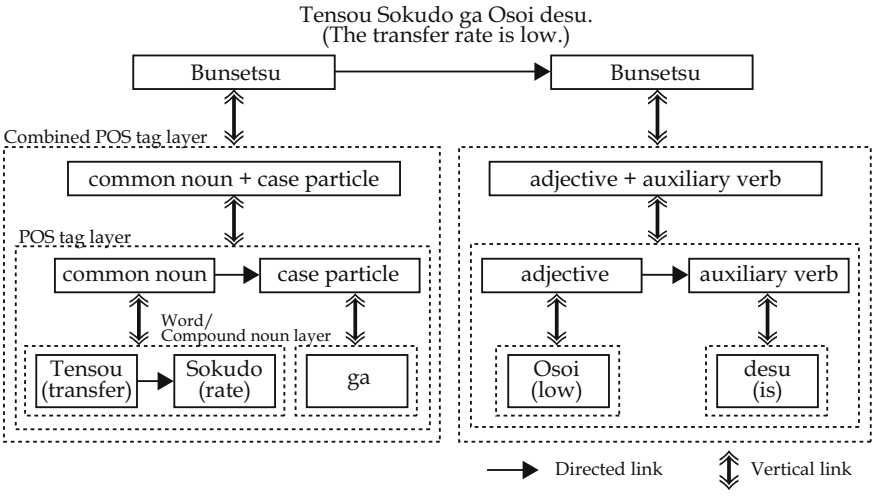


Fig. 1. An example of an HDAG expression

method can handle many linguistic features in a sentence and includes characteristics of tree and sequence kernels. The HDAG is a hierarchized graph-in-graph structure. It represents semantic or grammatical information in a sentence. In this paper, we use the HDAG structure for the sentiment sentence extraction. We compute a similarity between HDAGs generated from sentences. See [13] for more information about the HDAG.

2.2 Layer

Layers in the HDAG denote semantic or grammatical information in a sentence. To compute similarity between sentences correctly, we add new layers to the original and naive HDAG. The HDAG in this paper consists of three layers as follows:

- **Combined POS tag layer**
  This layer consists of part of speech tags of words. We unify the POS tags of words in a bunsetsu[1] into one node.
- **POS tag layer**
  This layer consists of the POS tags of each word or each compound noun.
- **Word/Compound noun layer**
  This layer contains two roles; the layer for words and compound nouns. The layer for each word contains the surface expression of a word. We can use the surface information for calculation of similarity by adding this layer. The 2nd role is handling compound nouns in bunsetsus. This layer often resolves a problem of difference between surface expressions. We unify nouns belonging to a compound noun and then dispose it under the POS node of its compound nouns. For example, we flexibly treat the difference of the following

[1] A bunsetsu is a linguistic unit in Japanese. It usually consists of one content word and its function words.

expressions in similarity calculation by adding this layer:"file downloading software", "downloading software" and "software".

Figure 1 shows an example of an HDAG expression in this paper. In the HDAG, the elements, such as "Bunsetsu" and "Common noun", in each rectangle are the attributes of each node. The directed links are a kind of the dependency relation between elements. The double-headed arrows denote the link between a node and a sub-graph enclosed with a dashed line.

3 Similarity Calculation

In this section, we explain a method of similarity calculation based on the HDAG structure. First, we describe a conversion method of sentences into the HDAGs. Next, we explain an extraction method of hierarchical attribute subsequences from HDAG structures for the similarity calculation. Finally, we introduce a method of similarity calculation and the extraction process using it.

3.1 Preprocessing

There are two processes as the preprocessing for similarity calculation; conversion and extraction of hierarchical attribute subsequences. First we explain the conversion process. To convert sentences into the HDAG structure, we need to analyze them, that is morphological analysis and dependency analysis. In this paper we use JUMAN[2] as the morphological analyzer and KNP[3] as the dependency analyzer. Figure 2 shows the graph structure[4] generated from the sentence "Sokudo ga Osoi desu. (The transfer rate is low.)[5]."

Next we need to extract hierarchical attribute subsequences for the similarity calculation. A hierarchical attribute subsequence is an attribute list with hierarchical structures. The similarity is computed from corresponding hierarchical attribute subsequences extracted from sentences that we want to compare.

Here Suzuki et al. [13] introduced two factors; β and λ. The β ($\beta > 0$) is the factor for the correspondence. The value of each hierarchical attribute sequence is multiplied by $\sqrt{\beta}^m$ where m represents the number of attributes in the hierarchical attribute sequence. The λ is the decay factor λ ($0 \leq \lambda \leq 1$). The system allows not only exactly matching structures but also similar structures by using this factor. The actual decay value of a skipping node v_i is $\Lambda(v_i) = \lambda^{n+1}$ where n is the number of nodes in a graph G if vertical link exists, or $\Lambda(v) = \lambda$ otherwise.

Figure 3 shows an example of hierarchical attribute subsequences and the factors. In the figure, a dependency relation and a hierarchical relation are

[2] `http://nlp.kuee.kyoto-u.ac.jp/nl-resource/juman.html`

[3] `http://nlp.kuee.kyoto-u.ac.jp/nl-resource/knp.html`

[4] In this simplified explanation, the graph structure for an example is expressed without the layers described in the previous section.

[5] Sokudo is a noun (transfer rate), ga is a case particle, Osoi is an adjective (low), and Desu is an auxiliary verb (is).

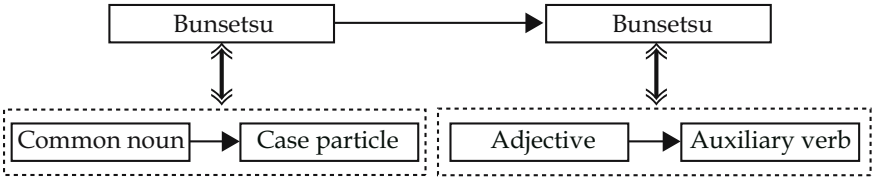


Fig. 2. An example of a graph structure

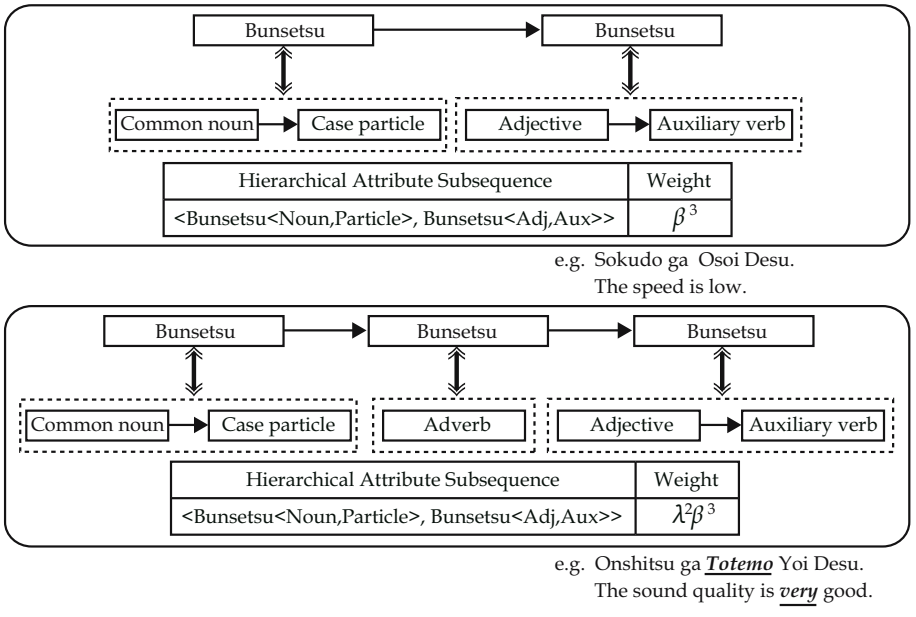


Fig. 3. An example of extraction of corresponding hierarchical attribute subsequences and the weight

expressed by using a comma and a nested structure, respectively. For the two HDAGs, the hierarchical attribute subsequence <Bunsetsu<Noun,Particle>, Bunsetsu<Adj,Aux>> appears in both sentences. Since the number of attributes in the hierarchical attribute subsequence is 6, the value of β is $\sqrt{\beta}^6 = \beta^3$. The hierarchical attribute subsequence of the 2nd sentence in the figure is generated by skipping a node, Adverb. Therefore, the weight contains $\lambda^{1+1} = \lambda^2$. These weights are used in the similarity calculation process.

3.2 Similarity between Two Sentences

Next, we compute a similarity between two HDAG structures. First, we search the common hierarchical attribute subsequences between HDAGs (See Figure 3). Then we multiply the weight values of them. For example, the correspondence of them in Figure 3 is $\lambda^2\beta^6$. Finally, we divide the sum total of correspondence

values by the product of the numbers of bunsetsus of the two sentences. We handle this value as the similarity between them.

Here we consider the factor β. Suzuki et al. [13] defined the range of the β as $0 < \beta \leq 1$. However, we define the range as $\beta > 0$ in this paper. Also we categorize the range into two types; $0 < \beta \leq 1$ and $\beta > 1$. Our method computes the similarity focusing on structural information if $\beta \leq 1$. If the β is more than 1, it computes the similarity focusing on surface expressions. This is due to layers that we constructed. In our layers, the word layer and compound noun layer are lower layer than the structural layer, i.e., the POS tag layer. Therefore surface expressions are treated as important element in the case that $\beta > 1$ because the elements in deeper layers possess high weight values. We apply these two types of the parameter β into our method.

3.3 Sentence Extraction

In this subsection, we explain the sentence extraction process based on the HDAG and the similarity calculation. The process is as follows:

1. prepare sample sentences as seeds for similarity calculation,
2. compute the similarity between each seed and target sentences,
3. extract n-best lists of each seed as sentiment sentence lists,
4. combine n-best lists obtained by two different parameters of β.

For the combination in the last step, we compare two strategies.

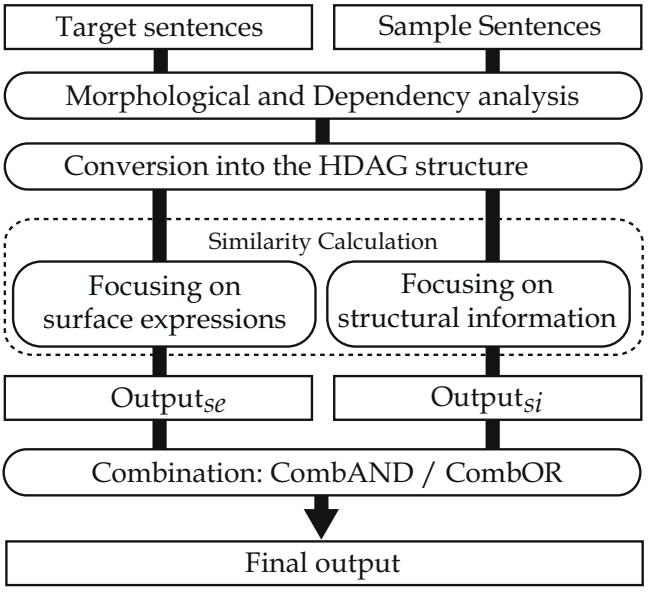


Fig. 4. The outline of the sentence extraction process

CombAND: We extract the intersection of each n-best list as the output.
CombOR: We extract the union of each n-best list as the output.

Figure 4 shows the outline of the extraction process.

4 Experiment

In this section we evaluated the proposed method with a review document set.

4.1 Dataset and Criteria

We used review documents of a portable audio player[6] posted in the bulletin board system of kakaku.com[7]. We extracted 1052 Japanese sentences from the review documents. The dataset consists of 610 sentiment sentences and 442 non-sentiment sentences. For the experiment, we prepared 10 sample sentences as seeds for the sentence extraction process. All the seed sentences in this experiment were sentiment sentences. We generated the seed sentences on the basis of some evaluation criteria which were mentioned in the review documents; e.g., "design of the product", "Sound quality" and so on.

In this experiment, we set $\lambda = 0.5$. Also we set $\beta = 0.5$ as the parameter for focusing on structural information and $\beta = 1.5$ as the parameter for focusing on surface expressions. The number of sentences we extracted in this experiment is 5 for each seed sentence, that is 5-best list. In other words, we extracted the top 5 sentences that possessed high similarity as the sentiment sentences that were estimated from each sample sentences. We did not employ any thresholds for the similarity in the extraction process.

We used the following three criteria for this evaluation.

- $Sent_{real}$: This criterion is the number of sentiment sentences extracted correctly from target sentences.
- $Sent_{non}$: This criterion is the number of non-sentiment sentences extracted from target sentences.
- Acc: This criterion is the accuracy computed from $Sent_{real}$ and $Sent_{non}$.

$$Acc = \frac{Sent_{real}}{Sent_{real} + Sent_{non}}$$

Note that we omitted same sentences in the output from the proposed method when we counted $Sent_{real}$ and $Sent_{non}$ in this experiment.

4.2 Results

Table 1 shows the experimental result. In the table, the BOW denotes a similarity calculation method based on the COS measure and bag-of-words features.

[6] SONY Walkman NW-A808.
[7] `http://www.kakaku.com/`

Table 1. The experimental result

	$Sent_{real}$	$Sent_{non}$	Acc
Structural information	32	4	0.889
Surface expressions	43	4	0.915
CombAND	22	**1**	**0.957**
CombOR	**53**	7	0.883
BOW (Baseline)	42	7	0.857

Table 2. The extracted sentences (translated into English)

Rank	SI ($\beta \leq 1$)	SE ($\beta > 1$)
1	The sound quality is barely good.	The sound quality is barely good.
2	The display is easily viewable.	The sound quality is wonderful.
3	The machine body is somewhat heavy.	The sound quality is great.

This is a baseline in this experiment. The accuracy rates of each approach in our method outperformed the baseline method based on BOW features. Our methods obtained high accuracies even without combinations, namely **CombAND** and **CombOR**. In addition, the method focusing on surface expressions (SE) outperformed the method focusing on structural information (SI) in terms of all criteria. Table 2 shows the top 3 sentences extracted from target sentences in the case that the seed sentence was "The sound quality is good.".

For the combinations, the accuracy of the **CombOR** was the lowest of the methods although the number of sentiment sentences extracted correctly was the best of them. On the other hand, the accuracy of the **CombAND** produced the best performance. Although the number of extracted sentences with the **CombAND** drastically decreased, the output possessed high reliability.

Besides, our method usually obtained long sentences as compared with seed sentences. The average lengths of seed sentences and output sentences were 5.5 and 9.1 words respectively. The maximum length in the output sentences was 27 words. This result shows that our method can extract great variety of sentiment sentences. The following sentences are the instances of seed sentences and their output sentences:

Seed$_1$: I am almost satisfied with this product (Zentai-teki ni manzoku dekiru seihin desu).

Output$_1$: I think that a good thing equipped with the function that I want was released (Yatto watashi ga nozomu kinou ga zyuzitusita yoi mono ga deta to iu kanzi desu).

Seed$_2$: It is difficult to push the play button (Saisei botan ga oshi nikui desu).

Output$_2$: It is not convenient for WinMediaPlayer users to use attached music file transfer software XYZ (Sen-you no ongaku fairu tensou sohuto XYZ ha WinMediaPlayer kara no norikae niha tukai durai desu).

5 Discussion and Conclusions

In this paper, we proposed a method of sentiment sentence extraction based on a graph-based approach, called Hierarchical Directed Acyclic Graph. Our method can extract sentiment sentences with several sample sentences. We obtained high accuracy in the experiment. However, the number of extracted sentences was not enough, that is the recall rate was low (less than 10%).

One of the solutions of this problem is to apply a bootstrapping approach into our method. We might acquire more sentiment sentences by adding the extracted sentences as new seeds for the extraction process because the accuracy of CombAND was extremely high. To use CombOR is one of the ideas to extract large quantities of sentences in the case that the number of sentences extracted in the bootstrapping process is saturated, i.e., the final step of the bootstrapping approach. Another approach to improve the recall rate is use of the extracted sentences for the training data of the sentiment classification task. Wiebe and Riloff [18] have proposed a method for creating subjective and objective classifiers from unannotated texts. They used some rules for constructing initial training data. Then they used the data for generating a classifier. We think that the outputs from our method also can be used for the training data of a classifier for this sentiment sentence classification task.

The value of n of the n-best list is one of the most important factors for the improvement of the recall rate although the accuracy decreases. We evaluated our method with different n values: $n = 10$ and $n = 15$ (The original value was 5). Table 3 shows the experimental result. The accuracy rates decreased in the case that the n became large. We need to discuss the appropriate number of sentences that we extract in our method.

Our method depends on seed sentences. If the seed sentences are changed, the accuracy also changes. We examined other seed sentences after the experiment in the previous section. As a result, the accuracy fluctuated; approximately $\pm 5\%$. To generate appropriate seed sentences is one of the most important tasks for our method.

In the previous section, we evaluated our method with fixed parameters. However, these are not always the best parameter values. The parameters β and λ are important factors for the similarity calculation. We compared several values of these parameters. Tables 4 and 5 show the experimental results. The best

Table 3. The n and accuracy

	$n = 10$			$n = 15$		
	$Sent_{real}$	$Sent_{non}$	Acc	$Sent_{real}$	$Sent_{non}$	Acc
Structural information	54	15	0.783	77	22	0.778
Surface expressions	74	14	0.841	103	26	0.798
CombAND	34	6	0.850	54	11	0.831
CombOR	94	23	0.803	126	37	0.773

Table 4. The best parameter for SI

		β					Ave
		0.1	0.3	0.5	0.7	0.9	
λ	0.1	0.867	0.897	0.826	0.875	0.896	0.872
	0.3	0.875	0.867	0.881	0.891	0.851	0.873
	0.5	0.833	0.800	0.889	0.886	0.870	0.856
	0.7	0.800	0.875	0.920	0.897	0.907	**0.880**
	0.9	0.833	0.857	0.727	0.938	**0.950**	0.861
	1.0	0.833	0.857	0.778	0.926	0.919	0.863
Ave		0.840	0.859	0.837	**0.902**	0.899	0.867

Table 5. The best parameter for SE

		β					Ave
		1.1	1.3	1.5	1.7	1.9	
λ	0.1	0.896	0.896	0.896	0.896	0.896	0.896
	0.3	0.915	0.917	0.915	0.915	0.915	0.915
	0.5	0.894	0.915	0.915	0.894	0.913	0.906
	0.7	0.935	0.891	0.891	0.872	0.896	0.897
	0.9	0.930	0.933	0.957	0.938	0.938	0.939
	1.0	**0.975**	0.932	0.956	0.957	0.957	**0.955**
Ave		**0.924**	0.914	0.922	0.912	0.919	0.918

accuracy rates of the SI and the SE in this dataset were 0.950 ($\lambda = 0.9$, $\beta = 0.9$, $Sent_{real} = 38$, $Sent_{non} = 2$) and 0.975 ($\lambda = 1.0$, $\beta = 1.1$, $Sent_{real} = 39$, $Sent_{non} = 1$) respectively. However, these values depend on the dataset in the experiment. Although we evaluated these parameters with another dataset, the method with the parameters did not produce the best accuracy. Therefore we need to consider the automatic determination of these parameters. The average accuracy rates for the dataset in the previous section were 0.867 for the SI and 0.918 for the SE respectively. The standard deviation values were 0.048 for the SI and 0.025 for the SE. These results show that our method provides high and stable accuracy.

Our future work includes (1) evaluation of our method in a large-scale dataset and other datasets, (2) improvement of the accuracy by adding other layers to the HDAG structure, such as semantic features of words [4], and (3) construction of a sentiment sentence maintenance tool based on this approach.

References

[1] Cancedda, N., Gaussier, E., Goutte, C., Renders, J.M.: Word-sequence kernels. J. Machine Learning Research 3, 1059–1082 (2003)

[2] Collins, M., Duffy, N.: Convolution kernels for natural language. In: Advances in Neural Information Processing Systems, vol. 14 (2002)

[3] Hirao, T., Suzuki, J., Isozaki, H., Maeda, E.: Dependency-based sentence alignment for multiple document summarization. In: Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004) (2004)
[4] Ikehara, S., Miyazaki, M., Shirai, S., Yokoo, A., Nakaiwa, H., Ogura, K., Ooyama, Y., Hayashi, Y. (eds.): Goi-Taikei. A Japanese Lexicon (in Japanese). Iwanami Shoten (1997)
[5] Kaji, N., Kitsuregawa, M.: Automatic construction of polarity-tagged corpus from html documents. In: Proceedings of the 21st International Conference on Computational Linguistics (COLING/ACL 2006), pp. 452–459 (2006)
[6] Kaji, N., Kitsuregawa, M.: Building lexicon for sentiment analysis from massive html documents. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-CoNLL 2007) (2007)
[7] Kawaguchi, T., Matsui, T., Ohwada, H.: Opinion extraction from weblog using svm and newspaper article (in Japanese). In: The 20th Annual Conference of the Japanese Society for Artificial Intelligence (2006)
[8] Kobayashi, N., Iida, R., Inui, K., Matsumoto, Y.: Opinion extraction using a learning-based anaphora resolution technique. In: Proceedings of the Second International Joint Conference on Natural Language Processing (IJCNLP 2005), pp. 175–180 (2005)
[9] Kudo, T., Matsumoto, Y.: A boosting algorithm for classification of semi-structured text. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP) (2004)
[10] Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., Watkins, C.: Text classification using string kernel. J. Machine Learning Research 2, 419–444 (2002)
[11] Osajima, I., Shimada, K., Endo, T.: Classification of evaluative sentences using sequential patterns. In: Proceedings of the 11nd Annual Meeting of The Association for Natural Language Processing (in Japanese) (2005)
[12] 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 (EMNLP), pp. 79–86 (2002)
[13] Suzuki, J., Sasaki, Y., Maeda, E.: Hierarchical directed acyclic graph kernel. Systems and Computers in Japan 37(10), 58–68 (2006)
[14] Takamura, H., Inui, T., Okumura, M.: Extracting semantic orientations of words using spin model. In: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005), pp. 133–140 (2005)
[15] Touge, Y., Ohashi, K., Yamamoto, K.: Extracting opinion sentence adapted to topic using iteration learning (in Japanese). In: IPSJ SIG Notes, pp. 43–50 (2004)
[16] Tsutsumi, K., Shimada, K., Endo, T.: Movie review classification based on a multiple classifier. In: the 21th Pacific Asia Conference on Language, Information and Computation (PACLIC) (2007)
[17] Turney, P.D.: Thumbs up? or thumbs down? semantic orientation applied to unsupervised classification of reviews. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, pp. 417–424 (2002)
[18] Wiebe, J., Riloff, E.: Creating subjective and objective sentence classifiers from unannotated texts. In: Gelbukh, A. (ed.) CICLing 2005. LNCS, vol. 3406, pp. 486–497. Springer, Heidelberg (2005)

Fuzzy Weighted Association Rule Mining with Weighted Support and Confidence Framework

Maybin Muyeba[1], M. Sulaiman Khan[2], and Frans Coenen[3]

[1] Dept. of Computing, Manchester Metropolitan University, Manchester, M1 5GD, UK
[2] School of Computing, Liverpool Hope University, Liverpool, L16 9JD, UK
[3] Dept. of Computer Science, University of Liverpool, Liverpool, L69 3BX, UK
M.Muyeba@mmu.ac.uk, khanm@hope.ac.uk, frans@csc.liv.ac.uk

Abstract. In this paper we extend the problem of mining weighted association rules. A classical model of boolean and fuzzy quantitative association rule mining is adopted to address the issue of invalidation of downward closure property (DCP) in weighted association rule mining where each item is assigned a weight according to its significance w.r.t some user defined criteria. Most works on DCP so far struggle with invalid downward closure property and some assumptions are made to validate the property. We generalize the problem of downward closure property and propose a fuzzy weighted support and confidence framework for boolean and quantitative items with weighted settings. The problem of invalidation of the DCP is solved using an improved model of weighted support and confidence framework for classical and fuzzy association rule mining. Our methodology follows an Apriori algorithm approach and avoids pre and post processing as opposed to most weighted ARM algorithms, thus eliminating the extra steps during rules generation. The paper concludes with experimental results and discussion on evaluating the proposed framework.

Keywords: Association rules, fuzzy, weighted support, weighted confidence, downward closure.

1 Introduction

The task of mining Association Rules (ARs) [11] is mainly to discover association rules (with strong support and high confidence) in large databases. Classical Association Rule Mining (ARM) deals with the relationships among the items present in transactional databases [9, 10] consisting of binary (boolean) attributes. The typical approach is to first generate all large (frequent) itemsets (attribute sets) from which the set of ARs is derived. A large itemset is defined as one that occurs more frequently in the given data set than a user supplied support threshold. To limit the number of ARs generated a confidence threshold is used. The number of ARs generated can therefore be influence by careful selection of the support and confidence thresholds, however great care must be taken to ensure that itemsets with low support, but from which high confidence rules may be generated, are not omitted.

S. Chawla et al. (Eds.): PAKDD 2008 Workshops, LNAI 5433, pp. 49–61, 2009.

Given a set of items $I = \{i_1, i_2, ...i_m\}$ and a database of transactions $D = \{t_1, t_2, ...t_n\}$ where $t_i = \{I_{i1}, I_{i2}, ...I_{ip}\}$, $p \leq m$ and $I_{ij} \in I$, if $X \subseteq I$ with K = |X| is called a k-itemset or simply an itemset. Let a database D be a multi-set of subsets of I as shown. Each $T \in D$ supports an itemset $X \subseteq I$ if $X \subseteq T$ holds. An association rule is an expression X → Y, where X, Y are item sets and $X \cap Y = \varnothing$ holds. Number of transactions T supporting an item X w.r.t D is called support of X, $Supp(X) = |\{T \in D \mid X \subseteq T\}| / |D|$. The strength or confidence (c) for an association rule X → Y is the ratio of the number of transactions that contain X U Y to the number of transactions that contain X, Conf (X → Y) = Supp (X U Y)/ Supp (X).

For non-boolean items fuzzy association rule mining was proposed using fuzzy sets such that quantitative and categorical attributes can be handled [12]. A fuzzy quantitative rule represents each item as (item, value) pair. Fuzzy association rules are expressed in the following form:

If X is A satisfies Y is B

e.g. if (age is young) ➔ (salary is low).

Given a database *T*, attributes *I* with itemsets $X \subset I, Y \subset I$ and $X = \{x_1, x_2, ...x_n\}$ and $Y = \{y_1, y_2, ...y_n\}$ and $X \cap Y = \varnothing$, we can define fuzzy sets $A = \{fx_1, fx_2, ..., fx_n\}$ and $B = \{fx_1, fx_2, ..., fx_n\}$ associated to X and Y respectively. For example (X, Y) could be (age, young), (age, old), (salary, high) etc. The semantics of the rule is that when the antecedent "X is A" is satisfied, we can imply that "Y is B" is also satisfied, which means there are sufficient records that contribute their votes to the attribute fuzzy set pairs and the sum of these votes is greater than the user specified threshold.

However, the above ARM frameworks assume that all items have the same significance or importance i.e. their weight within a transaction or record is the same (weight=1) which is not always the case. For example, [wine → salmon, 1%, 80%] may be more important than [bread → milk, 3%, 80%] even though the former holds a lower support of 1%. This is because those items in the first rule usually come with more profit per unit sale, but the standard ARM simply ignores this difference.

Weighted ARM deals with the importance of individual items in a database [2, 3, 4]. For example, some products are more profitable or may be under promotion, therefore more interesting as compared to others, and hence rules concerning them are of greater value.

Table 1. Weigted Items Database

ID	Item	Profit	Weight	…
1	Scanner	10	0.1	…
2	Printer	30	0.3	…
3	Monitor	60	0.6	…
4	Computer	90	0.9	…

Table 2. Transactions

TID	Items
1	1,2,4
2	2,3
3	1,2,3,4
4	2,3,4

In table 1, items are assigned weights (*w*) based on their significance. These weights may be set according to an item's profit margin. This generalized version of ARM is called Weighted Association Rule Mining (WARM). From table 1, we can see that the rule Computer → Printer is more interesting than Computer → Scanner because the profit of a printer is greater than that of a scanner. The main challenge in weighted ARM is that "downward closure property" doesn't hold, which is crucial for efficient iterative process of generating and pruning frequent itemsets from subsets.

In this paper we address the issue of downward closure property (DCP) in WARM. We generalize and solve the problem of DCP and propose a weighted support and confidence framework for datasets with boolean and quantitative items for classical and fuzzy WARM (FWARM). We evaluate our proposed framework with experimental results.

The paper is organised as follows: section 2 presents background and related work; section 3 & 4 give problem definitions 1 & 2 respectively; section 5 details weighted downward closure property; section 6 presents FWARM algorithm, section 7 reviews experimental results and section 8 concludes the paper with directions for future work.

2 Background and Related Work

In classical ARM, data items are viewed as having equal importance but recently some approaches generalize this where items are given weights to reflect their significance to the user [4]. The weights may correspond to special promotions on some products or the profitability of different items etc. Currently, two approaches exist: pre- and post-processing. Post processing solves first the non-weighted problem (weights=1 per item) and then prunes the rules later. Pre-processing prunes the non-frequent itemsets earlier by considering their weights in each database scan. The issue in post-processing weighted ARM is that first; items are scanned without considering their weights. Finally, the rule base is checked for frequent weighted ARs. This gives us a very limited itemset pool for weighted ARs and may miss many potential itemsets. In pre-processing, fewer rules are obtained as compared to post processing because many potential frequent super sets are missed.

In [2] a post-processing model is proposed. Two algorithms were proposed to mine itemsets with normalized and un-normalized weights. The K-support bound metric was used to ensure validity of the closure property. Even that didn't guarantee every subset of a frequent set being frequent unless the k-support bound value of (K-1) subset was higher than (K).

An efficient mining methodology for Weighted Association Rules (WAR) is proposed in [3]. A Numerical attribute was assigned for each item where the weight of the item was defined as part of a particular weight domain. For example, soda[4,6] → snack[3,5] means that if a customer purchases soda in the quantity between 4 and 6 bottles, he is likely to purchase 3 to 5 bags of snacks. WAR uses a post-processing approach by deriving the maximum weighted rules from frequent itemsets. Post WAR doesn't interfere with the process of generating frequent itemsets but focuses on how weighted AR's can be generated by examining weighting factors of items included in generated frequent itemsets.

Similar techniques were proposed for weighted fuzzy quantitative association rule mining [5, 7, 8]. In [6], a two-fold pre processing approach is used where firstly, quantitative attributes are discretised into different fuzzy linguistic intervals and weights assigned to each linguistic label. A mining algorithm is applied then on the resulting dataset by applying two support measures for normalized and un-normalized cases. The closure property is addressed by using the z-potential frequent subset for each candidate set. An arithmetic mean is used to find the possibility of frequent k+1itemset, which is not guaranteed to validate the downward closure property.

Another significance framework that handles the DCP problem is proposed in [1]. Weighting spaces were introduced as inner-transaction space, item space and transaction space, in which items can be weighted depending on different scenarios and mining focus. However, support is calculated by only considering the transactions that contribute to the itemset. Further, no discussions were made on interestingness issue of the rules produced.

In this paper we present a fuzzy weighted support and confidence framework to mine weighted boolean and quantitative data (by fuzzy means) to address the issue of invalidation of downward closure property. We then show that using the proposed framework, rules can be generated efficiently with a valid downward closure property without biases made by pre- or post-processing approaches.

3 Problem Definition One (Boolean)

Let the input data D have transactions $T = \{t_1, t_2, t_3, \cdots, t_n\}$ with a set of items $I = \{i_1, i_2, i_3, \cdots, i_{|I|}\}$ and a set of weights $W = \{w_1, w_2, \cdots, w_{|I|}\}$ associated with each item. Each i^{th} transaction t_i is some subset of I and a weight w is attached to each item $t_i[i_j]$ ("j^{th}" item in the "i^{th}" transaction).

Thus each item i_j will have associated with it a weight corresponding to the set W, i.e. a pair (i, w) is called a weighted item where $i \in I$. Weight for the "j^{th}" item in the "i^{th}" transaction is given by $t_i[i_j[w]]$.

We illustrate the concept and definitions using tables 3 and 4. Table 3 contains transactions for 5 items. Table 4 has corresponding weights associated to each item *i* in *T*. In our definitions, we use sum of votes for each itemset by aggregating weights per item as a standard approach.

Table 3. Transactional Database

T	Items
t_1	A B C D E
t_2	A C E
t_3	B D
t_4	A D E
t_5	A B C D

Table 4. Items with weights

Items *i*	Weights *(IW)*
A	0.1
B	0.3
C	0.6
D	0.9
E	0.7

Definition 1. Item Weight IW is a non-negative real value given to each item i_j ranging [0..1] with some degree of importance, a weight $i_j[w]$.

Definition 2. Itemset Transaction Weight ITW is the aggregated weights (using some aggregation operator) of all the items in the itemset present in a single transaction. Itemset transaction weight for an itemset X is calculated as:

$$\text{vote for } t_i \text{ satisfying } X = \prod_{k=1}^{|X|}{}_{(\forall[i[w]]\in X)}\, t_i[i_k[w]] \tag{1}$$

Itemset transaction weight of itemset (B, D) is calculated as: $ITW(B, D) = 0.3 \times 0.9 = 0.27$.

Definition 3. Weighted Support WS is the aggregated sum of itemset transaction weight (votes) ITW of all the transactions in which itemset is present, divided by the total number of transactions. It is calculated as:

$$WS(X) = \frac{\text{Sum of votes satisfying } X}{\text{Number of records in } T} = \frac{\sum_{i=1}^{n}\prod_{k=1}^{|X|}{}_{(\forall[i[w]]\in X)}\, t_i[i_k[w]]}{n} \tag{2}$$

Weighted Support WS of itemset (B, D) is calculated as:

$$WS(B, D) = \frac{\text{Sum of votes satisfying (B, D)}}{\text{Number of records in } T} = \frac{0.81}{5} = 0.162$$

Definition 4. Weighted Confidence WC is the ratio of sum of votes satisfying both $X \cup Y$ to the sum of votes satisfying X. It is formulated (with $Z = X \cup Y$) as:

$$WC(X \rightarrow Y) = \frac{WS(Z)}{WS(X)} = \sum_{i=1}^{n}\frac{\prod_{k=1}^{|Z|}{}_{(\forall[z[w]]\in Z)}\, t_i[z_k[w]]}{\prod_{k=1}^{|X|}{}_{(\forall[i[w]]\in X)}\, t_i[x_k[w]]} \tag{3}$$

Weighted Confidence WC of itemset (B, D) is calculated as:

$$WC(B, D) = \frac{WS(Z)}{WS(X)} = \frac{WS(X \cup Y)}{WS(X)} = \frac{WS(B \cup D)}{WS(B)} = \frac{0.16}{0.18} = 0.89$$

4 Problem Definition Two (Quantitative/Fuzzy)

Let a dataset D consists of a set of transactions $T = \{t_1, t_2, t_3, \cdots, t_n\}$ with a set of items $I = \{i_1, i_2, i_3, \cdots, i_{|I|}\}$. A fuzzy dataset D' consists of fuzzy transactions

$T' = \{t_1', t_2', t_3', ..., t_n'\}$ with fuzzy sets associated with each item in I, which is identified by a set of linguistic *labels* $L = \{l_1, l_2, l_3, ..., l_{|L|}\}$ (for example $L = \{small, medium, large\}$). We assign a weight w to each l in L associated with i. Each attribute $t_i'[i_j]$ is associated (to some degree) with several fuzzy sets.

Table 5. Fuzzy Transactional Database

TID	X		Y	
	Small	Medium	Small	Medium
1	0.5	0.5	0.2	0.8
2	0.9	0.1	0.4	0.6
3	1.0	0.0	0.1	0.9
4	0.3	0.7	0.5	0.5

Table 6. Fuzzy Items with weights

Fuzzy Items *i[l]*	Weights *(IW)*
(X, Small)	0.9
(X, Medium)	0.7
(Y, Small)	0.5
(Y, Medium)	0.3

The degree of association is given by a *membership degree* in the range $[0..1]$, which indicates the correspondence between the value of a given $t_i'[i_j]$ and the set of *fuzzy linguistic labels*. The "k^{th}" weighted fuzzy set for the "j^{th}" item in the "i^{th}" fuzzy transaction is given by $t_i'[i_j[l_k[w]]]$. Thus each label l_k for item i_j would have associated with it a weight, i.e. a pair $([i[l]], w)$ is called a weighted item where $[i[l]] \in L$ is a label associated with i and $w \in W$ is weight associated with label l.

We illustrate the fuzzy weighted ARM concept and definitions using tables 5 and 6. Table 5 contains transactions for 2 quantitative items discretised into two overlapped intervals with fuzzy values. Table 4 has corresponding weights associated to each fuzzy item *i[l]* in *T*.

Definition 5. Fuzzy Item Weight FIW is a value attached with each fuzzy set. It is a non-negative real number value range $[0..1]$ w.r.t some degree of importance (table 6). Weight of a fuzzy set for an item i_j is denoted as $i_j[l_k[w]]$.

Definition 6. Fuzzy Itemset Transaction Weight $FITW$ is the aggregated weights of all the fuzzy sets associated to items in the itemset present in a single transaction. Fuzzy Itemset transaction weight for an itemset *(X, A)* is calculated as:

$$\text{vote for } t_i' \text{ satisfying } X = \prod_{k=1}^{|L|} {}_{(\forall [i[l[w]]] \in X)}\, t_i'[i_j[l_k[w]]] \qquad (4)$$

Let's take an example of itemset <(X, Medium), (Y, Small)> denoted by (X, Medium) as *A* and (Y, Small) as *B*. Fuzzy Itemset transaction weight $FITW$ of itemset (A, B) in transaction 1 is calculated as *FITW* (*A*, *B*) = (0.5 × 0.7) × (0.2 × 0.5) = (0.35) × (0.1) = 0.035.

Definition 7. Fuzzy Weighted Support FWS is the aggregated sum of $FITW$ of all the transactions itemset is present, divided by the total number of transactions. It is denoted as:

$$FWS(X) = \frac{\text{Sum of votes satisfying } X}{\text{Number of records in } T} = \frac{\sum_{i=1}^{n}\prod_{k=1}^{|L|}{}_{(\forall[i[l[w]]]\in X)}\, t_i'[i_j[l_k[w]]]}{n} \quad (5)$$

Weighted Support FWS of itemset (A, B) is calculated as:

$$FWS(A,B) = \frac{\text{Sum of votes satisfying (A, B)}}{\text{Number of records in } T} = \frac{0.172}{4} = 0.043$$

Definition 8. Fuzzy Weighted Confidence FWC is the ratio of sum of votes satisfying both $\mathrm{X} \cup \mathrm{Y}$ to the sum of votes satisfying X with $Z = X \cup Y$. It is formulated as:

$$FWC(X \rightarrow Y) = \frac{FWS(Z)}{FWS(X)} = \sum_{i=1}^{n} \frac{\prod_{k=1}^{|Z|}{}_{(\forall[z[w]]\in Z)}\, t_i'[z_k[w]]}{\prod_{k=1}^{|X|}{}_{(\forall[i[w]]\in X)}\, t_i'[x_k[w]]} \quad (6)$$

FWC (A, B) is calculated as: $FWC(A,B) = \frac{WS(Z)}{WS(X)} = \frac{WS(A \cup B)}{WS(A)} = \frac{0.043}{0.227} = 0.19$

5 Downward Closure Property (DCP)

In a classical Apriori algorithm it is assumed that if the itemset is large, then all its subsets should also be large and is called Downward Closure Property (DCP). This helps the algorithm to generate large itemsets of increasing size by adding items to itemsets that are already large. In the weighted ARM case where each item is assigned

Table 7. Frequent itemsets with invalid DCP (weighted settings)

Large Itemsets	Support (40%)	Large?	Weighted Support (0.4)	Large
AB	40%	Yes	0.16	No
AC	60%	Yes	0.42	Yes
ABC	40%	Yes	0.4	Yes
BC	40%	Yes	0.36	No
BD	60%	Yes	0.72	Yes
BCD	40%	Yes	0.72	Yes

a weight, the DCP does not hold. Because of the weighted support, an itemset may be large even though some of its subsets are not large. This violates DCP (see table 7).

Table 7 shows four large itemsets of size 2 (AB, AC, BC, BD) and two large itemsets of size 3 (ABC, BCD), generated using tables 3 and 4. In classical ARM, when the weights are not considered, all of the six itemsets are large. But if we consider item weights and calculate the weighted support of itemsets according to definition 3 and 7, a new set of support values are obtained. In table 7, although the classical support of all itemsets is large, if ABC and BCD are frequent then their subsets must be large according to classical ARM. But considering the weighted support, AB and BC are no longer frequent.

5.1 Weighted Downward Closure Property (DCP)

We now argue that the DCP with boolean and fuzzy data can be validated by using this new weighted framework. We give a proof and an example to illustrate this. Consider figure 1, where items in the transaction are assigned weights and a user defined supports threshold is set to 0.01.

In figure 1, for each itemset, weighted support WS (the number above each itemset) is calculated by using definition 3 and weighted confidence WC (the number on top of each itemset i.e. above weighted support) is calculated by using definition 4. If an itemset weighted support is above the threshold, the itemset is frequent and we mark it with colour background, otherwise it is with white background, meaning that it's not large.

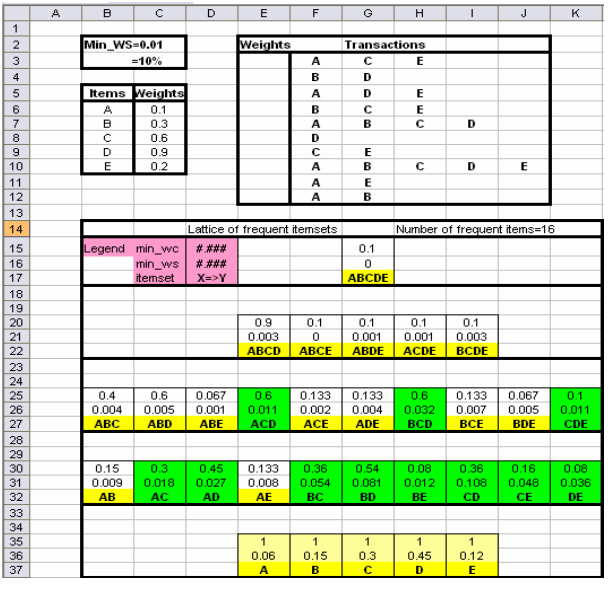


Fig. 1. The lattice of frequent itemsets

It can be noted that if an itemset is with white background i.e. not frequent, then any of its supersets in the upper layer of the lattice can not be frequent. Thus "weighted downward closure property", is valid under the "weighted support" framework. It justifies the efficient mechanism of generating and pruning significance iteratively.

We also briefly prove that the DCP is always valid in the proposed framework. The following lemma applies to both boolean and fuzzy/quantitative data and is stated as:

Lemma. If an itemset is not frequent them its superset cannot be frequent and $WS(subset) \geq WS(superset)$ is always true.

Proof. Given an itemset *X* not frequent i.e. $ws(X) < \min\_ws$. For any itemset Y, $X \subset Y$ i.e. superset of *X*, if a transaction *t* has all the items in Y, i.e. $Y \subset t$, then that transaction must also have all the items in X, i.e. $X \subset t$. We use *tx* to denote a set of transactions each of which has all the items in X, i.e. $\{tx \mid tx \subseteq T, (\forall t \in tx, X \subset t)\}$. Similarly we have $\{tx \mid ty \subseteq T, (\forall t \in ty, Y \subset t)\}$. Since $X \subset Y$, we have $tx \subset ty$. Therefore $WS(tx) \geq WS(ty)$. According to the definition of weighted support, $\mathrm{WS}(X) = \frac{\sum_{i=1}^{n} \prod_{k=1}^{|X|}{}_{(\forall [i[w]] \in X)} t_i[i_k[w]]}{n}$ the denominator stays the same, therefore we have $WS(X) \geq WS(Y)$. Because $ws(X) < \min\_ws$, we get $ws(Y) < \min\_ws$. This then proves that Y is not frequent if its subset is not frequent.

Figure 1 illustrates a concrete example. Itemset AC appears in transaction 1, 5 and 8, therefore the WS (AC) = 0.018. Intuitively, the occurrence of its superset ACE is only possible when AC appears in that transaction. But itemset ACE only appears in transactions 1 and 8, thus WS (ACE) = 0.0024, where WS (ACE) <WS (AC). Summatively, if AC is not frequent, it's superset ACE is impossible to be frequent; hence there is no need to calculate its weighted support.

6 FWARM Algorithm

For fuzzy weighted association rule mining standard ARM algorithms can be used or at least adopted after some modifications. The proposed Fuzzy Weighted ARM (FWARM) algorithm belongs to the *breadth first traversal* family of ARM algorithms, developed using tree data structures [13] and works in a fashion similar to the Apriori algorithm [10].

The FWARM algorithm is given in Table 8. In the Table: C_k is the set of candidate itemsets of cardinality k, w is the set of weights associated to items I. F is the set of frequent item sets, R is the set of potential rules and R' is the final set of generated fuzzy weighted ARs.

Table 8. FWARM Algorithm

Input: T = data set w = itemset weights ws = weighted support wc = weighted confidence
Output: R' = Set of Weighted ARs
1. k = 0; $C_k = \varnothing$; $F_k = \varnothing$ 2. C_k = Set of 1 item sets 3. $k \Leftarrow 1$ 4. Loop 5. if $C_k = \varnothing$ break 6. $\forall c \in C_k$ 7. c.weightedSupport $\Leftarrow$ weighted support count 8. if c.w*eightedS*upport > min_ws 9. $F \Leftarrow F \cup c$ 10. $k \Leftarrow k+1$ 11. C_k = generateCandidates(F_{k-1}) 12. *End Loop* 13. $\forall f \in F$ 14. generate set of candidate rules $\{r_1,...,r_n\}$ 15. $R \Leftarrow R \cup \{r_1,...,r_n\}$ 16. $\forall r \in R$ 17. r.weightedConfidence $\Leftarrow$ weighted confidence value 18. if r.weightedConfidence>min_wc $R' \Leftarrow R' \cup r$

7 Experimental Results

We performed several experiments using a T10I4D100K (average of 10 items per transaction, average of 4 items per interesting set, 10K attributes and 100K transactions) synthetic data set. The data set was generated using the IBM Quest data generator. Two sets of experiments were undertaken with four different algorithms namely Boolean WARM (BWARM), Fuzzy WARM (FWARM), Classical Apriori ARM and Classical WARM shown in the results below:

1. In the first experiment we tested algorithms using both boolean and fuzzy datasets and compared the outcome with classical ARM and WARM algorithms. Experiments show (i) the number of frequent sets generated (using four

algorithms), (ii) the number of rules generated (using weighted confidence) and (iii) execution time using all four algorithms.

2. Comparison of execution times using different weighted supports and data sizes.

7.1 Experiment One: Quality Measures

For experiment one, the T10I4D100K dataset described above was used with weighted attributes. Each item is assigned a weight range between $[0..1]$. With fuzzy dataset each attribute is divided into five different fuzzy sets. Figure 3 shows the number of frequent itemsets generated using (i) weighted boolean dataset and (ii) with weighted quantitative attributes with fuzzy partitions (iii) classical ARM with boolean dataset and (iv) and WARM with weighted boolean datasets. A range of support thresholds was used.

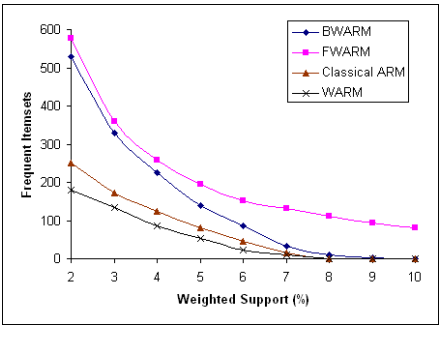


Fig. 2. No. of frequent Itemsets

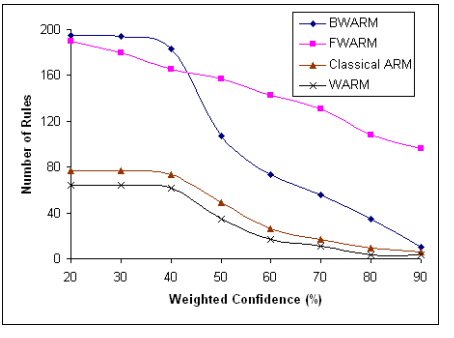


Fig. 3. No. of Interesting Rules

As expected the number of frequent itemsets increases as the minimum support decreases in all cases. In figure 2, BWARM shows the number of frequent itemsets generated using weighted boolean datasets. FWARM shows the number of frequent itemsets using attributes with fuzzy linguistic values, Classical Apriori shows the number of frequent itemset using boolean dataset and classical WARM shows number of frequent itemsets generated using weighted boolean datasets with different weighted support thresholds. More frequent itemsets and rules are generated because of a large itemset pool.

We do not use Apriori ARM to first find frequent itemsets and then re-prune them using weighted support measures. Instead all the potential itemsets are considered from beginning for pruning using Apriori approach in order to validating the DCP. In contrast classical WARM only considers frequent itemsets and prunes them (using pre or post processing). This generates less frequent itemsets and misses potential ones.

Figures 3 shows the number of interesting rules generated using weighted confidence, fuzzy weighted confidence and classical confidence values respectively. In all cases, the number of interesting rules is less as compared to figure 2. This is because the interestingness measure generates fewer rules. Figure 4 shows the execution time of four algorithms.

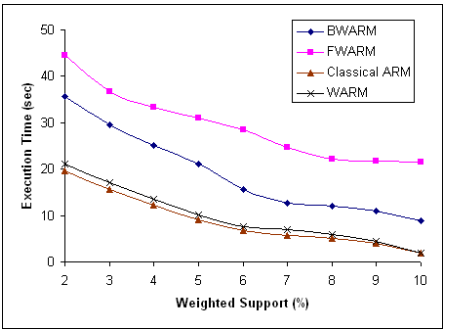


Fig. 4. Execution time to generate frequent itemsets

The experiments show that the proposed framework produces better results as it uses all the possible itemsets and generates rules using the DCP. Further, the novelty is the ability to analyse both boolean and fuzzy datasets with weighted settings.

7.2 Experiment Two: Performance Measures

Experiment two investigated the effect on execution time caused by varying the weighted support and size of data (number of records). A support threshold from 0.1 to 0.6 and confidence 0.5 was used. Figures 5 and 6 show the effect on execution time by increasing the weighted support and number of records. To obtain different data sizes, we partitioned T10I4D100K into 10 equal horizontal partitions labeled 10K, 20K... 100K.

Different weighted support thresholds were used with different datasets. Similarly from figures 5 and 6, the algorithms scales linearly with increasing weighted support and fuzzy weighted support thresholds and number of records, similar behaviour to Classical ARM.

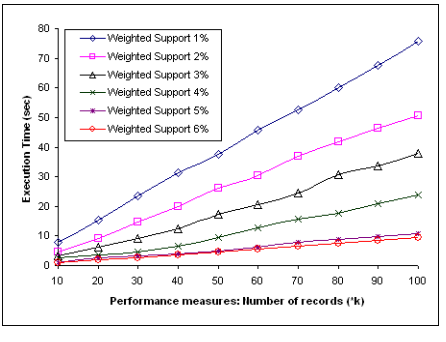


Fig. 5. Performance: weighted support

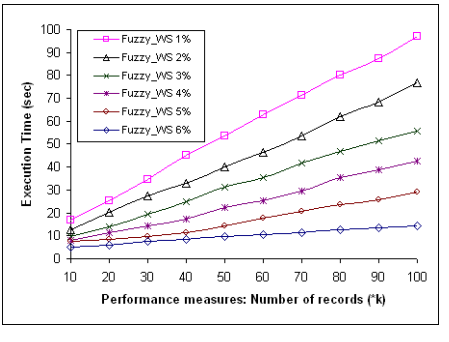


Fig. 6. Performance: fuzzy weighted support

8 Conclusion and Future Work

In this paper, we have presented a weighted support and confidence framework for mining weighted association rules with (Boolean and quantitative data) by validating the downward closure property (DCP). We used classical and fuzzy ARM to solve the issue of invalidation of DCP in weighted ARM. We generalized the DCP and proposed a fuzzy weighted ARM framework. The problem of invalidation of downward closure property is solved using improved model of weighted support and confidence framework for classical and fuzzy association rule mining.

There are still some issues with different measures for validating DCP, normalization of values etc which are worth investigating.

References

1. Tao, F., Murtagh, F., Farid, M.: Weighted Association Rule Mining Using Weighted Support and Significance Framework. In: Proceedings of 9th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington DC, pp. 661–666 (2003)
2. Cai, C.H., Fu, A.W.-C., Cheng, C.H., Kwong, W.W.: Mining Association Rules with Weighted Items. In: Proceedings of 1998 Intl. Database Engineering and Applications Symposium (IDEAS 1998), Cardiff, Wales, UK, pp. 68–77 (July 1998)
3. Wang, W., Yang, J., Yu, P.S.: Efficient Mining of Weighted Association Rules (WAR). In: Proceedings of the KDD, Boston, MA, pp. 270–274 (August 2000)
4. Lu, S., Hu, H., Li, F.: Mining Weighted Association Rules. Intelligent data Analysis Journal 5(3), 211–255 (2001)
5. Wang, B.-Y., Zhang, S.-M.: A Mining Algorithm for Fuzzy Weighted Association Rules. In: IEEE Conference on Machine Learning and Cybernetics, vol. 4, pp. 2495–2499 (2003)
6. Gyenesei, A.: Mining Weighted Association Rules for Fuzzy Quantitative Items. In: Proceedings of PKDD Conference pp. 416–423 (2000)
7. Shu, Y.J., Tsang, E., Yeung, D.S.: Mining Fuzzy Association Rules with Weighted Items. In: IEEE International Conference on Systems, Man, and Cybernetics (2000)
8. Lu, J.-J.: Mining Boolean and General Fuzzy Weighted Association Rules in Databases. Systems Engineering-Theory & Practice 2, 28–32 (2002)
9. Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules. In: 20th VLDB Conference, pp. 487–499 (1994)
10. Bodon, F.: A Fast Apriori implementation. In: ICDM Workshop on Frequent Itemset Mining Implementations, Melbourne, Florida, USA, vol. 90 (2003)
11. Agrawal, R., Imielinski, T., Swami, A.: Mining Association Rules Between Sets of Items in Large Databases. In: 12th ACM SIGMOD on Management of Data, pp. 207–216 (1993)
12. Kuok, C.M., Fu, A., Wong, M.H.: Mining Fuzzy Association Rules in Databases. SIGMOD Record 27(1), 41–46 (1998)
13. Coenen, F., Leng, P., Goulbourne, G.: Tree Structures for Mining Association Rules. Data Mining and Knowledge Discovery 8(1), 25–51 (2004)

A Framework for Mining Fuzzy Association Rules from Composite Items

Maybin Muyeba[1], M. Sulaiman Khan[2], and Frans Coenen[3]

[1] Department of Computing and Mathematics, Manchester Metropolitan University, Manchester, M1 5GD, UK
[2] Liverpool Hope University, Liverpool, L16 9JD, UK
[3] Department of Computer Science, University of Liverpool, Liverpool, L69 3BX, UK
M.Muyeba@mmu.ac.uk, khanm@hope.ac.uk, frans@csc.liv.ac.uk

Abstract. A novel framework is described for mining fuzzy Association Rules (ARs) relating the properties of composite attributes, i.e. attributes or items that each feature a number of values derived from a common schema. To apply fuzzy Association Rule Mining (ARM) we partition the property values into fuzzy property sets. This paper describes: (i) the process of deriving the fuzzy sets (Composite Fuzzy ARM or CFARM) and (ii) a unique property ARM algorithm founded on the correlation factor interestingness measure. The paper includes a complete analysis, demonstrating: (i) the potential of fuzzy property ARs, and (ii) that a more succinct set of property ARs (than that generated using a non-fuzzy method) can be produced using the proposed approach.

Keywords: Association rules, fuzzy association rules, composite attributes, quantitative attributes.

1 Introduction

Association Rule Mining (ARM) now a well known and established data mining topic among researchers. Mainly, ARM finds frequent items (attributes, usually binary valued) and then identifies patterns in the form of Association Rules (ARs) from large transaction data sets [5, 6, 12]. ARM has been applied to quantitative and categorical (non-binary) data [1, 13, 16]. With the former, values can be split into linguistically labeled ranges such that each range represents a binary valued; for example “low”, “medium”, “high” etc. Values can be assigned to these attribute ranges using crisp or fuzzy boundaries. The application of the former is referred to as fuzzy ARM (FARM) [1]. Objectively, fuzzy ARM identifies fuzzy ARs. Fuzzy ARM has been shown to produce more expressive ARs than the “crisp” methods [1, 3, 4,]. ARM (both fuzzy and standard) algorithms typically operate using the support-confidence framework, however with a number of disadvantages including (among others) the tendency to generate many and mostly redundant ARs not any more useful, expressive, succinct or significant. In contrast, the correlation measure produces a more succinct set of rules [3] and we explore this aspect.

S. Chawla et al. (Eds.): PAKDD 2008 Workshops, LNAI 5433, pp. 62–74, 2009.

We approach the problem differently in this paper by introducing "Composite item" Fuzzy ARM (CFARM) whose main objective is the generation of fuzzy ARs associating the "properties" linked with composite attributes [15] i.e. attributes or items composed of sets of sub-attributes or sub-items that conform to a common schema. For example, given an image mining application, we might represent different areas of each image in terms of groups of pixels such that each group is represented by the normalized summation of the RGB values of the pixels in that group. In this case the set of composite attributes (I) is the set of groups, and the set of properties (P) shared by the groups is equivalent to the RGB summation values (i.e. $P = \{R, G, B\}$). Another could be market basket analysis where I is a set of groceries, and P is a set of nutritional properties that these groceries posses i.e. P = {Pr, Fe, Ca, Cu,..} standing for protein, Iron etc[10]. Note that the actual values (properties) associated with each element of I will be constant, unlike in the case of the image mining example. We note that there are many examples depending on application area but we limit ourselves to these given here. For quantitative attributes, we can partition them into intervals [13] and rename these with linguistic values (fuzzy sets) [1].

The contributions in this paper are :

- The framework of the concept of "Composite item" mining of property ARs
- The potential of using property ARs in many applications
- Greater accuracy using the certainty factor measure as against confidence
- Demonstration of a more succinct set of property ARs (than that generated using a non-fuzzy method) can be produced using the proposed approach.

The paper is organised as follows. In section 2 we present the background and related work to the proposed composite fuzzy ARM approach described, Section 3 presents a sequence of formal definitions for the work and section 4, the detail of the CFARM algorithm; a complete analysis of the CFARM algorithm is given in Section 5, and section 6 concludes the paper with a summary of the contribution of the work and directions for future work.

2 Background and Related Work

Most ARM algorithms in general concentrate on performance [2, 3, 5] by first generating all large (frequent) itemsets and then find ARs from them. To limit the number of ARs generated a confidence threshold is used. However great care must be taken not to remove low support items but from which high confidence rules may be generated. In literature the term "composite item" has been used in the context of data mining. In [8, 16], a composite item is defined as a combination of several items e.g. if itemset {A, B} and {A, C} are not large then rules {B}→{A} and {C}→{A} will not be generated, but by combining B and C to make a new *composite* item {BC} which may be large, rules such as {BC}→{A} may be generated. In this paper we define composite items differently as indicated earlier, to be an item with properties (see Section 3) and also in [15], composite attributes are defined in this manner.

In ARM, quantitative attributes are usually discretised into various partitions, with each partition regarded as a binary valued attribute. One major problem in this

approach is that of "sharp boundary problems". Fuzzy ARM [3, 7, 14] has been shown to resolve this problem by mapping numeric values to membership degrees from their partitions with total individual item contributions to support counts remaining as unity value (1.0) regardless of whether an item value belongs to one or more fuzzy sets (similar to the approach in [1]). Detailed overviews of FARM are given in [1, 3, 9, 14].

To illustrate the concepts, we consider super market basket analysis (table 1) where the set of groceries (I) (or edible items) have a common set of nutritional quantitative properties.

Table 1. Example composite attributes (groceries) with their associated properties (nutrients)

Items/Nutrients	Protein	Fibre	Carbohydrate	Fat	…
Milk	3.1	0	4.7	0.2	…
Bread	8	3.3	43.7	1.5	…
Biscuit	6.8	4.8	66.3	22.8	…
…	…	…	…	…	…

To illustrate the context of our problem, composite items (edible items) have common properties like Protein, Fibre, Iron etc and are defined by the same five fuzzy sets {Very Low, Low, Ideal, High, Very High}. The objective is then to identify patterns linking these properties and so derive fuzzy association rules (see next section).

3 Problem Definition

In this section a sequence of formal definitions is presented to define composite attributes, describe FARM concept, the normalization process for Fuzzy Transactions (*FT)* and interestingness measures.

3.1 Terms and Definitions

Definition 1: A *Fuzzy Association Rules* [3] is an implication of the form:

$$\text{if } \langle A, X \rangle \text{ then } \langle B, Y \rangle$$

where A and B are disjoint itemsets and X and Y are fuzzy sets.

Definition 2: *Raw Dataset* (the input data) D consists of a set of transactions $T = \{t_1, t_2, t_3, \cdots, t_n\}$, composite items $I = \{i_1, i_2, i_3, \cdots, i_{|I|}\}$ and properties $P = \{p_1, p_2, p_3, \cdots, p_m\}$. Each transaction t_i is some subset of I, and each item $t_i[i_j]$ (the "j^{th}" item in the "i^{th}" transaction) is a subset of P. Thus i_j has associated sets of values in set P, i.e. $t_i[i_j] = \{v \mid v_1, v_2, v_3, \cdots, v_m\}$.

Table 2. Example raw dataset D

TID	Record
1	{<a,{2,4,6}>, <b,{4,5,3}>}
2	{<c,{1,2,5}>, <d,{4,2,3}>}
3	{<a,{2,4,6}>, <c,{1,2,5}>, <d,{4,2,3}>}
4	{<b,{4,5,3}>, <d,{4,2,3}>}

The "k^{th}" property (categorical or quantitative) value for the "j^{th}" item in the "i^{th}" transaction is given by $t_i[i_j[v_k]]$. An example is given in Table 2 where each composite item is represented using the notation <label, value>. In the rest of this paper the term "item" is used to mean an item in an itemset as used in traditional ARM, and the term attribute is used to mean a property item (sub-item).

Definition 3: A given raw dataset D is initially transformed into a *property data set* D^p with property transactions $T^p = \{t_1^p, t_2^p, t_3^p, \cdots, t_n^p\}$ and property attributes P (instead of a set of composite items I). Thus $\forall t_i^p \subset P$. The value for each property attribute $t_i^p[p_j]$ (the "j^{th}" property attribute in the "i^{th}" property transaction) is obtained by aggregating the numeric values for all p_j in t_i (See Table 3). Thus:

$$\text{Prop Value}(t_i^p[p_j]) = \frac{\sum_{j=1}^{|t_i|} t_i[i_j[v_k]]}{|t_i|} \tag{1}$$

Table 3. Example property data set D^p generated from raw data set given in table 2

TID	X	Y	Z
1	3.0	4.5	4.5
2	3.0	2.0	4.0
3	2.3	2.3	4.7
4	4.0	3.5	3.0

Definition 4. Once a property data set D^p is defined, it is then transformed into a *Fuzzy Dataset* D'. A fuzzy dataset D' consists of fuzzy transactions $T' = \{t_1', t_2', t_3', ..., t_n'\}$ and a set of fuzzy property attributes P' each of which has fuzzy sets with linguistic *labels* $L = \{l_1, l_2, l_3, ..., l_{|L|}\}$. Each property attribute $t_i^p[p_j]$ is associated (to some degree) with several fuzzy sets and given by a *membership degree* value in $[0..1]$ in some *fuzzy linguistic labels*. The "k^{th}" label for

the "j^{th}" property attribute for the "i^{th}" fuzzy transaction is given by $t_i'[p_j[l_k]]$. The nature of the user defined fuzzy ranges is expressed in a *properties table* (see definition 6 below). The numeric values for each property attribute $t_i^p[p_j]$ are *fuzzified* (mapped) into the appropriate membership degree values using a membership function $\mu(t_i^p[p_j], l_k)$ that applies the value of $t_i^p[p_j]$ to a label $l_k \in L$, e.g. $t_i'[p_j] = \{\mu(t_i^p[p_j]], l_1), \mu(t_i^p[p_j]], l_2), \mu(t_i^p[p_j]], l_3), \cdots, \mu(t_i^p[p_j]], l_{|L|})\}$. The complete set of fuzzy property attributes P' is then given by $P \times L$. A fuzzy data (Table 4) based on the property data set (Table 3) is given.

Table 4. Example Fuzzy data set ($L = \{\text{small, medium, large}\}$, μ unspecified)

TID	X			Y			Z		
	Small	Medium	Large	Small	Medium	Large	Small	Medium	Large
1	0.0	1.0	0.0	0.0	0.4	0.6	0.0	1.0	0.0
2	0.0	1.0	0.0	1.0	0.0	0.0	0.3	0.7	0.0
3	0.3	0.7	0.0	1.0	0.0	0.0	0.0	0.9	0.1
4	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0

Definition 5. *Composite Itemset Value (CIV)* table allows us to get property values for specific items. Note that a CIV table is not always required; the values may be included in the raw data as in the case of the example raw dataset presented in Table 2 where property values are all in the range [1..6]. The CIV table for the example raw dataset given in Table 2 is given in Table 5 below.

Table 5. Composite Itemset Value Table for raw dataset given in Table 2

Item	Property attributes		
	X	Y	Z
A	2	4	6
B	4	5	3
C	1	2	5
D	4	2	3

Definition 6. *Properties Table* is a table that maps all possible values for each property attribute $t_i^p[p_j]$ onto user defined (and overlapping) linguistic labels L. An example is given in Table 6 for the raw data set given in Table 2.

Definition 7. A property attribute set A, where $A \subseteq P \times L$, is a *Fuzzy Frequent Attribute Set* if its fuzzy support value is greater than or equal to a user supplied minimum support threshold (see sub-section 3.2 below).

Table 6. Properties Table for raw dataset given in Table 2

Property	Linguistic values		
	Low	Medium	High
X	$v_k \leq 2.3$	$2.0 < v_k \leq 3.7$	$3.3 < v_k$
Y	$v_k \leq 3.3$	$3.0 < v_k \leq 4.3$	$4.1 < v_k$
Z	$v_k \leq 4.0$	$3.6 < v_k \leq 5.1$	$4.7 < v_k$

Definition 8. *Fuzzy Normalisation* is the process of finding the contribution to the fuzzy support value, m', for individual property attributes ($t_i^p[p_j[l_k]]$) such that a partition of unity is guaranteed. This is given by the equation (where μ is the membership function):

$$t_i'[p_j[l_k]] = \frac{\mu(t_i^p[p_j[l_k]])}{\sum_{x=1}^{|L|} \mu(t_i^p[p_j[l_x]])} \tag{2}$$

Without normalisation, the sum of the support contributions of individual fuzzy sets associated with an attribute in a single transaction may no longer be unity. This is illustrated in Tables 7 and 8. In the tables, the possible values for the item "Proteins" have been ranged into five fuzzy sets labelled: "Very Low" (VL), "Low" (L), "Ideal", "High" (H) and "Very High" (VH). Table 7 shows a set of raw membership degree values, while Table 8 shows the normalised equivalents. The normalisation process ensures membership values for each property attribute are consistent and are not affected by boundary values.

Table 7. Fragment data set without normalization

TID	Proteins					…
	VL	L	Ideal	H	VH	…
1	0.0	0.0	0.0	1.0	0.32	…
2	0.83	0.38	0.0	0.0	0.0	…
3	…	…	…	…	…	…

Table 8. Fragment data set with normalization

TID	Proteins					
	VL	L	Ideal	H	VH	…
1	0.0	0.0	0.0	0.76	0.24	…
2	0.69	0.31	0.0	0.0	0.0	…
3	…	…	…	…	…	…

3.2 Fuzzy Support and Confidence

The support-confidence framework can also be applied to fuzzy association rule mining through fuzzy support (significance) values. Fuzzy Support (FS) is typically calculated as follows [1]:

$$FS(A) = \frac{\text{Sum of votes satisfying A}}{\text{Number of records in } T}$$

where $A = \{a_1, a_2, a_3, ..., a_{|A|}\}$ is a set of property attribute-fuzzy set (label) pairs such that $A \subseteq P \times L$. A record t_i' "satisfies" A if $A \subseteq t_i'$. The individual vote per record is found by multiplying the membership degree with an attribute-fuzzy set pair $[i[l]] \in A$:

$$\text{vote for } t_i \text{ satisfying } A = \prod_{\forall [i[l]] \in A} t_i'[i[l]] \tag{3}$$

So we have,

$$\text{FS}(A) = \frac{\sum_{i=1}^{i=n} \prod_{\forall [i[l]] \in A} t_i'[i[l]]}{n} \tag{4}$$

Frequent attribute sets with fuzzy support above the specified threshold are used to generate all possible rules. A fuzzy AR derived from a fuzzy frequent attribute set C is of the form:

$$A \rightarrow B$$

where A and B are disjoint subsets of the set $P \times L$ such that $A \cup B = C$. Fuzzy Confidence *(FC)* is calculated in the same manner that confidence is calculated in classical ARM:

$$FC\ (A \rightarrow B) = \frac{FS\ (A \cup B)}{FS\ (A)} \tag{5}$$

3.3 Fuzzy Correlation

The Fuzzy Confidence measure (FC) described does not use $FS(B)$ but the fuzzy correlation measure (F_{CORR}) addresses this. The correlation measure is a statistical measure founded on the concepts of *covariance* (Cov) and *variance* (Var) and is calculated as follows:

$$F_{CORR}\ (A \rightarrow B) = \frac{Cov\ (A, B)}{\sqrt{Var\ (A) \times Vat\ (B)}} \tag{6}$$

In statistics covariance is calculated by subtracting the product of the individual expected values for A and B from the expected value of C where $C = A \cup B$. The value of correlation ranges from -1 to +1. Value -1 means no correlation and +1 means maximum correlation. Thus we are only interested in rules that have a correlation value that is greater than 0. As the certainty value increases from 0 to 1, the more related the attributes are and consequently the more interesting the rule.

4 The CFARM Algorithm

Fuzzy ARM can use standard ARM algorithms and few works report on their efficient implementations [7]. Fuzzy ARM do a significant amount of processing (filtration, conversions, normalization) to prepare the raw data prior to mining it.

The proposed Composite Fuzzy ARM (CFARM) algorithm (similar to Apriori [5]), belongs to the *breadth first traversal* family of ARM algorithms, developed using tree data structures [6]. The CFARM algorithm consists of four major steps:

1. Transformation of ordinary transactional data set (T) into a property data set (T^P).
2. Transformation of property data set (T^P) into a fuzzy data set T'.
3. Apply an Apriori style fuzzy association rule mining algorithm to T' using fuzzy support, confidence and correlation measures of the form described above to produce a set of frequent item sets F.
4. Process F and generate a set of fuzzy ARs R such that $\forall r \in R$ the certainty factor (either confidence or correlation as desired by the end user) is above some user specified threshold.

Table 9. rawToPropertyDataSetConverter(T)

Input: T = **Raw data set**
Output: T^P = Property data set
1. $\forall t_i \in T$
2. $\forall p_k \in P$
3. $\forall i_j \in t_i$
4. $value \Leftarrow value + t_i[i_j[v_k]]]$
5. $t_i^P[p_j] \Leftarrow value/\lvert t_i \rvert$
6. $T^P \Leftarrow T^P \cup t_i^P$

Table 10. propertToFuzzyDataSetConverter(T^P)

Input: T^P = **property data set**
Output: T' = Fuzzy data set
7. $\forall t_i^P \in T^P$
8. $\forall p_j \in t_i^P$
9. $\forall l_k \in L$
10. $t'[p_j[l_k]] \Leftarrow \mu([t_i^P[p_j]], l_k)$
11. $T' \Leftarrow T' \cup t'[p_j]$

The algorithms for steps 1and 2 are presented in Tables 9 and 10. To illustrate steps 1 and 2 a fragment of a raw data set (T) is given in Table 11(a). This raw data is then cast into a properties data set (T^P) by averaging the property values for each transaction (see definition 3 and table 3). For example, assuming the CIV table given in table 5 and considering transaction $t_1 = \{a, b\}$, from Table 5, a has property values {2, 4, 6} and b has property values {4, 5, 3}. Thus $t_1^P = \{(2+4)/2, (4+5)/2, (6+3)/2\} = \{3.0, 4.5, 4.5\}$, assuming the properties table of the form presented in Table 4 where $L = \{\text{Small}, \text{Medium}, \text{Large}\}$. The result is as shown in Table 11(b) which is then cast into a fuzzy data set T' as shown in Table 11(c).

An alternative approach is to discretise the data. For example, again assuming no overlapping (say) Small < 2, 2 < Medium < 4 and 4 < *Large*, then the values in Table 12(b) can be discretised into the set of attributes $\{X_{\text{Small}}, X_{\text{Medium}}, X_{\text{Large}}, Y_{\text{Small}}, Y_{\text{Medium}}, Y_{\text{Large}}, Z_{\text{Small}}, Z_{\text{Medium}}, Z_{\text{Large}}\}$ and then

assigned to a sequence {1,2,3,4,5,6,7,8,9}. In that case the property data set in Table 11(b) could be represented in conventional ARM terms, which can then be mined using a conventional ARM algorithm. The significance is that we shall use an example property dataset cast into this format for evaluation purposes in Section 5.

The final part of the CFARM algorithm is given in Table 12. In the Table: C_k is the set of candidate itemsets of cardinality k, F is the set of frequent item sets, R is

Table 11. Some example data sets (raw, property, conventional)

(a) Raw data (T)

TID	Items
1	a, b
2	c
3	a, b, d
4	…

(b) Property data set (T^P)

TID	X	Y	Z
1	3.0	4.5	4.5
2	1	2	5
3	3.3	3.3	4.0
4	…	…	…

(c) Fuzzy data set (T')

TID	X			Y			Z		
	S	M	L	S	M	L	S	M	L
1	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.7	0.3
2	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.2	0.8
3	0.0	0.7	0.3	0.4	0.6	0.0	0.0	1.0	0.0
4	…	…	…	…	…	…	…	…	…

Table 12. fuzzyDataSetToFuzzyARs(T')

Input:
T' = Fuzzy data set

Output:
R' = Set of Fuzzy ARs

1. k = 0; $C_k = \varnothing$; $F_k = \varnothing$
2. C_k = Set of 1item sets
3. $k \Leftarrow 1$
4. Loop
5. if $C_k = \varnothing$ break
6. Add fuzzy support values to C_k
7. $\forall c \in C_k$
8. c.support $\Leftarrow$ fuzzy support count
9. if c.support > minSuuport $F \Leftarrow F \cup c$
10. $k \Leftarrow k+1$
11. C_k = generateCandidates(F_{k-1})
12. *End Loop*
13. $\forall f \in F$
14. generate set of candidate rules $(r_1,\ldots,r_n)$
15. $R \Leftarrow R \cup (r_1,\ldots,r_n)$
16. $\forall r \in R$
17. r.certaintyFactor $\Leftarrow$ fuzzy confidence or correlation value
18. if r.certaintyFactor>minCertainty $R' \Leftarrow R' \cup r$

the set of potential rules and R' is the final set of generated fuzzy ARs. Note that the certainty factor can be confidence or a correlation or some other certainty measure.

5 Experimental Results

To demonstrate the effectiveness of the approach, we performed several experiments using a T10I4N0.6KD100k data set generated using IBM Quest data generator [11]. The data is a transactional database containing 100K records. For the purpose of the experiment we mapped the 600 item numbers onto 600 products in a real RDA table.

5.1 Experiment One: Quality Measures

Our experiment in the first instance compares CFARM, with and without normalisation, against standard (discrete) ARM with respect to: (i) the number of frequent sets generated and (ii) the number of rules generated (using both the confidence and the correlation measure). Figure 1 shows the results and demonstrates the difference between the number of frequent itemsets generated using (i) Standard ARM using discrete intervals, (ii) CFARM with fuzzy partitions without normalization (CFARM1), and (iii) Fuzzy ARM with fuzzy partitions with normalization (CFARM2).

For standard ARM, the Apriori-TFP algorithm was used [6] with a range of support thresholds. As expected the number of frequent itemsets increases as the minimum support decreases. From the results, it is clear that standard ARM produces more frequent itemsets (and consequently rules) than fuzzy ARM (figure 1).

This is because the frequent itemsets generated more accurately reflect the true patterns in the data set than the numerous artificial patterns resulting from the use of crisp boundaries in standard ARM. At low support threshold levels, the approach with normalization (CFARM2) starts to produce less frequent itemsets than the approach without normalization (CFARM1). This is because the average contribution to support counts per transaction is greater without using normalization than with normalization.

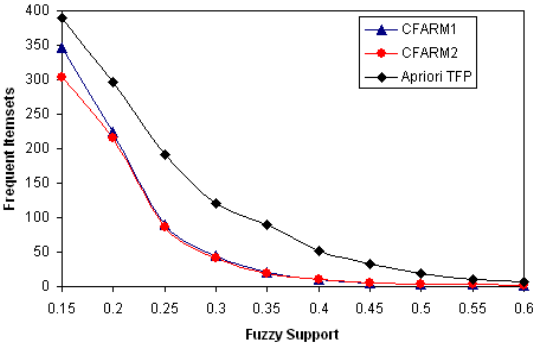


Fig. 1. Number of frequent Itemsets

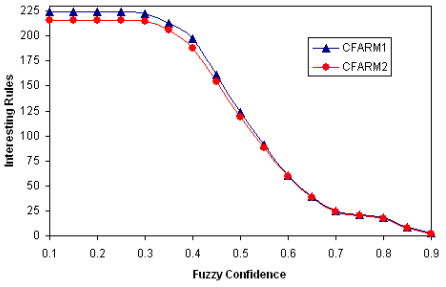


Fig. 2. No. of Interesting Rules using confidence

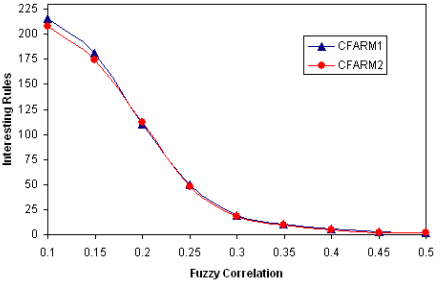


Fig. 3. No. of Interesting Rules using Correlation

Figures 2 and 3 shows the comparison of number of interesting rules generated using user specified fuzzy confidence and fuzzy correlation values respectively. In both cases, the number of interesting rules is less as using CFARM2; this is a direct consequence of the fact that CFARM 2 generates fewer frequent itemsets. Note that fewer, but arguably better, rules are generated using the correlation measure (Figure 3) than the confidence measure (Figure 2). The experiments show that using the proposed fuzzy normalization process less fuzzy ARs are generated. In addition, the novelty of the approach is its ability to analyse datasets comprised of composite items e.g. nutritional properties. Some example fuzzy ARs generated has the form:

IF *Protein* intake is *Low* THEN *Vitamin A* intake is *High.*
IF *Protein* intake is *High* AND *Vitamin A* intake is *Low* THEN *Fat* intake is *High.*

These rules would be useful in analysing customer buying patterns concerning their nutrition.

5.2 Experiment Two: Performance Measures

Experiment two investigated the effect on execution time by varying the number of attributes and the size of data (number of records) with and without normalization using a support threshold of 0.3, confidence 0.5 and correlation value to 0.25. Figure 4

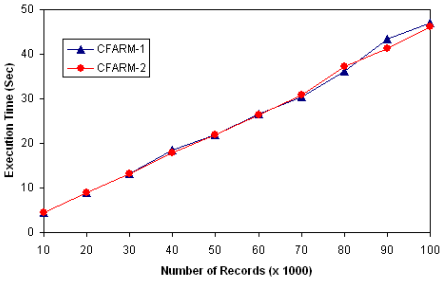


Fig. 4. Execution time: No. of Records

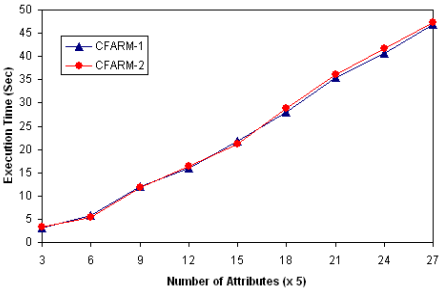


Fig. 5. Execution time: No. of Attributes

shows the effect of increasing the number of records partitioned into 10 equal partitions 10K, 20K,..,100K with all 27 nutrients (properties) used [10]. Both algorithms have similar timings with execution time scales linearly with the number of records. Figure 5 shows the effect on execution time using different numbers of attributes each with 5 fuzzy sets and thus uses 135 columns (27x5).

6 Conclusion and Future Work

In this paper, we have presented a novel framework for extracting fuzzy ARs from "composite items" with quantitative properties (sub itemsets) using derived fuzzy sets. The CFARM algorithm produces a more succinct set of fuzzy association rules using fuzzy measures and correlation as the interestingness (certainty) measure and thus presents a new way for extracting association rules from items with properties. This is different from normal quantitative ARM. We also showed the experimental results with market basket data where edible items were used with nutritional content as properties. Largely, CFARM offers potential to apply this framework in varied applications with composite items.

References

1. Gyenesei, A.: A Fuzzy Approach for Mining Quantitative Association Rules. Acta Cybernetical 15(2), 305–320 (2001)
2. Lee, C.H., Chen, M.S., Lin, C.R.: Progressive Partition Miner, an Efficient Algorithm for Mining General Temporal Association Rules. IEEE Trans. on Knowledge and Data Engineering 15(4), 1004–1017 (2003)
3. Kuok, C., Fu, A., Wong, H.: Mining Fuzzy Association Rules in Databases. ACM SIGMOD Record 27(1), 41–46 (1998)
4. Dubois, D., Hüllermeier, E., Prade, H.: A Systematic Approach to the Assessment of Fuzzy Association Rules. DM and Knowledge Discovery Journal 13(2), 167–192 (2006)
5. Bodon, F.: A Fast Apriori implementation. In: Proc. (FIMI 2003), IEEE ICDM Workshop on Frequent Itemset Mining Implementations, Florida, USA, vol. 90 (2003)
6. Coenen, F., Leng, P., Goulbourne, G.: Tree Structures for Mining Association Rules. Data Mining and Knowledge Discovery 8(1), 25–51 (2004)
7. Chen, G., Wei, Q.: Fuzzy Association Rules and the Extended Mining Algorithms. Information Sciences 147(1-4), 201–228 (2002)
8. Wang, K., Liu, J.K., Ma, W.: Mining the Most Reliable Association Rules with Composite Items. In: Proc. ICDMW 2006, pp. 749–754 (2006)
9. Delgado, M., Marin, N., Sanchez, D., Vila, M.A.: Fuzzy Association Rules, General Model and Applications. IEEE Transactions on Fuzzy Systems 11(2), 214–225 (2003)
10. Muyeba, M., Sulaiman, M., Malik, Z., Tjortjis, C.: Towards Healthy Association Rule Mining (HARM), A Fuzzy Quantitative Approach. In: Corchado, E., Yin, H., Botti, V., Fyfe, C. (eds.) IDEAL 2006. LNCS, vol. 4224, pp. 1014–1022. Springer, Heidelberg (2006)
11. Agrawal, R., Srikant, R.: Quest Synthetic Data Generator. IBM Almaden Research Center
12. Agrawal, R., Imielinski, T., Swami, A.: Mining Association Rules Between Sets of Items in Large Databases. In: Proc. ACM SIGMOD Int. Conf. on Management of Data, Washington, D.C, pp. 207–216 (1993)

13. Srikant, R., Agrawal, R.: Mining Quantitative Association Rules in Large Relational Tables. In: Proc. ACM SIGMOD Conf. on Management of Data, pp. 1–12. ACM Press, Montreal (1996)
14. Au, W.H., Chan, K.: Farm, A Data Mining System for Discovering Fuzzy Association Rules. In: Proc. 8th IEEE Int'l Conf. on Fuzzy Systems, Seoul, Korea, pp. 1217–1222 (1999)
15. Kim, W., Bertino, E., Garza, J.: Composite objects revisited. ACM SIGMOD Record 18(2), 337–347 (1989)
16. Ye, X., Keane, J.A.: Mining Composite Items in Association Rules. In: Proc. IEEE Int. Conf. on Systems, Man and Cybernetics, pp. 1367–1372 (1997)

Mining Mutually Dependent Ordered Subtrees in Tree Databases

Tomonobu Ozaki[1] and Takenao Ohkawa[2]

[1] Organization of Advanced Science and Technology, Kobe University
[2] Graduate School of Engineering, Kobe University
1-1 Rokkodai-cho, Nada, Kobe, 657-8501, Japan
tozaki@cs.kobe-u.ac.jp, ohkawa@kobe-u.ac.jp
http://www25.cs.kobe-u.ac.jp/

Abstract. In this paper, in order to discover significant patterns, we focus on the problem of mining *frequent mutually dependent ordered subtrees*, *i.e.* frequent ordered subtrees in which all building blocks are mutually dependent, in tree databases. While three kinds of mutually dependent ordered subtrees are considered based on the building blocks used, we propose efficient breadth-first algorithms for each kind of subtrees. The effectiveness of the proposed framework is assessed through the experiments with synthetic and real world datasets.

Keywords: tree mining, mutually dependent patterns, h-confidence.

1 Introduction

Recently, frequent pattern mining in tree-structured domain has been paid a big attention and several algorithms have been proposed [2, 3, 4, 6, 13, 14]. However, frequent subtree miners often discover unmanageable number of patterns. To overcome this problem, several algorithms for mining condensed representation [5, 8, 10] as well as constrained patterns [9, 15] have been proposed.

On the other hand, in order to discover significant patterns and to decrease the number of patterns to be extracted, the concept of *hyperclique pattern* has been proposed [11, 12] in the research area of frequent itemset mining. A hyperclique pattern is a set of highly-correlated items that has high value of an objective measure *h-confidence* which is designed for capturing the strong affinity relationship. The h-confidence measure of an itemset $P = \{i_1, \cdots, i_m\}$ is defined as follows [11, 12].

$$hconf(P) = \min_{l=1,\cdots,m} \{conf(i_l \rightarrow P \setminus \{i_l\})\} = sup(P) / \max_{l=1,\cdots m} \{sup(\{i_l\})\}$$

where *sup* and *conf* are the conventional definitions of support and confidence in association rules [1], respectively.

In this paper, as a tree-structured version of hyperclique pattern, we consider the problem of discovering *mutually dependent ordered subtrees*, *i.e.* subtrees in which all components or building blocks are highly correlated with each other.

S. Chawla et al. (Eds.): PAKDD 2008 Workshops, LNAI 5433, pp. 75–86, 2009.

While it is obvious that the building blocks of an itemset are items contained in the set, the meaningful building blocks of a subtree are not necessarily obvious. Thus, we propose the three kinds of mutually dependent ordered subtrees based on the building blocks used. To discover each kind of subtrees efficiently, we propose three algorithms which are based on an existing breadth-first frequent ordered subtree miner named AMIOT [6].

This paper is organized as follows. In section 2, we introduce basic notations and definitions on ordered subtree mining and give a brief overview of AMIOT as the basis of the proposed algorithms. In section 3, we propose three kinds of mutually dependent ordered subtrees. Our mining algorithms for each kind of mutually dependent ordered subtrees are also proposed in this section. We show the results of the experiments in section 4. Finally, we conclude this paper and describe future work in section 5.

2 Preliminaries

2.1 Notations and Definitions

In this subsection, we introduce basic notations and definitions on the ordered subtree mining according to [2, 4, 6].

A *labeled ordered tree* $t = (V, E, S, L, r)$ on a finite set of labels $\mathcal{L}$ consists of a *vertex* set V, an *edge* set $E \subseteq V \times V$ which represents the set of parent-child relations, $S \subseteq V \times V$ which represents the set of *sibling relations*, a labeling function $L : V \to \mathcal{L}$ which gives a label $l \in \mathcal{L}$ of a vertex, and a unique vertex r that has no entering edges. The label of a vertex $v \in V$ is denoted as $\mathrm{l}(v)$. The depth of v, denoted as $\mathrm{d}(v)$, is defined as the number of edges from the root to v. Given two labeled ordered trees $t = (V_t, E_t, S_t, L_t, r_t)$ and $s = (V_s, E_s, S_s, L_s, r_s)$, t is called *induced ordered subtree* of s, denoted as $t \prec s$, if there exists an injective function $\phi : V_t \to V_s$ which satisfies the following conditions: $(1)(v_1, v_2) \in E_t \Leftrightarrow (\phi(v_1), \phi(v_2)) \in E_s$, $(2)(v_1, v_2) \in S_t \Leftrightarrow (\phi(v_1), \phi(v_2)) \in S_s$, and $(3)L_t(v) = L_s(\phi(v))$. We show examples of ordered trees in Fig. 1. In this figure, for a vertex Ⓕ in t_0, $\mathrm{l}(Ⓕ) = F$ and $\mathrm{d}(Ⓕ) = 2$, respectively. $t_6 \prec t_5 \prec t_3 \prec t_0$ holds. Hereafter, we refer *labeled ordered tree* as *tree* and *induced ordered subtree* as *subtree* simply.

A vertex having no child is called a *leaf*. A tree having just one leaf is called a *serial tree*. Given a tree t, $\mathrm{rl}(t)$ and $\mathrm{ll}(t)$ denote the rightmost and leftmost leaf in t, respectively. A set of all paths from the root to leaves in t is denoted as $\mathrm{B}(t)$. $\mathrm{rb}(t) = (r, \cdots, \mathrm{rl}(t)) \in \mathrm{B}(t)$ and $\mathrm{lb}(t) = (r, \cdots, \mathrm{ll}(t)) \in \mathrm{B}(t)$ denote

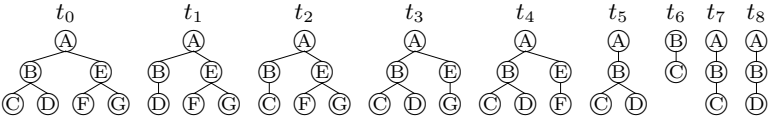


Fig. 1. Example of Ordered Trees

the rightmost and leftmost branch in t, respectively. We denote a set of non-rightmost and non-leftmost branches in t as $\mathrm{Cb}(t) = \mathrm{B}(t) \setminus \{\mathrm{rb}(t), \mathrm{lb}(t)\}$. In Fig. 1, t_6, t_7 and t_8 are serial trees. While $\mathrm{rl}(t_0)$ is Ⓖ, $\mathrm{ll}(t_0)$ is Ⓒ. On the paths from a root to leaves, $\mathrm{B}(t_0) = \{(Ⓐ, Ⓑ, Ⓒ), (Ⓐ, Ⓑ, Ⓓ), (Ⓐ, Ⓔ, Ⓕ), (Ⓐ, Ⓔ, Ⓖ)\}$ holds. $\mathrm{rb}(t_0)$ is $(Ⓐ, Ⓔ, Ⓖ)$ and $\mathrm{lb}(t_0)$ is $(Ⓐ, Ⓑ, Ⓒ)$, respectively. While $\mathrm{Cb}(t_0) = \{(Ⓐ, Ⓑ, Ⓓ), (Ⓐ, Ⓔ, Ⓕ)\}$ holds, $\mathrm{Cb}(t_5) = \emptyset$ because t_5 has only two leaves.

The size of a tree t is denoted as $|t|$ and is defined as the number of vertices in t. $\mathrm{P}(t) = \{t' \prec t \mid |t'| = |t| - 1\}$ denotes a set of all immediate subtrees of t. While $\mathrm{p_l}(t) \in \mathrm{P}(t)$ denotes the tree obtained by removing $\mathrm{rl}(t)$ from t, $\mathrm{p_r}(t) \in \mathrm{P}(t)$ denotes the tree obtained by removing $\mathrm{ll}(t)$. For a serial tree t, a tree obtained by removing the root from t is denoted as $\mathrm{p_d}(t)$.

Given a tree t, while $t \cdot (d, l)$ denotes a tree obtained by adding a vertex v such that $\mathrm{d}(v) = d$ and $\mathrm{l}(v) = l$ as the rightmost leaf to t, $(d, l) \bullet t$ denotes a tree obtained by adding v as the leftmost leaf to t. Given a vertex v, we also use the notation $t \cdot v$ and $v \bullet t$ to represent $t \cdot (\mathrm{d}(v), \mathrm{l}(v))$ and $(\mathrm{d}(v), \mathrm{l}(v)) \bullet t$ for the sake of the simplicity. In Fig. 1, $\mathrm{P}(t_0) = \{t_1, t_2, t_3, t_4\}$, $\mathrm{p_l}(t_0) = t_4$ and $\mathrm{p_r}(t_0) = t_1$, respectively. For a serial tree t_7, $\mathrm{p_d}(t_7) = t_6$ holds. While $t_0 = (2, C) \bullet t_1 = t_4 \cdot (2, G)$ holds, $t_5 = (2, C) \bullet t_8 = t_7 \cdot (2, D)$ holds.

Let $D = \{t_1, \cdots, t_N\}$ be a database of labeled ordered trees. The *support* of a tree t in D is defined as follows.

$$sup(D, t) = \sum_{s \in D} \mathrm{Occ}(t, s) \qquad where \qquad \mathrm{Occ}(t, s) = \begin{cases} 1\ (t \prec s) \\ 0\ (otherwise) \end{cases}$$

Given a user defined threshold $\sigma > 0$, a tree t is called a *frequent induced ordered subtree* in D, if $sup(D, t) \geq \sigma$ holds.

2.2 Breadth-First Frequent Induced Ordered Subtree Miner

In this subsection, as the basis of the algorithm for solving the problems of mutually dependent subtree mining introduced later, we introduce a frequent induced ordered subtree miner named AMIOT [6].

AMIOT [6] is a natural extension of apriori algorithm [1] and it discovers all frequent induced ordered subtrees in a breadth-first search strategy. In order to enumerate all frequent subtrees without duplication, AMIOT employs two enumeration techniques. For each two frequent subtrees t and s such that $\mathrm{p_r}(t) = \mathrm{p_l}(s)$, a new candidate $t \cdot \mathrm{rl}(s) = \mathrm{ll}(t) \bullet s$ is obtained. This operation is called *Right and Left Tree Join*. In addition, a set of candidates $C = \{t \cdot (\mathrm{d}(\mathrm{rl}(t)) + 1, l) \mid l \in \mathcal{L}\}$ is generated from a frequent serial tree t. This operation is called *Serial Tree Extension*. For example in Fig. 1, while t_0 will be generated by joining t_4 and t_1, t_5 will be obtained from t_7 and t_8.

AMIOT generates all frequent subtrees without duplication by repeatedly applying these two kinds of operations to frequent subtrees by size [6]. The pseudo code of AMIOT is shown in Fig. 2. In this figure, *c.sup* denotes a support value of a tree c and $sup(D, c)$ means the actual computation of support value. A set $\mathcal{F}_1 = \{t \mid sup(D, t) \geq \sigma, |t| = 1\}$ denotes a set of all frequent subtrees of

Algorithm AMIOT ($\mathcal{F}_1$, D, σ)
1: **for each** $t \in \mathcal{F}_1$
2: **output** t
3: **end for**
4: AMIOT-Enum($\mathcal{F}_1$, D, σ)

Function AMIOT-Enum(F, D, σ)
1: $F' := \emptyset$
2: **for each** $t, s \in F$ s.t. $\mathrm{p_r}(t) = \mathrm{p_l}(s)$
3: $c := t \cdot \mathrm{rl}(s)$; $c.sup := sup(D, c)$
4: **if** $c.sup \geq \sigma$ **then** $F' = F' \cup \{c\}$; **output** c
5: **end for**
6: **for each** $l \in \mathcal{L}$ and $t \in F$ s.t. t is a serial tree
7: $c := t \cdot (\mathrm{d}(\mathrm{rl}(t)) + 1, l)$; $c.sup := sup(D, c)$
8: **if** $c.sup \geq \sigma$ **then** $F' = F' \cup \{c\}$; **output** c
9: **end for**
10: **if** $F' \neq \emptyset$ **then** AMIOT-Enum(F', D, σ)

Fig. 2. Pseudo Code of AMIOT

size 1. In AMIOT-Enum, line 2–5 and line 6–9 correspond to Right and Left Tree Join and Serial Tree Extension, respectively. By utilizing the anti-monotone property of support value, *i.e.* $\forall t' \succ t$ ($sup(D, t') \leq sup(D, t)$), AMIOT avoids enumerating candidate subtrees having no chance to be frequent. As a result, AMIOT succeeds in reducing the number of candidates to be evaluated.

3 Discovery of Mutually Dependent Ordered Subtrees

3.1 Mutually Dependent Ordered Subtrees

To understand the meaning of a structured pattern, we often pay attention to the relation between whole pattern and its building blocks. In case of ordered subtrees, the relation between a subtree t and its building block $c \prec t$ can be represented in the tree association rule $c \rightarrow t$. In this rule, the strength of relation is measured as confidence $conf(c \rightarrow t) = sup(D, t)/sup(D, c)$. We believe that ordered subtrees which have strong relations in arbitrary building blocks are meaningful, significant and easy to understand because such patterns do not contain redundant parts. Motivated by these backgrounds, as patterns concerning the relation between arbitrary building blocks and the whole, we propose mutually dependent ordered subtrees, which can be regarded as an extension of hyperclique pattern to ordered subtrees.

Given a database D of ordered subtrees and a user defined threshold h_c, then a subtree t such that $\mathrm{hconf}_D(t) \geq h_c$ is called a *mutually dependent ordered subtree* (MDOT in short).

$$\begin{aligned}\mathrm{hconf}_D(t) = \min_{c \prec t}\{conf(c \rightarrow t)\} &= sup(D, t)/\max(\{sup(D, c) \mid c \prec t\}) \\ &= sup(D, t)/\max(\{sup(D, c) \mid c \prec t,\ |c| = 1\})\end{aligned}$$

In case of tree structured patterns, it is not necessarily obvious what meaningful building block is. Therefore, it may be not suitable to use all subtrees for measuring the affinity relationship within a pattern. To overcome this drawback, instead of using all subtrees, we consider that only subtrees satisfying some property contribute for evaluating the affinity relationship. According to the property

that subtrees should satisfy, we propose three kinds of MDOT named (1)sMDOT, (2)iMDOT and (3)pMDOT. In the computation of h-confidence in sMDOT, only subtrees whose size is more than or equal to k are considered. While immediate subtrees are used in iMDOT, all paths from the root to leaves are considered in pMDOT.

In the following subsections, we give the formal definitions of each kind of MDOT and those mining problems.

3.2 Discovery of sMDOT

sMDOT is a subtree in which every subtrees whose size is more than or equal to k are mutually dependent, where k is a user defined parameter. This means that too small subtrees as building block are not considered. Given a database D and a user defined parameter $k(\geq 1)$, we define the modified version of h-confidence for sMDOT as follows.

$$\begin{aligned} \mathrm{hconf}^s_D(t,k) &= sup(D,t)/\max(\{sup(D,c) \mid c \prec t,\ |c| \geq k\}) \\ &= sup(D,t)/\max(\{sup(D,c) \mid c \prec t,\ |c| = k\}) \end{aligned}$$

By using this measure, our data mining problem is stated formally as follow.

(σ, h_c, k)-sMDOT Discovery Problem. Given a database D of labeled ordered trees, one positive number $h_c \geq 0$ and two integers $\sigma \geq 1$ and $k(\geq 1)$, then the problem of (σ, h_c, k)-sMDOT discovery is to find all subtrees t in D such that $|t| > k$, $sup(D,t) \geq \sigma$ and $\mathrm{hconf}^s_D(t,k) \geq h_c$.

Note that, the (σ, h_c, 1)-sMDOT discovery problem is identical with the problem of finding all frequent MDOTs, in which the original h-confidence is employed as a measure for affinity.

The measure hconf^s_D satisfies the anti-monotone property. Formally speaking, given an ordered subtree t, the relation $\forall t' \succ t\,(\mathrm{hconf}^s_D(t',k) \leq \mathrm{hconf}^s_D(t,k))$ holds because $\{c' \mid c' \prec t'\} \supseteq \{c \mid c \prec t\}$ holds. By incorporating the pruning capability based on the anti-monotone property of hconf^s_D into AMIOT, we propose an algorithm named MINE-sMDOT for the (σ, h_c, k)-sMDOT discovery problem. The pseudo code of MINE-sMDOT is shown in Fig. 3. Instead of Serial Tree Extension in AMIOT, every serial trees will be generated by joining two serial trees (line 7–8 in sMDOT-Enum). Any subtree c keeps the value $c.max$ for computing hconf^s_D efficiently. In addition to the support-based pruning (line 2 in sMDOT-Chk), another pruning based on h-confidence is realized (line 5 in sMDOT-Chk). The correctness of MINE-sMDOT is derived from the complete enumeration of frequent subtrees based on AMIOT and the admissible pruning guaranteed by the anti-monotone property of hconf^s_D.

3.3 Discovery of iMDOT

Intuitively speaking, iMDOT is a subtree, all of whose immediate subtrees are iMDOT and are tightly correlated with each other. The h-confidence is modified

Algorithm MINE-sMDOT($\mathcal{F}_1$, D, σ, k, h_c)
1: **for each** $t \in \mathcal{F}_1$
2: $t.max := t.sup$
3: **end for**
4: sMDOT-Enum($\mathcal{F}_1$, D, σ, k, h_c)

Function sMDOT-Chk(t, s, d, D, σ, k, h_c)
1: $c := t \cdot (d, \mathrm{l}(\mathrm{rl}(s)))$; $c.sup := sup(D, c)$
2: **if** $c.sup < \sigma$ **then return** $\emptyset$
3: **if** $|c| > k$ **then**
4: $c.max := \max(\ \{p.max \mid p \in \mathrm{P}(c)\}\ )$
5: **if** $c.sup/c.max < h_c$ **then return** $\emptyset$
6: **output** c
7: **else**
8: $c.max := c.sup$
9: **return** $\{c\}$

Function sMDOT-Enum(F, D, σ, k, h_c)
1: $F' := \emptyset$
2: **for each** $t, s \in F$ s.t. $\mathrm{p_r}(t) = \mathrm{p_l}(s)$
3: $F' := F' \cup$
4: sMDOT-Chk
5: (t, s, $\mathrm{d}(s)$, D, σ, k, h_c)
6: **end for**
7: **for each** $t, s \in F$ s.t. t and s are
8: serial trees and $\mathrm{p_d}(t) = \mathrm{p_r}(s)$
9: $F' := F' \cup$
10: sMDOT-Chk
11: (t, s, $\mathrm{d}(s)$+1, D, σ, k, h_c)
12: **end for**
13: **if** $F' \neq \emptyset$ **then**
14: sMDOT-Enum(F', D, σ, k, h_c)

Fig. 3. Pseudo Code of sMDOT

for considering only the immediate subtrees as follows.

$$\mathrm{hconf}^i_D(t) = \begin{cases} sup(D,t)/\max(\{sup(D,c) \mid c \in \mathrm{P}(t)\}) & (|t| > 1) \\ 1 & (|t| = 1) \end{cases}$$

By using the measure hconf^i_D, the formal definition of iMDOT is given as follows: (1)a subtree c of size 1 is iMDOT and (2)a subtree c such that $|c| > 1$ is iMDOT if $\mathrm{hconf}^i_D(c) \geq h_c$ and all of $c' \in \mathrm{P}(c)$ are iMDOT, where h_c is a user defined parameter. Note that, the definition of iMDOT is recursive.

Based on the above preparation, we propose the second data mining problem.

(σ, h_c)-iMDOT Discovery Problem. Given a database D of labeled ordered trees, an integer $\sigma \geq 1$, and a positive number $h_c \geq 0$, then the problem of (σ, h_c)-iMDOT discovery is to find all subtrees t in D such that $|t| > 1$, $sup(D, t) \geq \sigma$, and t is iMDOT.

iMDOT satisfies the anti-monotone property by definition. In other words, if a subtree t is not iMDOT, then any subtree $t' \succ t$ must not be iMDOT. Thus, as similar to the case of sMDOT, an effective mining algorithm can be composed by introducing the pruning capability based on hconf^i_D into AMIOT . The algorithm MINE-iMDOT for the (σ, h_c)-iMDOT discovery problem is shown in Fig. 4. The pruning based on hconf^i_D is realized in line 3–4 in iMDOT-Chk. The completeness of this algorithm is derived from the complete enumeration of frequent patterns in AMIOT and the safe pruning based on the anti-monotone property of support value and hconf^i_D.

3.4 Discovery of pMDOT

The third kind of mutually dependent ordered subtrees is referred to as pMDOT. We regard that pMDOT is composed of a set of paths from the root to the

Algorithm MINE-iMDOT($\mathcal{F}_1$, D, σ, h_c)
1: iMDOT-Enum($\mathcal{F}_1$, D, σ, h_c)

Function iMDOT-Chk(t, s, d, D, σ, h_c)
1: $c := t \cdot (d, \mathrm{l}(\mathrm{rl}(s)))$; $c.sup := sup(D, c)$
2: **if** $c.sup < \sigma$ **then return** $\emptyset$
3: $max := \max(\ \{p.sup \mid p \in \mathrm{P}(c)\}\ )$
4: **if** $sup/max < h_c$ **then return** $\emptyset$
5: **output** c
6: **return** $\{c\}$

Function iMDOT-Enum(F, D, σ, h_c)
1: $F' := \emptyset$
2: **for each** $t, s \in F$ s.t. $\mathrm{p_r}(t) = \mathrm{p_l}(s)$
3: $F' := F' \cup$ iMDOT-Chk
4: (t, s, $\mathrm{d}(s)$, D, σ, h_c)
5: **end for**
6: **for each** $t, s \in F$ s.t. t and s are
7: serial trees and $\mathrm{p_d}(t) = \mathrm{p_r}(s)$
8: $F' := F' \cup$iMDOT-Chk
9: (t, s, $\mathrm{d}(s)$+1, D, σ, h_c)
10: **end for**
11: **if** $F' \neq \emptyset$ **then**
12: iMDOT-Enum(F', D, σ, h_c)

Fig. 4. Pseudo Code of iMDOT

leaves. The h-confidence measure for considering paths only is defined formally as follows.

$$\mathrm{hconf}^p_D(t) = sup(D, t) / \max(\{sup(D, c) \mid c \in \mathrm{B}(t)\})$$

In the following, we state the third data mining problem.

(σ, h_c)-pMDOT Discovery Problem. Given a database D of labeled ordered trees, an integer $\sigma \geq 1$, and a positive number $h_c \geq 0$, then the problem of (σ, h_c)-pMDOT discovery is to find all non-serial subtrees t in D such that $sup(D, t) \geq \sigma$ and $\mathrm{hconf}^p_D(t, k) \geq h_c$.

Unlike the case of sMDOT and iMDOT, the measure hconf^p_D does not satisfy the anti-monotone property since $\mathrm{B}(t) \subseteq \mathrm{B}(t')$ does not hold for two subtrees $t \prec t'$. Therefore, other pruning criterion is necessary for the efficient discovery of pMDOT. As described before, any non-serial subtree must be generated by Right and Left Tree Join in AMIOT . In this operation, a new tree will be obtained by adding one vertex to the rightmost or leftmost branch of the existing tree. No vertex will be added to the branches other than rightmost and leftmost ones. Thus, given a non-serial subtree t and t' which is obtained via t in AMIOT , then $\mathrm{Cb}(t) \subseteq \mathrm{Cb}(t')$ holds. By utilizing this relation, we propose three pruning techniques.

1. Center Branch Pruning: In the enumeration strategy of AMIOT , if a non-serial tree t satisfies the following condition, then t will be pruned.

$$sup(D, t) / \max(\{sup(D, c) \mid c \in \mathrm{Cb}(t)\}) < h_c$$

The admissibility of this pruning is proved as follows. Let t' be a subtree obtained via t in AMIOT . From the previous discussion, $\mathrm{Cb}(t') \supseteq \mathrm{Cb}(t)$ as well as $\max(\{sup(D, c) \mid c \in \mathrm{Cb}(t')\}) \geq \max(\{sup(D, c) \mid c \in \mathrm{Cb}(t)\})$ hold. In addition,

Algorithm MINE-pMDOT($\mathcal{F}_1$, D, σ, h_c)
1: pMDOT-Enum($\mathcal{F}_1$, D, σ, h_c)

Function pMDOT-Enum(F, D, σ, h_c)
1: $F' := \emptyset$
2: **for each** $t, s \in F$ s.t. $\mathrm{p_r}(t) = \mathrm{p_l}(s)$
3: **if** $t.sup / \max(\ \{p.sup \mid p \in \mathrm{Cb}(t) \cup \{\mathrm{rb}(t)\}\}\ ) < h_c$
4: $\wedge\ \mathrm{d}(\mathrm{rl}(t)) \geq \mathrm{d}(\mathrm{rl}(s))$ **then continue**
5: **if** $s.sup / \max(\ \{p.sup \mid p \in \mathrm{Cb}(s) \cup \{\mathrm{lb}(s)\}\}\ ) < h_c$
6: $\wedge\ \mathrm{d}(\mathrm{ll}(s)) \geq \mathrm{d}(\mathrm{ll}(t))$ **then continue**
7: $c := t \cdot \mathrm{rl}(s)$; $c.sup := sup(D, c)$
8: **if** $c.sup < \sigma$ **then continue**
9: **if** $\mathrm{Cb}(c) \neq \emptyset \wedge c.sup / \max(\ \{p.sup \mid p \in \mathrm{Cb}(c)\}\ ) < h_c$ **then continue**
10: **if** $c.sup / \max(\ \{p.sup \mid p \in \mathrm{B}(c)\}\ ) \geq h_c$ **then output** c
11: $F' := F' \cup \{c\}$
12: **end for**
13: **for each** $t, s \in F$ s.t. t and s are serial trees and $\mathrm{p_d}(t) = \mathrm{p_r}(s)$
14: $c := t \cdot (\mathrm{d}(\mathrm{rl}(s)) + 1, \mathrm{l}(\mathrm{rl}(s)))$; $c.sup := sup(D, c)$
15: **if** $c.sup \geq \sigma$ **then** $F' := F' \cup \{c\}$
16: **end for**
17: **if** $F' \neq \emptyset$ **then** pMDOT-Enum(F', D, σ, h_c)

Fig. 5. Pseudo Code of pMDOT

because of $t' \succ t$, $sup(D, t') \leq sup(D, t)$ also holds. By these inequalities, the following inequality is obtained.

$$\frac{sup(D, t')}{\max(\{sup(D, c) \mid c \in \mathrm{Cb}(t')\})} \leq \frac{sup(D, t)}{\max(\{sup(D, c) \mid c \in \mathrm{Cb}(t)\})} \leq h_c$$

2. Rightmost Branch Pruning: In the enumeration strategy of AMIOT , if a non-serial tree t satisfies the following condition, then we prune any subtree $t \cdot (d, l)$ such that $d < \mathrm{d}(\mathrm{rl}(t))$.

$$sup(D, t) / \max(\{sup(D, c) \mid c \in \mathrm{Cb}(t) \cup \{\mathrm{rb}(t)\}\}) < h_c$$

Because $\mathrm{Cb}(t \cdot (d, l)) = \mathrm{Cb}(t) \cup \{\mathrm{rb}(t)\}$ holds, the center branch pruning will be applied to $t \cdot (d, l)$ as a result. This shows the admissibility of this pruning.

3. Leftmost Branch Pruning: In the enumeration strategy of AMIOT , if a non-serial tree s satisfies the following condition, then we prune any subtree $(d, l) \bullet s$ such that $d < \mathrm{d}(\mathrm{ll}(s))$.

$$sup(D, s) / \max(\{sup(D, c) \mid c \in \mathrm{Cb}(s) \cup \{\mathrm{lb}(s)\}\}) < h_c$$

The admissibility of this pruning is guaranteed by the same reason of rightmost branch pruning.

On the ground of the above discussion, we propose an algorithm named MINE-pMDOT for the (σ, h_c)-pMDOT discovery problem in Fig. 5. In Fig. 5, center branch pruning is realized in line 9 in pMDOT-Enum. The rightmost and leftmost branch prunings are realized in line 3–4 and line 5–6, respectively. In this algorithm, while non-pMDOT will be output by the check of hconf_D^p(line 10), all pMDOT will be generated because of the complete enumeration strategy of AMIOT and safe prunings.

4 Experimental Evaluation

To assess the effectiveness of the proposed framework, we implement three algorithms MINE-sMDOT, MINE-iMDOT and MINE-pMDOT in Java and conduct experiments with the datasets shown in Table 1. Datasets $10K$ and $50K$ are synthetically generated by *Tree Generator* [13]. Glycan [7] and CSLOGS [13] are real world datasets. While the former is a set of biochemical compounds, the latter contains the access trees to the website.

In the experiments, we measure the number of MDOTs obtained and the execution time with the decrease of minimum support and minimum h-confidence gradually. All experiments were run on a PC (CPU: Intel(R) Core2Quad 2.4GHz) with 4GB of main memory running Windows XP. The experimental results are shown in Table 2 and Table 3. In these tables, while "closed" means the number of frequent closed induced ordered subtrees [5], "d-free" means frequent $\delta = d$-free induced ordered subtrees [10] which are noise-tolerant minimal frequent subtrees. The rows of "tree ar" mean the numbers of tree association rules satisfying the minimum confidence h_c in form of $c \rightarrow t$ where c and t are restricted to closed patterns and $c \prec t$.

In all the experiments, MDOTs are extracted within a very small computational time. These results show the effectiveness of the proposed criteria for the pruning based on the h-confidence. Although the minimal size for sMDOT increases as the value of k in sMDOT discovery is enlarged, the number of extracted sMDOTs increases. The reason might be that the number of subtrees for computing hconf_D^s decreases. Regardless of the kinds of MDOT, compared with closed patterns, δ-free patterns and tree association rules, the number of obtained MDOT is very small in almost all cases. From these results, we believe that only significant subtrees have been selectively discovered in the proposed frameworks.

Table 1. Overview of Datasets

	$\lvert D\rvert$	Size	Depth	$\lvert\mathcal{L}\rvert$		$\lvert D\rvert$	Size	Depth	$\lvert\mathcal{L}\rvert$
$10K$	10,000	15.00 / 143	5.99 / 10	100	Glycan	10,951	6.52 / 54	3.71 / 25	875
$50K$	50,000	10.66 / 143	4.83 / 10	100	CSLOGS	59,691	12.93 / 428	3.43 / 85	13,355

$\lvert D\rvert$: Number of trees in the database, Size: Average / Maximal number of nodes per tree, Depth: Average / Maximal height per tree, $\lvert\mathcal{L}\rvert$: Number of labels

Table 2. Experimental Results for Synthetic Datasets: Number of Extracted Patterns

		$10K$			$50K$		
	h_c	$\sigma = 50$	$\sigma = 25$	$\sigma = 10$	$\sigma = 250$	$\sigma = 100$	$\sigma = 50$
sMDOT(k=1)		7 (0.2)	7 (0.2)	7 (0.2)	7 (0.7)	7 (0.8)	7 (0.7)
sMDOT(k=3)		59 (1.1)	69 (1.1)	82 (1.1)	93 (3.5)	110 (4.2)	115 (4.3)
sMDOT(k=5)	0.9	127 (1.4)	152 (1.5)	202 (1.6)	197 (6.5)	247 (6.9)	268 (6.9)
iMDOT		7 (0.2)	7 (0.2)	7 (0.2)	7 (0.0)	7 (0.8)	7 (0.7)
pMDOT		0 (2.6)	0 (3.0)	0 (3.2)	0 (8.6)	0 (14.4)	0 (15.5)
sMDOT(k=1)		15 (0.3)	15 (0.2)	15 (0.2)	16 (0.7)	16 (0.8)	16 (0.8)
sMDOT(k=3)		96 (1.1)	118 (1.1)	161 (1.2)	124 (3.6)	147 (5.0)	169 (5.1)
sMDOT(k=5)	0.8	229 (1.5)	284 (1.4)	432 (1.5)	267 (6.6)	340 (6.8)	408 (6.9)
iMDOT		15 (0.3)	15 (0.3)	15 (0.3)	16 (0.0)	16 (0.8)	16 (0.8)
pMDOT		0 (2.4)	0 (3.1)	7 (3.3)	0 (10.9)	0 (10.7)	0 (11.4)
sMDOT(k=1)		16 (0.3)	16 (0.3)	16 (0.3)	20 (0.7)	20 (0.8)	20 (0.8)
sMDOT(k=3)		130 (1.0)	164 (1.1)	260 (1.1)	137 (4.3)	172 (4.2)	217 (4.3)
sMDOT(k=5)	0.7	312 (1.4)	410 (1.4)	713 (1.7)	297 (6.7)	415 (6.3)	541 (7.8)
iMDOT		16 (0.3)	16 (0.3)	16 (0.3)	20 (1.3)	20 (1.3)	20 (0.8)
pMDOT		0 (2.7)	0 (3.0)	56 (3.2)	18 (8.6)	18 (10.5)	18 (14.6)
closed		8,925	21,605	65,933	2,974	9,108	20,320
0-free		9,148	22,387	70,665	3,046	9,386	21,254
3-free		5,868	12,677	30,985	1,872	5,008	10,511
5-free		5,005	10,417	22,547	1,535	4,061	8,321
tree ar	0.9	9,468	19,527	43,091	5,336	14,619	27,147
tree ar	0.8	15,787	35,184	89,311	7,080	20,748	40,906
tree ar	0.7	21,716	51,781	147,129	9,105	27,126	54,422

* Numbers in parentheses are execution time in second.

Table 3. Experimental Results for Real Datasets: Number of Extracted Patterns

		Glycan			CSLOGS		
	h_c	$\sigma = 30$	$\sigma = 20$	$\sigma = 10$	$\sigma = 150$	$\sigma = 125$	$\sigma = 100$
sMDOT(k=1)		0 (0.2)	0 (0.2)	1 (0.2)	1 (1.3)	1 (0.9)	1 (1.0)
sMDOT(k=3)		1 (0.4)	1 (0.4)	3 (0.5)	108 (4.0)	139 (4.7)	192 (5.8)
sMDOT(k=5)	0.9	2 (0.5)	3 (0.6)	5 (0.7)	385 (4.3)	453 (5.3)	828 (6.0)
iMDOT		0 (0.2)	0 (0.2)	1 (0.2)	1 (0.8)	1 (0.9)	1 (1.4)
pMDOT		3 (0.5)	7 (0.6)	13 (0.7)	148 (4.3)	340 (4.8)	1,646 (6.7)
sMDOT(k=1)		2 (0.2)	3 (0.2)	4 (0.3)	5 (0.9)	5 (1.3)	6 (1.4)
sMDOT(k=3)		6 (0.4)	7 (0.4)	10 (0.5)	229 (4.2)	300 (5.0)	491 (5.8)
sMDOT(k=5)	0.8	7 (0.5)	14 (0.5)	25 (0.6)	774 (4.3)	1,261 (5.1)	4,894 (6.0)
iMDOT		2 (0.2)	3 (0.2)	4 (0.3)	6 (1.0)	6 (1.3)	7 (1.5)
pMDOT		4 (0.5)	15 (0.6)	23 (0.7)	266 (4.3)	931 (5.1)	7,925 (10.1)
sMDOT(k=1)		2 (0.2)	3 (0.3)	4 (0.3)	19 (0.9)	19 (1.4)	22 (1.4)
sMDOT(k=3)		18 (0.4)	20 (0.4)	32 (0.5)	521 (4.1)	672 (5.0)	1,302 (5.9)
sMDOT(k=5)	0.7	24 (0.5)	42 (0.6)	71 (0.7)	1,532 (4.3)	4,764 (5.1)	15,221 (6.8)
iMDOT		3 (0.3)	4 (0.2)	5 (0.2)	24 (1.3)	24 (1.4)	27 (1.0)
pMDOT		21 (0.5)	35 (0.5)	74 (0.7)	524 (4.2)	2,355 (5.4)	76,167 (64.8)
closed		2,073	2,709	4,532	1,717	2,527	4,282
0-free		2,091	2,769	4,753	1,724	2,540	4,439
3-free		1,478	1,904	2,966	1,661	2,412	3,884
5-free		1,307	1,677	2,517	1,602	2,295	3,534
tree ar	0.9	2,650	2,910	3,480	267	637	2,918
tree ar	0.8	5,283	5,874	7,255	608	1,441	6,661
tree ar	0.7	7,599	8,610	11,016	1,031	2,375	10,407

* Numbers in parentheses are execution time in second.

To assess the difference of the kinds of MDOT, we show the numbers of subtrees which are extracted as each kinds of MDOT only in Table 4. It is understood that different subtrees have been discovered in sMDOT and pMDOT. These results

Table 4. Number of Uniquely Extracted Patterns

		Glycan			CSLOGS		
	h_c	$\sigma = 30$	$\sigma = 20$	$\sigma = 10$	$\sigma = 150$	$\sigma = 120$	$\sigma = 100$
sMDOT(k=3)	0.9	0 (0.00)	0 (0.00)	2 (0.67)	95 (0.88)	122 (0.88)	170 (0.89)
sMDOT(k=5)		2 (1.00)	3 (1.00)	4 (0.80)	325 (0.84)	388 (0.86)	715 (0.86)
iMDOT		0 (–)	0 (–)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)
pMDOT		2 (0.67)	6 (0.86)	11 (0.85)	84 (0.57)	269 (0.79)	1,525 (0.93)
sMDOT(k=3)	0.8	5 (0.83)	6 (0.86)	9 (0.90)	154 (0.67)	205 (0.68)	308 (0.63)
sMDOT(k=5)		7 (1.00)	14 (1.00)	24 (0.96)	602 (0.78)	970 (0.77)	3,699 (0.76)
iMDOT		0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)
pMDOT		3 (0.75)	14 (0.93)	21 (0.91)	83 (0.31)	621 (0.67)	6,740 (0.85)
sMDOT(k=3)	0.7	16 (0.89)	18 (0.90)	29 (0.91)	213 (0.41)	280 (0.42)	535 (0.41)
sMDOT(k=5)		20 (0.83)	37 (0.88)	62 (0.87)	1,025 (0.67)	3,310 (0.69)	10,344 (0.68)
iMDOT		0 (0.00)	0 (0.00)	0 (0.00)	1 (0.04)	1 (0.04)	1 (0.04)
pMDOT		14 (0.67)	27 (0.77)	61 (0.82)	71 (0.14)	969 (0.41)	71,498 (0.94)

* Numbers in parentheses are ratios of patterns extracted as unique pattern.

mean that the patterns which are extracted will greatly depend on the building blocks employed. In addition, these results show that it may be possible to extract more reasonable patterns by considering appropriate size and shape of building blocks which are suitable for the application domains.

5 Conclusion

In this paper, we formulate the problems of mining mutually dependent ordered subtrees. To solve these problems efficiently, we developed the breadth-first mining algorithms based on an existing frequent subtree miner. Through the experiments, compared to the condensed representation mining, we show that the proposed algorithms can discover small number of patterns within reasonable computational time.

For future work, (1)a theoretical analysis of the proposed framework, (2)detailed examinations of the quality of obtained subtrees and (3)further experiments with large scale real world datasets are necessary. We also plan to apply the proposed framework to mining more complex structured data such as unordered tree and graphs.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proc. of 20th International Conference on Very Large Data Bases (VLDB 1994), pp. 487–499 (1994)
2. Asai, T., Abe, K., Kawasoe, S., Arimura, H., Sakamoto, H., Arikawa, S.: Efficient substructure discovery from large semi-structured data. In: Proc. of the 2nd SIAM International Conference on Data Mining (2002)
3. Asai, T., Arimura, H., Uno, T., Nakano, S.: Discovering frequent substructures in large unordered trees. In: Proc. of the 6th International Conference on Discovery Science, pp. 47–61 (2003)

4. Chi, Y., Nijssen, S., Muntz, R.R., Kok, J.N.: Frequent subtree mining – an overview. Fundamenta Informaticae 66(1-2), 161–198 (2005)
5. Chi, Y., Xia, Y., Yang, Y., Muntz, R.R.: Mining closed and maximal frequent subtrees from databases of labeled rooted trees. IEEE Transactions on Knowledge and Data Engineering 17(2), 190–202 (2005)
6. Hido, S., Kawano, H.: Amiot: Induced ordered tree mining in tree-structured databases. In: Proc. of the 5th IEEE International Conference on Data Mining (ICDM 2005), pp. 170–177 (2005)
7. Kanehisa, M., Goto, S., Kawashima, S., Okuno, Y., Hattori, M.: The KEGG resource for deciphering the genome. Nucleic Acids Research 32, D277–D280 (2004)
8. Ozaki, T., Ohkawa, T.: Efficient mining of closed induced ordered subtrees in tree-structured databases. In: Proc. of the 6th IEEE International Conference on Data Mining - Workshops, pp. 279–283 (2006)
9. Ozaki, T., Ohkawa, T.: Efficiently mining closed constrained frequent ordered subtrees by using border information. In: Zhou, Z.-H., Li, H., Yang, Q. (eds.) PAKDD 2007. LNCS, vol. 4426, pp. 745–752. Springer, Heidelberg (2007)
10. Ozaki, T., Ohkawa, T.: Mining frequent δ-free induced ordered subtrees in tree-structured databases. In: Proc. of the 5th Workshop on Learning with Logics and Logics for Learning (LLLL 2007), pp. 3–9 (2007)
11. Xiong, H., Tan, P.-N., Kumar, V.: Mining strong affinity association patterns in data sets with skewed support distribution. In: Proc. of the 3rd IEEE International Conference on Data Mining (ICDM 2003), pp. 387–394 (2003)
12. Xiong, H., Tan, P.-N., Kumar, V.: Hyperclique pattern discovery. Data Mining and Knowledge Discovery 13(2), 219–242 (2006)
13. Zaki, M.J.: Efficiently mining frequent trees in a forest. In: Proc. of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 71–80 (2002)
14. Zaki, M.J.: Efficiently mining frequent embedded unordered trees. Fundamenta Informaticae, special issue on Advances in Mining Graphs, Trees and Sequences (2005)
15. Zou, L., Lu, Y., Zhang, H., Hu, R.: Mining frequent induced subtree patterns with subtree-constraint. In: Proc. of the 6th IEEE International Conference on Data Mining - Workshops, pp. 3–7 (2006)

A Tree Distance Function Based on Multi-sets

Arnoldo José Müller-Molina, Kouichi Hirata, and Takeshi Shinohara

Department of Artificial Intelligence, Kyushu Institute of Technology
Kawazu 680-4 Iizuka 820-8502, Japan
arnoldo@daisy.ai.kyutech.ac.jp,
{hirata,shino}@ai.kyutech.ac.jp

Abstract. We introduce a tree distance function based on multi-sets. We show that this function is a metric on tree spaces, and we design an algorithm to compute the distance between trees of size at most n in $O(n^2)$ time and $O(n)$ space. Contrary to other tree distance functions that require expensive memory allocations to maintain dynamic programming tables of forests, our function can be implemented over simple and static structures. Additionally, we present a case study in which we compare our function with other two distance functions.

Keywords: Tree edit distance, Program matching, Triangle inequality, Metric.

1 Introduction

Analysis of tree structured data is required in many fields [6, 9, 18]. Our research group is particularly interested in the field of approximate binary program matching for the detection of Open Source/Libre software license violations [11, 12]. A recent technique in this field works by generating fragments, that are regarded as trees, from control flow graphs. By means of a tree distance function, the similarity of two programs can be computed. We are interested in distance functions that satisfy the triangle inequality. If this property is satisfied, then we can employ dimension reduction techniques like SMAP [14], and therefore matching speed on massive databases improves considerably. Additionally, it is desirable that the distance function is a metric. Specifically, it is necessary for the distance function to satisfy the property: $d(x, y) = 0$ if and only if $x = y$. If this property is satisfied, the matching of programs is "safer" and false positives are reduced.

An overview of current literature [2] presents a tree edit distance function and several variants. To compute every distance function that is metric in [2], it is necessary at least $O(n^2)$ time where n is the maximum number of nodes in two given trees. In this paper, we propose a new tree distance function based on multi-sets that is simple to compute, and that does not require dynamic programming or intensive memory allocations. The main contribution of this paper is to introduce a metric for rooted ordered labeled trees that can be computed in $O(n^2)$ time and $O(n)$ space. We denote this metric as *mtd*. The rest of the

S. Chawla et al. (Eds.): PAKDD 2008 Workshops, LNAI 5433, pp. 87–98, 2009.

paper is organized as follows. Section 2 introduces related works. In Section 3, the proposed distance *mtd* is described and in Section 4, a naive algorithm to compute *mtd* is presented. Finally in Section 5, we present a case study.

2 Related Works

The generally accepted similarity measure for trees is the tree-edit distance metric (*ted*) introduced by Tai[15] in 1979. Klein[10] proposed an $O(n^3 \log(n))$ algorithm to compute *ted*. This problem is known to be NP-hard for unordered trees[19]. Currently, the fastest algorithm for ordered trees is $O(n^3)$[5].

Chawathe *et al.*[3] introduced an approach in which the set of edit operations is extended. This distance is NP-complete, and the fastest heuristic runs in $O(n^3)$ time. A different variation [4] is linear, however it is not clear if these similarity functions are metrics nor if they satisfy the triangle inequality.

The tree alignment distance introduced by Jiang *et al.*[7] is not a metric. Finally, the fastest metric that we are aware of is the constrained edit distance[17], with time $O(n^2)$ and space $O(n \log(n))$. The distance functions described above can be seen as an edit script minimization problem. On the other hand, functions based on q-grams [1, 13, 16] are a vector feature distance minimization. These functions have linear complexity, but they are not metrics.

2.1 Approximate Program Matching

The function presented in this paper, is intended to be used in the context of approximate binary program matching field. Müller-Molina and Shinohara introduced a technique based on extracting *program fragments*[11] from binary programs. A program fragment is a tree whose nodes are machine-level instruction codes. They are extracted from program control flow graphs. By analyzing the frequency distribution of the fragments, it is possible to employ "standard" information retrieval techniques to rank programs by similarity. The technique also uses tree distance functions to find similar pairs of program fragments. In this way, even if fragment normalization rules fail, the distance function can compensate for differences introduced by a program transformation. This process is analogous to the stemming process web information retrieval engines apply to each word of a natural language query. The distance introduced in this paper has been employed successfully in this context, but its properties were not studied formally.

3 Tree Similarity Distance

A *tree* is a connected graph without cycles. For a tree $T = (V, E)$, we sometimes denote $v \in T$ instead of $v \in V$, and $|T|$ instead of $|V|$. A *rooted tree* is a tree with one node r chosen as its *root*.

Let r be a root of T and v a node in T. Then, the *parent* of $v(\neq r)$ is its adjacent node in a path from v to r in T, and the *ancestors* of $v(\neq r)$ are the

nodes contained in a path from the parent of v to r in T. The parent and the ancestors of the root r are undefined. We say that u is a *child* of v if v is the parent of u, and u is a *descendant* of v if v is an ancestor of u. A *leaf* is a node having no children. Furthermore, a *complete subtree* of T at v is a subtree of T of which its root is v and that contains all descendants of v.

A rooted tree $T = (V, E)$ is *labeled* (by an alphabet Σ of labels) if there exists an onto function $l : V \to \Sigma$ such that $l(v) = a$ $(v \in V, a \in \Sigma)$. A tree is *ordered* if a left-to-right order for the children of each node is given; *unordered* otherwise. In this paper, we call a rooted labeled ordered tree simply by a tree.

Next, we introduce the notions of *multi-sets*. Intuitively, a *multi-set* is a set that allows an element to occur more than once. The *multiplicity* $m_A(x)$ of an element x in a multi-set A is the number of the occurrences of x in A. For a (standard) set A, it holds that $m_A(x) = 1$ for every $x \in A$. It is clear that $x \notin A$ if $m_A(x) = 0$. The *set view* $v(A)$ of a multi-set A is a set $\{x \in A \mid m_A(x) \geq 1\}$. For example, for a multi-set $A = \{a, a, a, b, c, c\}$, it holds that $v(A) = \{a, b, c\}$. Additionally, the *cardinality* $|A|$ of a multi-set A is defined as $\sum_{x \in v(A)} m_A(x)$.

We now introduce the multi-set operations. Let A and B be multi-sets. Then, the *intersection* $A \sqcap B$, the *union* $A \sqcup B$ and the *additive union* $A \uplus B$ of A and B are defined as follows.

$$A \sqcap B = \{x \in v(A) \cap v(B) \mid m_{A \sqcap B}(x) = \min\{m_A(x), m_B(x)\}\},$$
$$A \sqcup B = \{x \in v(A) \cup v(B) \mid m_{A \sqcap B}(x) = \max\{m_A(x), m_B(x)\}\},$$
$$A \uplus B = \{x \in v(A) \cup v(B) \mid m_{A \uplus B}(x) = m_A(x) + m_B(x)\}.$$

For example, let $A = \{a, a, b, c, c\}$ and $B = \{a, b, b, c\}$. Then, it holds that $A \sqcap B = \{a, b, c\}$, $A \sqcup B = \{a, a, b, b, c, c\}$ and $A \uplus B = \{a, a, a, b, b, b, c, c, c\}$.

Based on the previous operations, we define a similarity measure for multi-sets:

$$\delta(A, B) = |A \sqcup B| - |A \sqcap B|. \tag{1}$$

Finally, we introduce the function $s(T)$ which is the multi-set of all complete subtrees of T. For example, in Figure 1 the tree $C(B)$ is not a complete subtree of T_1, but $C(B, E)$ is a complete subtree of T_1. The function $n(T)$, is the multi-set of all the nodes of T.

3.1 Distance Definition

The first step to calculate our similarity function is to convert a tree into two multi-sets, one multi-set of complete subtrees and another multi-set that contains all the nodes (without children) of the tree. This can be achieved by invoking functions s and n. Once the multi-sets of the trees have been generated, the function *mtd* can be computed. The function *mtd* is defined as:

$$d_s(T_1, T_2) = \delta(s(T_1), s(T_2)), \tag{2}$$

$$d_n(T_1, T_2) = \delta(n(T_1), n(T_2)), \tag{3}$$

$$mtd(T_1, T_2) = \frac{d_s(T_1, T_2) + d_n(T_1, T_2)}{2}. \tag{4}$$

The *mtd* function is composed of two different distance operations: d_s and d_n. The first function d_s is a measure based on the number of equal complete subtrees between T_1 and T_2. Note that a change of only one leaf node can have multiple repercussions in all its parents. In general, this measure is very sensitive to changes and quickly the intersection $s(T_1) \sqcap s(T_2)$ becomes void. Nevertheless, an important reason to have this intersection is that it makes sure that $mtd(T_1, T_2) = 0$ iff $T_1 = T_2$. Additionally, trees that share many common complete subtrees will receive a high score.

The second function d_n is a measure that is likely to find matches since only individual nodes are considered. Note that d_n is Kailing's distance based on the label histogram [8]. Using only this measure would make trees very similar to each other, however it is necessary to balance the strictness of d_s.

To summarize, the intuitive idea of our matching procedure is that a very strict matching (d_s) combined with a permissive matching (d_n) brings a balance to the scoring technique. In the context of binary program matching, we have found that both *mtd* and d_s return acceptable results. Function *mtd* produces slightly better results than d_s.

Note that *mtd* always returns a natural number. Precisely, we have the following proposition:

Proposition 1. *$d_s(T_1, T_2) + d_n(T_1, T_2)$ is even for any trees T_1 and T_2.*

Proof. For multi-sets A and B, since $|A \sqcup B| + |A \sqcap B| = |A \uplus B|$, it holds that $|A \sqcup B| - |A \sqcap B| = |A \uplus B| - 2|A \sqcap B|$. Then, for multi-sets $s(T_i)$, and $n(T_i)$ $(i = 1, 2)$, the following equation holds:

$$|s(T_1) \sqcup s(T_2)| - |s(T_1) \sqcap s(T_2)| + |n(T_1) \sqcup n(T_2)| - |n(T_1) \sqcap n(T_2)| \quad (5)$$

$$= |s(T_1) \uplus s(T_2)| - 2|s(T_1) \sqcap s(T_2)| + |n(T_1) \uplus n(T_2)| - 2|n(T_1) \sqcap n(T_2)| \quad (6)$$

Since $|s(T_i)| = |n(T_i)|$, let $|n(T_i)| = |s(T_i)| = k_i$. Hence, we can obtain the following even expression:

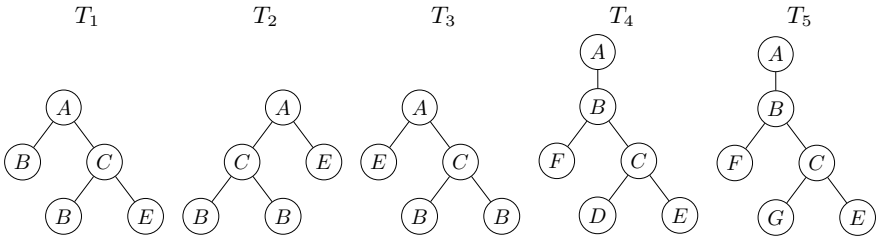
$$2k_1 + 2k_2 - 2|s(T_1) \sqcap s(T_2)| - 2|n(T_1) \sqcap n(T_2)|. \quad (7)$$

□

In what follows, we give some examples of *mtd*.

In Figure 1 three examples are displayed. In the first example, $mtd(T_1, T_2)$, returns 2 because it does not consider in which place a complete subtree is found (multi-sets do not record the position of the subtrees). The function $ted(T_1, T_3)$ returned 2 because "C" is deleted, and a new "C" node is inserted over the two "B" nodes.

In the second example, $mtd(T_1, T_3)$ is calculating the distance of two node renaming operations. The result is again the same for $ted(T_1, T_3)$. In the third example, $mtd(T_4, T_5)$ returns 5 because all of the complete subtrees of T_4 except "E" and "F" are not in T_5. The deeper the modification is, the greater the distance will be. In the context of binary program matching[12] this behavior is desirable because changes in the deepest part of an expression are likely to



$s(T_1) = \{A(B, C(B, E)), B, C(B, E), B, E\}$
$s(T_2) = \{A(C(B, B), E), C(B, B), B, B, E\}$
$s(T_3) = \{A(E, C(B, B)), E, C(B, B), B, B\}$
$s(T_4) = \{A(B(F, C(D, E))), B(F, C(D, E)), F, C(D, E), D, E\}$
$s(T_5) = \{A(B(F, C(G, E))), B(F, C(G, E)), F, C(G, E), G, E\}$

$n(T_1) = \{A, B, C, B, E\}$
$n(T_2) = \{A, C, B, B, E\}$
$n(T_3) = \{A, E, C, B, B\}$
$n(T_4) = \{A, B, F, C, D, E\}$
$n(T_5) = \{A, B, F, C, G, E\}$

$|s(T_1) \sqcap s(T_2)| = 3$ $|s(T_1) \sqcap s(T_3)| = 3$ $|s(T_4) \sqcap s(T_5)| = 2$
$|n(T_1) \sqcap n(T_2)| = 5$ $|n(T_1) \sqcap n(T_3)| = 5$ $|n(T_4) \sqcap n(T_5)| = 5$

$mtd(T_1, T_2) = \frac{(7-3)+(5-5)}{2} = 2$ $ted(T_1, T_2) = 2$
$mtd(T_1, T_3) = \frac{(7-3)+(5-5)}{2} = 2$ $ted(T_1, T_3) = 2$
$mtd(T_4, T_5) = \frac{(10-2)+(7-5)}{2} = 5$ $ted(T_4, T_5) = 1$

Fig. 1. This example calculates $mtd(T_1, T_2)$, $mtd(T_1, T_3)$, and $mtd(T_4, T_5)$. T_1 and T_2 illustrate the cost of complete subtree movement. T_1 and T_3, illustrate two label modification operations. T_3 and T_4 depict the cost of one node modified at deep levels of the tree.

change more drastically the "meaning" of the expression than changes in upper layers of the expression.

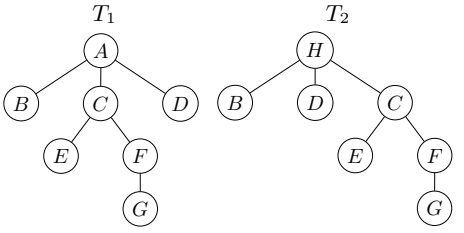
In Figure 2, an example in which *mtd* returns a smaller value than *ted* is shown. In this case, $mtd(T_1, T_2)$ returns a smaller distance than $ted(T_1, T_2)$ because swapped nodes (rooted in "C" and "D") were children of a node whose label was modified. Because the algorithm does not differentiate when the root node *and* the children are modified, the score is lower. The function $ted(T_1, T_2)$ returns 3 because "A" is renamed by "H", "D" is deleted, and finally, "D" is inserted to the left of "C".

Proposition 2. *For any trees T_1, T_2 and T_3, the following statements hold:*

1. $mtd(T_1, T_2) \geq 0$ *non-negativity*
2. $mtd(T_1, T_2) = mtd(T_1, T_2)$ *symmetry*
3. $mtd(T_1, T_2) = 0 \iff T_1 = T_2$ *identity of indiscernibles*
4. $mtd(T_1, T_3) \leq mtd(T_1, T_2) + mtd(T_2, T_3)$ *triangle inequality*

Proof. Properties 1 and 2 are obvious. Property 3 can be deduced from the fact that T_1 and T_2 are returned by the d_s function. If there is any change between the trees, the original tree will not be matched in the intersection operation and a distance greater than zero will be computed.

We now prove property 4. It is sufficient to prove that δ satisfies the triangle inequality, $\delta(A, C) \leq \delta(A, B) + \delta(B, C)$.



$s(T_1) = \{A(B, C(E, F(G)), D), B, C(E, F(G)), E, F(G), G, D\}$
$s(T_2) = \{H(B, D, C(E, F(G))), B, D, C(E, F(G)), E, F(G), G\}$

$n(T_1) = \{A, B, C, E, F, G, D\}$
$n(T_2) = \{H, B, D, C, E, F, G\}$

$|s(T_1) \sqcap s(T_2)| = 6$
$mtd(T_1, T_2) = \frac{(8-6)+(8-6)}{2} = 2$

$|n(T_1) \sqcap n(T_2)| = 6$
$ted(T_1, T_2) = 3$

Fig. 2. In this example, *mtd* returns a smaller distance than *ted*

The function $\delta(A, B)$ can be rewritten as:

$$\delta(A, B) = \sum_{x \in v(A) \cup v(B)} |m_A(x) - m_B(x)|. \tag{8}$$

Since:

$$|m_A(x) - m_C(x)| \leq |m_A(x) - m_B(x)| + |m_B(x) - m_C(x)|, \tag{9}$$

the triangle inequality holds. Therefore *mtd* is a metric on tree spaces. □

4 Naive Algorithm for Computing *mtd*

In what follows, we describe a naive algorithm for computing δ and *mtd*. Once δ is obtained, *mtd* can be computed easily. In lines 3 to 7 of Algorithm 1, the tree v is compared against all the trees of size $|v|$ of the multi-set A. This optimization is necessary to keep the complexity quadratic as we shall see later. The function δ (lines 12 to 14) calculates the intersection between the multi-sets A and B by obtaining the minimum of the multiplicity of the common elements. Finally in line 15, the cardinality of the union of $|T_1|$ and $|T_2|$ is subtracted from the cardinality of the intersection of T_1 and T_2 and the result is returned to the caller.

We now proceed to analyze the complexity of algorithm *mtd*. The following lemma is useful:

Lemma 1. *Every tree of size m has at most $\frac{m}{n}$ complete subtrees of size n.*

Theorem 1. *The distance function $mtd(F, G)$ can be computed in $O(|T_1| \times |T_2|)$ time and $O(|T_1| + |T_2|)$ space.*

Algorithm 1. δ and m functions implementation

```
    Receives a multi-set an a tree
 1: function m(A, v)                                   ▷ Returns m_A(v)
 2:     c ← 0                                          ▷ Multiplicity counter
 3:     for m ∈ A such that |v| = |m| do                ▷ Only compare trees of size |v|.
 4:         if m = v then
 5:             c ← c + 1
 6:         end if
 7:     end for
 8:     return c
 9: end function
    Receives two multi-sets
    v(x) receives a multi-set x and returns a set "view" of x.
10: function δ(A, B)                                   ▷ Returns δ(A, B)
11:     c ← 0                                          ▷ Intersection counter
12:     for v ∈ v(A) do
13:         c ← c + min(m_A(v), m_B(v))
14:     end for
15:     return |A ⊔ B| − c
16: end function
```

Proof. It is necessary to consider the cost involved in comparing the equality of two trees. For two trees T_1 and T_2, it is obvious that checking whether or not $T_1 = T_2$, can be computed in $O(|T_1|)$ time. The computation of s and n can be achieved in linear time. Additionally, the space required by the multi-sets returned by s and n is linear because it is sufficient to store pointers to the original trees. Regarding m, we can see in line 3 of Algorithm 1 that for a complete subtree $v \in A$, the function will only compare complete subtrees of size $|v|$ in B.

Since d_n matches only nodes (trees of size 1), it can be computed in linear time, therefore we will focus only on d_s. Each equality comparison for any complete subtree $v \in F$ will take at most $|v|$ steps, and will be executed at most $\frac{|T_2|}{|v|}$ times. This is because we will only match complete subtrees in F of size $|v|$ (Algorithm 1, line 3) and because Lemma 1 guarantees that $|T_2|$ has only $\frac{|T_2|}{|v|}$ complete subtrees of size $|v|$. Therefore, the multiplicity $m_{s(T_2)}(v)$ can be computed in $O(|v| \times \frac{|T_2|}{|v|}) = O(|T_2|)$ time. Since the multiplicity function m will be called by δ at most $|T_1|$ times, the function $mtd(T_1, T_2)$ can be computed in $O(|T_1| \times |T_2|)$ time. Since it is possible to create a multi-set of pointers to the original nodes of the tree, the space complexity for $mtd(T_1, T_2)$ is $O(|T_1| + |T_2|)$. □

Our current implementation uses hash tables to improve performance. Hash codes have greater pruning power than the pruning by tree size described above, however this is not enough to lower the complexity of the algorithm. In the experiments of section 5, we will see that in practice, our hash-based *mtd* implementation matches the performance of a linear q-gram based distance function.

5 Case Study

In this section, we compare our function against *BDist* [16] and *ted* [15]. Currently, we are not able to run *ted* on our approximate program matching framework[1] because of its enormous computational cost. Therefore, our experiments were executed on real tree data extracted from program fragments[12] but focus on the distance functions only. The procedure to extract these fragments is detailed in [12]. Our data-set[2] includes 244668 trees. The average depth is 4.75, the average number of nodes is 11.11, and the number of nodes range between 1 to 20 nodes. In the experiment, we randomly selected 1000 trees (queries) from the data-set and compared them against the rest of the data-set. About 244 million tree comparisons were performed.

We implemented the *ted* [5] that can be computed in $O(n^3)$ time, *BDist*[16] that can be computed in $O(n)$ time, and *mtd* that has a time complexity of $O(n^2)$. The distance functions were implemented in Java. The source code of the functions is available under the GPL license[3]. For the performance benchmarks, we loaded all the trees into memory and the time is counted after all the trees have been loaded. The experiments were executed on a Intel(R) Xeon(R) CPU 2.66 GHz with 4 processors. Four threads were created to reduce execution time. Each thread performs the same job.

Regarding the q-gram function, we used Yang's *BDist* function[16]. In this case, *BDist* will create $|T|$ grams from a tree T by creating for each node $v \in T$, a triple of the left child of v, v, and the right sibling of v. The original motivation of the paper was to create a similarity search engine by creating an inverted file of each q-gram. In this paper we simply calculated *BDist* in a sequential manner to show the real cost of a linear distance function. By using the similarity index presented by Yang, better performance results can be achieved for *BDist*. Finally, it can be proved that $BDist(T_1, T_2) \leq 5 \times ted(T_1, T_2)$[16].

Distance Evaluation

In Figure 3, we show the distribution of data according to distances between data and queries. In the y axis, the percentage of calculations that returned the distance shown in the x axis is displayed. We can see that all the functions maintain a similar, normal distribution.

In Figures 4(a) and 4(b), we compare respectively, how *mtd* and *BDist* vary from the result returned by *ted*. For *mtd*, the mean and standard deviation are closer than *BDist*. Additionally, maximum values for *mtd* tend to be smaller than for *BDist*. On the other hand, *Bdist*'s minimum values seem to be closer to *ted*. When *BDist* is used in its similarity search framework (and when a range r is provided), additional pruning can be performed by the pre-order and post-order counts of the node. In this case, *BDist* cannot satisfy the triangle inequality and that is why we did not include this pruning. *BDist* is returning

[1] `http://www.furiachan.org`
[2] `http://furia-chan.googlecode.com/files/trees1-20.tar.bz2`
[3] `http://furia-chan.googlecode.com/files/mtd-1.tar.bz2`

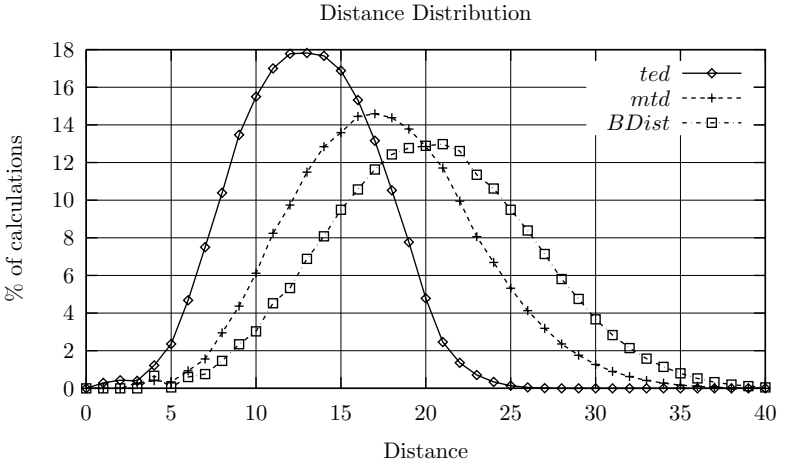


Fig. 3. Distribution of data according to distances between data and queries

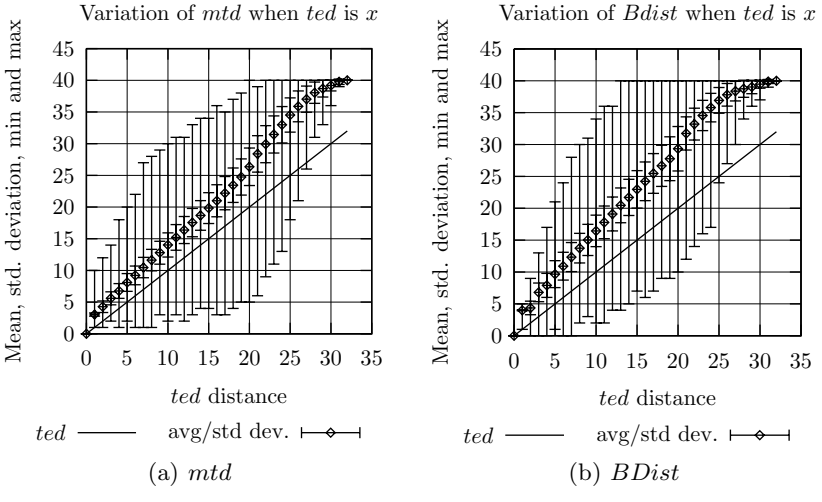


Fig. 4. Variation of *mtd* and $BDist$ with respect to the result x of *ted*

bigger results than *mtd* because subtree movements return lower values than for $BDist$ as siblings are not taken into account. For example in Figure 2, the distance returned by $BDist$ is 8 for T_1 and T_2. Since the shape of the distribution and the overall distance results are closer to *ted*, we can conclude that *mtd* performs better than $BDist$.

In Figures 5(a) and 5(b), we compare how d_s and d_n vary from the result returned by *ted*. We can see how d_s has bigger maximum values than *mtd* (Figure 4(a)). On the other hand, d_n has smaller minimum values than *mtd* or d_s. Finally, we can see that *mtd*'s minimum and maximum range is improved by the combination of d_s and d_n. The mean and standard deviation for *mtd* is

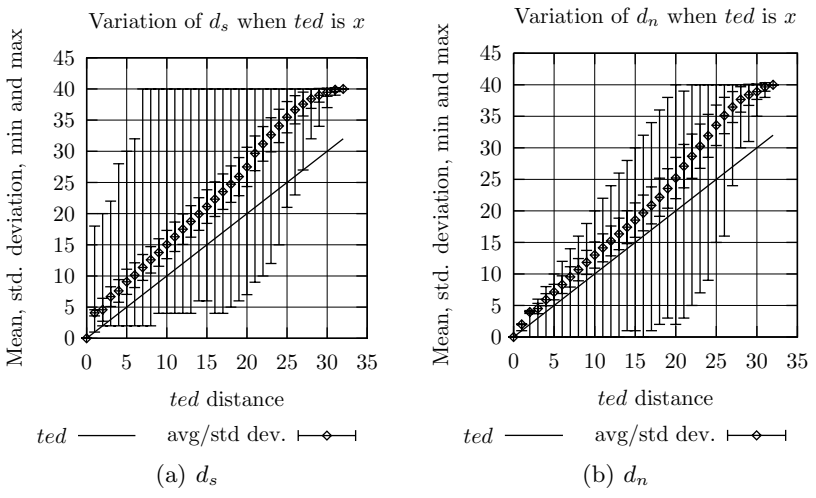


Fig. 5. Variation of d_s and d_n with respect to the result x of ted

centered between d_s and d_n. If we base our comparison on ted, the use of d_n is justified. It is interesting to see that the closest function to ted when considering only the standard deviation is d_n.

Benchmarks

For the same experiment described at the beginning of the section, we recorded the execution time for ted, $BDist$ and mtd. The function ted took about 85 hours to complete, $BDist$ took 8 minutes, and mtd took 7.4 minutes. The small difference between $BDist$ and mtd might be related to the hash function being called more times in $BDist$ than in mtd. On the other hand, mtd was implemented recursively and there is much room for improvement. One of the reasons ted is performing so poorly is that for each distance computation, many objects are being created (dynamic programming table objects and forests created and destroyed during the search). On the contrary, mtd and $BDist$ only have to access some static structures that are computed once during the lifetime of a tree. From the benchmarking results, can see how the actual performance of our function seems to be on par with a linear q-gram distance function.

6 Conclusions and Future Work

We have introduced a tree distance function that is based on multi-sets. When compared to ted, our function is more sensitive to changes. This is in general an undesirable property, however the performance gains are considerable enough to make it a worthwhile candidate for matching trees. Our function is fast because it can perform matchings without resorting to expensive dynamic programming table memory allocations. Our hash table based implementation achieved similar

performance than $BDist$[16], a q-gram based function that runs in time $O(n)$. Our distance function also has the added property of being a metric.

Regarding future works, because *mtd* is faster than *ted*, it can be employed by techniques like SMAP [14] to find a subset of the data suitable for pivoting. From that subset, a set of pivots based on *ted* could be computed. Finally, we would like to study the effect of adding N-grams greater than 1 to function n.

References

[1] Augsten, N., Bhlen, M., Gamper, J.: Approximate matching of hierarchical data using pq-grams. In: VLDB 2005, pp. 301–312 (2005)

[2] Bille, P.: A survey on tree edit distance and related problems. Theoretical Computer Science 337(1-3), 217–239 (2005)

[3] Chawathe, S.S., Garcia-Molina, H.: Meaningful change detection in structured data. SIGMOD Rec. 26(2), 26–37 (1997)

[4] Chawathe, S.S., Rajaraman, A., Garcia-Molina, H., Widom, J.: Change detection in hierarchically structured information. SIGMOD Rec. 25(2), 493–504 (1996)

[5] Demaine, E., Mosez, S., Rossman, B., Weimann, O.: An optimal decomposition algorithm for tree edit distance. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 146–157. Springer, Heidelberg (2007)

[6] Garofalakis, M., Kumar, A.: Xml stream processing using tree-edit distance embeddings. ACM Trans. Database Syst. 30(1), 279–332 (2005)

[7] Jiang, T., Wang, L., Zhang, K.: Alignment of trees - an alternative to tree edit. Theoretical Computer Science 143(1), 148–157 (1995)

[8] Kailing, K., Kriegel, H.-P., Schönauer, S., Seidl, T.: Efficient similarity search for hierarchical data in large databases. In: Bertino, E., Christodoulakis, S., Plexousakis, D., Christophides, V., Koubarakis, M., Böhm, K., Ferrari, E. (eds.) EDBT 2004. LNCS, vol. 2992, pp. 676–693. Springer, Heidelberg (2004)

[9] Klein, P., Tirthapura, S., Sharvit, D., Kimia, B.: A tree-edit-distance algorithm for comparing simple, closed shapes. In: SODA 2000, Philadelphia, USA. Society for Industrial and Applied Mathematics, pp. 696–704 (2000)

[10] Klein, P.N.: Computing the edit-distance between unrooted ordered trees. In: Bilardi, G., Pietracaprina, A., Italiano, G.F., Pucci, G. (eds.) ESA 1998. LNCS, vol. 1461, pp. 91–102. Springer, Heidelberg (1998)

[11] Müller-Molina, A.J., Shinohara, T.: On approximate matching of programs for protecting libre software. In: CASCON 2006, pp. 275–289. ACM Press, New York (2006)

[12] Müller-Molina, A.J., Shinohara, T.: Fast approximate matching of programs for protecting libre/open source software by using spatial indexes. In: SCAM 2007, pp. 111–122. IEEE Computer Society, Los Alamitos (2007)

[13] Ohkura, N., Hirata, K., Kuboyama, T., Harao, M.: The q-gram distance for ordered unlabeled trees. In: Hoffmann, A., Motoda, H., Scheffer, T. (eds.) DS 2005. LNCS (LNAI), vol. 3735, pp. 189–202. Springer, Heidelberg (2005)

[14] Shinohara, T., Ishizaka, H.: On dimension reduction mappings for approximate retrieval of multi-dimensional data. In: Arikawa, S., Shinohara, A. (eds.) Progress in Discovery Science. LNCS, vol. 2281, pp. 224–231. Springer, Heidelberg (2002)

[15] Tai, K.-C.: The tree-to-tree correction problem. JACM 26(3), 422–433 (1979)

[16] Yang, R., Kalnis, P., Tung, A.K.H.: Similarity evaluation on tree-structured data. In: SIGMOD 2005, pp. 754–765 (2005)

[17] Zhang, K.: Algorithms for the constrained editing distance between ordered labeled trees and related problems. Pattern Recognition 28(3), 463–474 (1995)
[18] Zhang, K.: Computing similarity between rna secondary structures. In: INTSYS 1998, pp. 126–132 (1998)
[19] Zhang, K., Statman, R., Shasha, D.: On the editing distance between unordered labeled trees. Information Processing Letters 42(3), 133–139 (1992)

Sibling Distance for Rooted Labeled Trees*

Taku Aratsu[1], Kouichi Hirata[2], and Tetsuji Kuboyama[3]

[1] Graduate School of Computer Science and Systems Engineering
[2] Department of Artificial Intelligence
Kyushu Institute of Technology
Kawazu 680-4, Iizuka 820-8502, Japan
{aratsu,hirata}@dumbo.ai.kyutech.ac.jp
[3] Computer Center, Gakushuin University
Mejiro 1-5-1, Toshima, Tokyo 171-8588, Japan
kuboyama@gakushuin.ac.jp

Abstract. In this paper, we introduce a *sibling distance* δ_s for rooted labeled trees as an L_1-distance between their *sibling histograms*, which consist of the frequencies of every pair of the label of a node and the sequence of labels of its children. Then, we show that δ_s gives a *constant factor lower bound* on the tree edit distance δ such that $\delta_s(T_1, T_2) \leq 4\delta(T_1, T_2)$. Next, we design the algorithm to compute the sibling histogram in $O(n)$ time for *ordered* trees and in $O(gn)$ time for *unordered* trees, where n and g are the number of nodes and the degree of a tree. Finally, we give experimental results by applying the sibling distance to glycan data.

1 Introduction

It is one of the important tasks for data mining from tree-structured data such as HTML and XML data in web mining or DNA and glycan data in bioinformatics to introduce a *similarity measure* between two *rooted labeled trees* (*trees*, for short), and compare them based on the similarity measure. Here, a tree is *ordered* if a left-to-right order for the children of each node is given; *unordered* otherwise.

We can divide such similarity measures between two trees into two types. One type is the similarity measure based on *the maximization of the size of a common pattern*. In this type, two trees sharing a larger common pattern are regarded as more similar. The most famous similarity measure of this type is the *tree edit distance* [3,12]. While it is a metric, computing the tree edit distance is not efficient; $O(n^3)$ time [4] for ordered trees, where n is the maximum number of nodes of two trees, and intractable for unordered trees [11,13].

Another type is the similarity measure based on *the maximization of the frequencies of common patterns*. In this type, two trees sharing a larger number of common patterns are regarded as more similar. Such a similarity measure is formulated by the *leaf histogram*, the *degree histogram* and the *label histogram* [5], the frequency of *binary branches* (through the binary tree representation for

* This work is partially supported by Grand-in-Aid for Scientific Research 17200011, 19300046 and 20500126 from the Ministry of Education, Culture, Sports, Science and Technology, Japan.

S. Chawla et al. (Eds.): PAKDD 2008 Workshops, LNAI 5433, pp. 99–110, 2009.

ordered trees) [10], the *q-gram* [7,8], the *bifoliate q-gram* [6], or the *pq-gram* [2]. While none of them is a metric, we can compute all of them by traversing a single tree just once, which implies more efficient computing. Furthermore, some of them give *constant factor lower bounds* on the tree edit distance and can be applied to unordered trees with preserving efficiency. We summarize such similarity measures in Figure 1.

In this paper, we introduce a new similarity measure between (both ordered and unordered) trees based on the maximization of the frequency of common patterns. First, we introduce a *sibling sequence* of a node as a pair of the label of the node and the sequence of labels of its children, and a *sibling histogram* as a histogram of the frequencies of sibling sequences for every node. Then, we formulate a *sibling distance* δ_s as an L_1-distance between sibling histograms.

Note that δ_s is a natural extension of the distance based on the degree histogram [5], denoted by δ_d. While δ_s is not a metric as similar as other similarity measures based on the maximization of the frequencies of common patterns, we show that δ_s gives a *constant factor lower bound* on the tree edit distance δ such that $\delta_s(T_1, T_2) \leq 4\delta(T_1, T_2)$. In particular, we show that $\delta_d(T_1, T_2) \leq \delta_s(T_1, T_2)$.

Next, by representing a tree as its depth and label sequences based on *postorder traversal*, we design the algorithm to compute the sibling histogram in $O(n)$ time for *ordered* trees and in $O(gn)$ time for *unordered* trees, where n and g are the number of nodes and the degree of a tree. Also the algorithm requires $O(dg)$ space, where d is the depth of a tree. Furthermore, if the number of labels in a tree is bounded by some constant, then we can reduce the space complexity $O(dg)$ to $O(d)$ for ordered trees and $O(d + g)$ for unordered trees.

Finally, in order to evaluate that the sibling distance δ_s is appropriate to approximate the tree edit distance δ, we give experimental results by computing the above distances (and δ_d) for glycan data.

the basis of the similarity measure	computation time		constant factor lower bound on δ
	ordered	unordered	
tree edit distance	$O(n^3)$ [4]	intractable [11,13]	(δ)
leaf histogram [5]	$O(n)$	$O(n)$	$\leq \delta$
degree histogram [5]	$O(n)$	$O(n)$	$\leq 3\delta$
label histogram [5]	$O(n)$	$O(n)$	$\leq 2\delta$
binary branch [10]	$O(n)$	not applicable	$\leq 5\delta$ (ordered)
q-gram [7,8]	$O(qln)$	$O(qln)$	not exist
bifoliate q-gram [6]	$O(qd\min(q,d)ln)$	$O(qd\min(q,d)ln)$	not exist
pq-gram [2]	$O(pqn)$	not applicable	not exist
sibling histogram	$O(n)$	$O(gn)$	$\leq 4\delta$

Fig. 1. Summary of the similarity measures for trees based on the maximization of the frequencies of common patterns. Here, n, l, d and g denote the number of nodes, the number of leaves, the depth and the degree of a tree.

2 Preliminary

A *tree* is a connected graph without cycles. For a tree $T = (V, E)$, we sometimes denote $v \in T$ instead of $v \in V$, and $|T|$ instead of $|V|$. A *rooted tree* is a tree with one node r chosen as its *root*. For each nodes v and u in T, let $UP_v(u)$ be the unique path from v to u in T.

For a root r of T, we call the number of edges in $UP_r(v)$ the *depth* of v (in T) and denote it by $dep(v)$. In particular, since $UP_r(r)$ has no edges, we set $dep(r) = 0$. For a tree T, we call $\max\{dep(v) \mid v \in T\}$ the *depth* of T and denote it by $dep(T)$. The *parent* of $v(\neq r)$ is its adjacent node on $UP_r(v)$. We say that u is a *child* of v if v is the parent of u. A *leaf* is a node having no children. Two nodes are *siblings* if they have the same parent. The number of all children of a node v is called the *degree* of v and denoted by $deg(v)$. For a tree T, $\max\{deg(v) \mid v \in T\}$ is called the *degree* of T and denoted by $deg(T)$.

A rooted tree $T = (V, E)$ is *labeled* (by an alphabet Σ of labels) if there exists an onto function $l : V \to \Sigma$ such that $l(v) = a$ $(v \in V, a \in \Sigma)$. A tree is *ordered* if a left-to-right order for the children of each node is given; *unordered* otherwise. In this paper, we call a rooted labeled ordered tree and a rooted labeled unordered tree a tree and an unordered tree, respectively.

Next, we introduce the *tree edit distance* [3,4,11,12,13]. Let $\varepsilon \notin \Sigma$ denote a special *blank* symbol and define $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$. We often define a *cost function* $\gamma : (\Sigma_\varepsilon \times \Sigma_\varepsilon) \setminus (\varepsilon, \varepsilon) \mapsto \mathbf{R}$ on pairs of labels.

Definition 1 (Edit operations for ordered trees). Let T be an ordered labeled tree. Then, we call the following tree operations *edit operations for ordered trees*. See Figure 2.

1. Substitution: Change the label of the node v in T.
2. Deletion: Delete a non-rooted node v in T with parent v', making the children of v become the children of v'. The children are inserted in the place of v as a subsequence in the left-to-right order of the children of v'.
3. Insertion: The complement of deletion. Insert a node v as a child of v' in T making v the parent of a consecutive subsequence of the children of v'.

Here, we represent each edit operation by $(l_1 \mapsto l_2)$, where $(l_1, l_2) \in (\Sigma_\varepsilon \times \Sigma_\varepsilon) \setminus (\varepsilon, \varepsilon)$. The operation is a substitution if $l_1 \neq \varepsilon$ and $l_2 \neq \varepsilon$, a deletion if $l_2 = \varepsilon$, and an insertion if $l_1 = \varepsilon$. Also we extend the notation such that $(v \mapsto w)$ for nodes v and w denotes $(label(v) \mapsto label(w))$.

For unordered trees, we can define the operations similarly, by replacing a *subsequence* with a *permutation* in the deletion and the insertion.

We constrain a cost function γ to be a distance metric. For a cost function γ, we define the cost of an edit operation by setting $\gamma(l_1 \mapsto l_2) = \gamma(l_1, l_2)$. The cost of a sequence $S = s_1, \ldots, s_k$ of edit operations is given by $\gamma(S) = \sum_{i=1}^{k} \gamma(s_i)$. Then, the *tree edit distance* $\delta(T_1, T_2)$ between T_1 and T_2 is defined as follow:

$$\delta(T_1, T_2) = \min \left\{ \gamma(S) \,\middle|\, \begin{array}{l} S \text{ is a sequence of edit operations} \\ \text{transforming from } T_1 \text{ to } T_2 \end{array} \right\}.$$

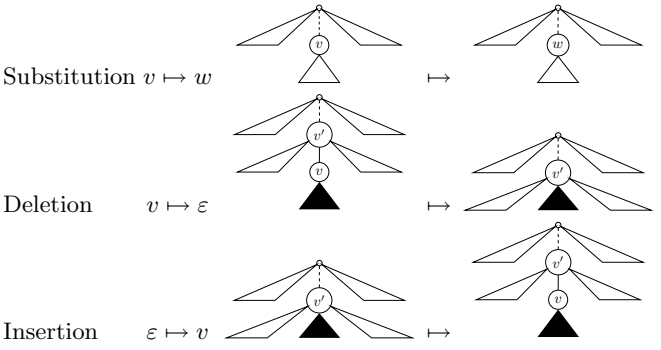


Fig. 2. Edit operations for ordered trees

3 Sibling Distance

In this section, we introduce a *sibling sequence* and a *sibling distance*, and show the several properties of the sibling distance.

Definition 2 (Sibling sequence). Let v be a node in a tree T with label function l and $v_1, \ldots, v_n$ the sequence of all children of v from left-to-right. We denote the sequence $l(v_1) \cdots l(v_n)$ of labels by $cl(v)$. Then, a pair $\langle l(v), cl(v) \rangle$ is called a *sibling sequence* of v. The sibling sequence of a leaf v is $\langle l(v), \varepsilon \rangle$.

Definition 3 (Sibling histogram and sibling distance). The histogram for the frequencies of sibling sequences for every node in T is called a *sibling histogram* and we denote it by $sh(T)$. Also we call the L_1-distance between $sh(T_1)$ and $sh(T_2)$ a *sibling distance* between T_1 and T_2, and denote it by $\delta_s(T_1, T_2)$.

In other words, let Σ be an alphabet and $g = \max\{deg(T_1), deg(T_2)\}$. Also let $f(T, \langle a, w \rangle)$ denote the frequency of a sibling sequence $\langle a, w \rangle$ in T, where $a \in \Sigma$ and $w \in \Sigma^g$. Then, the *sibling distance* between T_1 and T_2 is defined as follows:

$$\delta_s(T_1, T_2) = \sum_{a \in \Sigma, w \in \Sigma^g} |f(T_1, \langle a, w \rangle) - f(T_2, \langle a, w \rangle)|.$$

Example 1. Consider the trees T_1 and T_2 in Figure 3, where $\delta(T_1, T_2) = 3$. Then, we obtain the following sibling histograms $sh(T_1)$ and $sh(T_2)$ of T_1 and T_2, where we omit the case that the frequency is 0.

$\langle l(v),\ cl(v) \rangle$	freq.
$\langle a,\ bbe \rangle$	1
$\langle b,\ cd \rangle$	2
$\langle c,\ \varepsilon \rangle$	2
$\langle d,\ \varepsilon \rangle$	2
$\langle e,\ \varepsilon \rangle$	1

$sh(T_1)$

$\langle l(v),\ cl(v) \rangle$	freq.
$\langle a,\ bcde \rangle$	1
$\langle b,\ cdb \rangle$	1
$\langle b,\ e \rangle$	1
$\langle c,\ \varepsilon \rangle$	2
$\langle d,\ \varepsilon \rangle$	2
$\langle e,\ \varepsilon \rangle$	2

$sh(T_2)$

Hence, it holds that $\delta_s(T_1, T_2) = 7$.

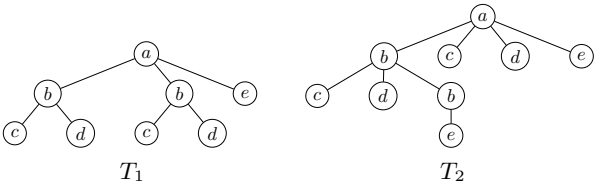


Fig. 3. Trees T_1 and T_2 in Example 1

Proposition 1. *The sibling distance δ_s is a pseudo-metric, that is:*

1. $\delta_s(T_1, T_2) \geq 0$ *and* $\delta_s(T_1, T_1) = 0$.
2. $\delta_s(T_1, T_2) = \delta_s(T_2, T_1)$.
3. $\delta_s(T_1, T_3) \leq \delta_s(T_1, T_2) + \delta_s(T_2, T_3)$.

Proof. Since the statements 1 and 2 are obvious, we just show the statement 3. Let g be $\max\{deg(T_1), deg(T_2), deg(T_3)\}$ and Σ an alphabet of labels in T_1, T_2 and T_3. Then, we obtain the following sequence.

$$\begin{aligned}\delta_s(T_1, T_3) &= \sum_{a\in\Sigma, w\in\Sigma^g} |f(T_1, \langle a, w\rangle) - f(T_3, \langle a, w\rangle)| \\ &\leq \sum_{a\in\Sigma, w\in\Sigma^g} |f(T_1, \langle a, w\rangle) - f(T_2, \langle a, w\rangle)| \\ &\quad + \sum_{a\in\Sigma, w\in\Sigma^g} |f(T_2, \langle a, w\rangle) - f(T_3, \langle a, w\rangle)| \\ &= \delta_s(T_1, T_2) + \delta_s(T_2, T_3).\end{aligned}$$

□

Example 2. Unfortunately, δ_s is *not a metric*, that is, there exist two trees T_1 and T_2 such that $\delta_s(T_1, T_2) = 0$ but T_1 and T_2 are not isomorphic as Figure 4.

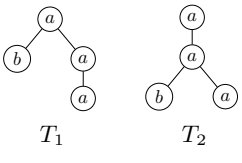


Fig. 4. Non-isomorphic trees T_1 and T_2 such that $\delta_s(T_1, T_2) = 0$

The sibling distance δ_s gives a *constant factor lower bound* on the tree edit distance δ as follows.

Theorem 1 (Constant factor lower bound on δ). $\delta_s(T_1, T_2) \leq 4\delta(T_1, T_2)$.

Proof. Let v be a node in T with the parent v_0. It is sufficient to show that the sibling distance δ_s changes at most 4 when an edit operation is applied.

(1) Deletion : Let T' be a tree obtained by deleting a node v in T. When transforming from T to T', the frequencies of $\langle l(v), cl(v)\rangle$ and $\langle l(v_0), cl(v_0)\rangle$ in

T decrease 1, and the frequency of $\langle l(v_0), cl(v_0)\rangle$ in T' increases 1. Hence, it holds that $\delta(T, T') \leq 3$.

(2) Substitution : Let T' be a tree obtained by changing the label of a node v in T from l to l' such that $l(u) = l'(u)$ if $u \neq v$ and $l(v) \neq l'(v)$. When transforming from T to T', the frequencies of $\langle l(v), cl(v)\rangle$ and $\langle l(v_0), cl(v_0)\rangle$ in T decrease 1, and the frequencies of $\langle l'(v), cl(v)\rangle$ and $\langle l'(v_0), cl(v_0)\rangle$ in T' increase 1. Hence, it holds that $\delta(T, T') \leq 4$. □

Let $D(T, m)$ be the number of nodes with degree m in T. Then, Kailing's distance δ_d based on the *degree histogram* [5], called the *degree distance* in this paper, is defined as follows:

$$\delta_d(T_1, T_2) = \sum_{m=0}^{\max\{deg(T_1), deg(T_2)\}} |D(T_1, m) - D(T_2, m)|.$$

It is known that $\delta_d(T_1, T_2) \leq 3\delta(T_1, T_2)$ [5]. Furthermore, we can characterize the relationship between δ_s and δ_d as follows.

Theorem 2. $\delta_d(T_1, T_2) \leq \delta_s(T_1, T_2)$.

Proof. Note that $|cl(v)| = m$ if $deg(v) = m$. Then, for every T and m, it holds that $D(T, m) = \sum_{|w|=m} f(T, \langle a, w\rangle)$. Hence, the following sequence holds.

$$\begin{aligned} |D(T_1, m) - D(T_2, m)| &= \left| \sum_{|w|=m} f(T_1, \langle a, w\rangle) - \sum_{|w|=m} f(T_2, \langle a, w\rangle) \right| \\ &\leq \sum_{|w|=m} |f(T_1, \langle a, w\rangle) - f(T_2, \langle a, w\rangle)|. \end{aligned}$$

□

Note that the sibling sequence, the sibling histogram and the sibling distance can be applied to not only ordered trees but also unordered trees, with sorting w by a lexicographic order on Σ for every sibling sequence $\langle a, w\rangle$. Then, Theorem 1 and 2 also hold for unordered trees.

4 Algorithm to Compute a Sibling Histogram

In this section, we design the algorithm to compute sibling histograms. In the following, for a tree T, we assume that $n = |T|$, $d = dep(T)$ and $g = deg(T)$.

First note that we can compute the sibling histogram for an *ordered tree* in $O(gn)$ time, by traversing every node v in T and then by collecting all of the labels of children of v. On the other hand, in this section, we design a more efficient algorithm by using *depth* and *label sequences* under *postorder* traversal.

Let T be a tree with the root v and the children $v_1, \ldots, v_m$ of v. The *postorder traversal* (*postorder*, for short) of T is obtained by visiting v_i $(1 \leq i \leq m)$ in order, recursively, and then visiting v. Furthermore, suppose that the sequence

$v_1 \cdots v_n$ is the postorder of T. Then, we formulate the *depth sequence* $D(T)$ and the *label sequence* $L(T)$ of T as follow.

$$D(T) = dep(v_1) \cdots dep(v_n),\ L(T) = l(v_1) \cdots l(v_n).$$

We design the algorithm *SibHist* in Algorithm 1 to compute a sibling histogram from the depth and the label sequences of a given tree. Here, $D[i]$ and $L[i]$ denotes the i-th element of a depth sequence D and a label sequence L, that is, $D[i] = dep(v_i)$ and $L[i] = l(v_i)$. Also $SH[l][w]$ stores the frequency of the sibling sequence $\langle l, w \rangle$. The algorithm *SibHist* uses an array S of strings with the size $dep(T)$. If S is an empty array $\emptyset$, then we assume that every $S[i]$ is set to an empty string ε. Furthermore, "$\cdot$" denotes the concatenation of strings.

```
   procedure SibHist(D,L)
       /* D: a depth sequence, L: a label sequence */
1      S ← ∅; SH ← ∅; /* S: an array of strings, SH[l][w]: the frequency of ⟨l, w⟩*/
2      pd ← 0; /* pd: the depth of the previous node in postorder traversal */
3      for i = 1 to |D| do
4          if pd = D[i] then
5              S[D[i]] ← S[D[i]]·L[i]; SH[L[i]][ε]++;
6          if pd < D[i] then
7              S[D[i]] ← L[i]; SH[L[i]][ε]++;
8          else
               /* pd > D[i], that is, pd = D[i] + 1 */
9              SH[L[i]][S[pd]]++; S[pd] ← ε; S[D[i]] ← S[D[i]]·L[i];
10         pd ← D[i];
11     return SH;
```

Algorithm 1. *SibHist*

Example 3. Consider the trees T_1 and T_2 in Example 1 (Figure 3). Then, the depth and the label sequences of T_i are given as follows.

$$D(T_1) = 22122110,\ D(T_2) = 223211110,$$
$$L(T_1) = cdbcdbea,\quad L(T_2) = cdebbcdea.$$

The results applying them to the algorithm *SibHist* are described as Figure 5.

Theorem 3. *The algorithm SibHist correctly computes the sibling histogram of a tree T in $O(n)$ time and in $O(dg)$ space.*

Proof. First, we show the correctness of the algorithm *SibHist*. Remember that $D[i] = dep(v_i)$. For $i > 1$, it holds that $pd = D[i-1]$. Since the depth sequence is postorder, it holds that $pd \leq D[i]$ if and only if v_i is a leaf.

Consider the case $pd = D[i]$. Suppose that v_j is a parent of v_{i-1}. By postorder, it is obvious that $j > i$, so v_j does not appear until the i-th iteration in the algorithm *SibHist*. Then, it holds that $pd = D[i]$ if and only if v_j is a parent of v_i. In this case, for the sibling sequence $\langle l(v_j), w \rangle$ of v_j, w contains the substring

T_1

i	1	2	3	4	5	6	7	8
$D[i]$	2	2	1	2	2	1	1	0
$L[i]$	c	d	b	c	d	b	e	a
pd	0	2	2	1	2	2	1	1
$pd \circ D[i]$	$<$	$=$	$>$	$<$	$=$	$>$	$=$	$>$
SH	$\langle c, \varepsilon\rangle$	$\langle d, \varepsilon\rangle$	$\langle b, cd\rangle$	$\langle c, \varepsilon\rangle$	$\langle d, \varepsilon\rangle$	$\langle b, cd\rangle$	$\langle e, \varepsilon\rangle$	$\langle a, bbe\rangle$
$S[0]$	ε	ε	ε	ε	ε	ε	ε	a
$S[1]$	ε	ε	b	b	b	bb	bbe	ε
$S[2]$	c	cd	ε	c	cd	ε	ε	ε

T_2

i	1	2	3	4	5	6	7	8	9
$D[i]$	2	2	3	2	1	1	1	1	0
$L[i]$	c	d	e	b	b	c	d	e	a
pd	0	2	2	3	2	1	1	1	1
$pd \circ D[i]$	$<$	$=$	$<$	$>$	$>$	$=$	$=$	$=$	$<$
SH	$\langle c, \varepsilon\rangle$	$\langle d, \varepsilon\rangle$	$\langle e, \varepsilon\rangle$	$\langle b, e\rangle$	$\langle b, cdb\rangle$	$\langle c, \varepsilon\rangle$	$\langle d, \varepsilon\rangle$	$\langle e, \varepsilon\rangle$	$\langle a, bcde\rangle$
$S[0]$	ε	ε	ε	ε	ε	ε	ε	ε	a
$S[1]$	ε	ε	ε	ε	b	bc	bcd	$bcde$	ε
$S[2]$	c	cd	cd	cdb	ε	ε	ε	ε	ε
$S[3]$	ε	ε	e	ε	ε	ε	ε	ε	ε

Fig. 5. The results applying T_1 and T_2 to the algorithm *SibHist*

$l(v_{i-1})l(v_i)$. Furthermore, since $S[D[i]]$ is corresponding to the substring of w ends the label $l(v_{i-1})$, the algorithm *SibHist* concatenates $S[D[i]]$ to $l(v_i)$ and then increments the frequency of $\langle l(v_i), \varepsilon\rangle$ since v_i is a leaf.

Consider the case $pd < D[i]$. Suppose that v_j is a parent of v_i, where v_j does not appear in the i-th iteration in the algorithm *SibHist*. In this case, for the sibling sequence $\langle l(v_j), w\rangle$ of v_j, w contains the label $l(v_i)$ and, in particular, starts the label $l(v_i)$. Then, the algorithm *SibHist* substitutes $l(v_i)$ to $S[D[i]]$, and then increments the frequency of $\langle l(v_i), \varepsilon\rangle$ since v_i is a leaf.

Consider the case $pd > D[i]$. In this case, it always holds that $pd = D[i] + 1$, so v_i is a parent of v_{i-1}, and v_{i-1} is the right-most child of v_i. Since $S[d]$ ends the label $l(v_{i-1})$ and w is the sequence of siblings of v_i, the algorithm *SibHist* increments the frequency of $\langle l(v_i), w\rangle$. Furthermore, it stores the sibling of v_i by concatenating $S[D[i]]$ to $l(v_i)$. Hence, the algorithm *SibHist* runs correctly.

It is obvious that the running time of the algorithm *SibHist* is $O(n)$. Furthermore, since the size of the array S is d and the size of the element of the array is at most g, the space complexity of the algorithm *SibHist* is $O(dg)$. □

If $|\Sigma|$ is bounded by some constant k, then we can reduce the space complexity in the algorithm *SibHist* by using an *integer encode* of a string as a k-ary integer as follows. Suppose that $\Sigma = \{a_1, \ldots, a_k\}$. Then, the *integer code* $\widehat{a_i}$ of a label a_i is defined as $\widehat{a_i} = i$. Also the *integer code* $\widehat{w}$ of a string $w = v_1 \cdots v_n$ is defined as $\widehat{w} = \sum_{i=1}^{n} \widehat{v_i} k^{n-i}$, where $\widehat{\varepsilon} = 0$ (*cf.*, [9]). For the concatenation $w{\cdot}a$ of a string w and a label a in lines 5 and 9, we can compute the integer code $\widehat{w{\cdot}a}$ as $\widehat{w}k + \widehat{a}$. Hence, we can extend Theorem 3 as follows.

Corollary 1. *If $|\Sigma|$ is bounded, then we can compute the sibling histogram of a tree T in $O(n)$ time and in $O(d)$ space.*

On the other hand, the size $O(d)$ of an array in the algorithm *SibHist* is tight, because there exists the tree described in Figure 6 necessary to use the array with size d.

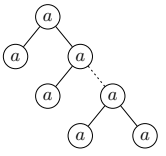


Fig. 6. A tree necessary to use the array in the algorithm *SibHist* of which size coincides with the depth of the tree

Next, consider how to compute the sibling histogram for an *unordered tree*. By dealing with unordered trees as ordered trees under some order of children, we can compute a sibling histogram for an unordered tree in $O(g(\log g)n)$ time, by traversing every node v in T and then by collecting all of the labels of children of v. Here, $O(g \log g)$ denotes the time to sort all of the labels of children.

On the other hand, we can design a more efficient algorithm, named as *USibHist*, as an extension of the algorithm *SibHist* to unordered trees. First note that it is necessary for the algorithm *USibHist* to store an element of S after sorting lexicographically under a fixed order on an alphabet Σ of labels, which can be realized as an insertion of a label of the node into an appropriate position. This process requires $O(g)$ time for every node, so the total time of the algorithm *USibHist* is $O(gn)$. As similar as Corollary 1, if $|\Sigma|$ is bounded, then it is necessary for *USibHist* to decode an integer code of an element in S when applying the insertion, so we can reduce the space complexity from $O(dg)$ to $O(d + g)$.

Corollary 2. *We can compute the sibling histogram of an unordered tree T in $O(gn)$ time and in $O(dg)$ space. Furthermore, if $|\Sigma|$ is bounded, then we can reduce the space complexity to $O(d + g)$.*

5 Experimental Results

In this section, we apply the sibling distance to N-glycan data from KEGG[1], the number of which data is 2151. Then, we compute the sibling distance, the degree distance and the tree edit distance between every pair of N-glycan data. Since the difference of the sibling distance between ordered trees and unordered trees is negligible (about 1%), we deal here with just the latter.

First, we analyze the distributions of the tree edit distance δ (for ordered trees), the degree distance δ_d and the sibling distance δ_s between N-glycan data. In Figure 7, the x-axis is the distance and the y-axis is the ratio of the number of pairs of N-glycan data with distance (plotted on the x-axis) to the number of all pairs of N-glycan data.

[1] KEGG: Kyoto Encyclopedia of Genes and Genomes, `http://www.genome.jp/kegg`

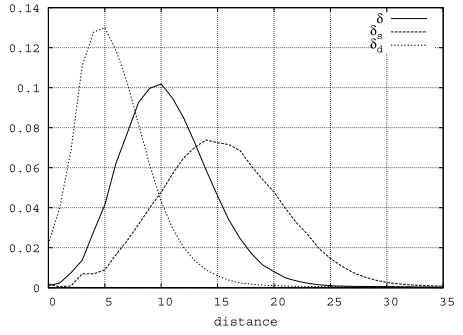


Fig. 7. The distributions of δ, δ_d and δ_s between N-glycan data

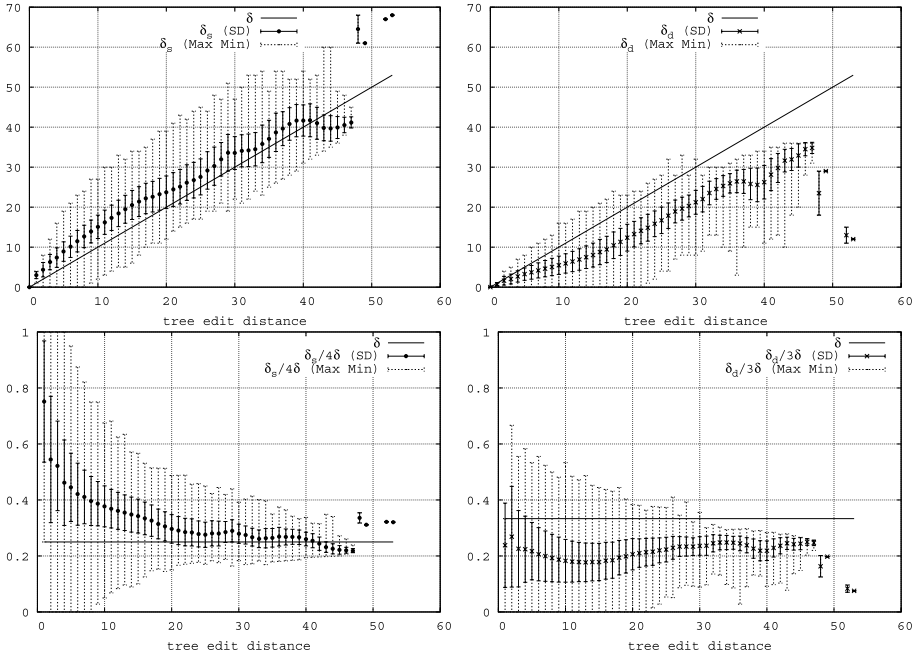


Fig. 8. The distributions of δ_d and δ_s for δ (upper) and of $\delta_s/4\delta$ and $\delta_d/3\delta$ (lower)

Figure 7 shows that all of the distributions are nearly normal, and the distribution of δ is located to the right of the distribution of δ_d and to the left of the distribution of δ_s.

Next, we investigate the relationship among δ, δ_s and δ_d. Figure 8 (upper) describes the distributions of δ_s (left) and δ_d (right) for δ, respectively, where the x-axis is the value of δ, and the y-axis is the distance δ_s (left) and δ_d (right) between the pairs of trees with the tree edit distance δ plotted on the x-axis. The dashed vertical lines represent the ranges from minimum to minimum values, and the solid vertical lines represent the standard deviations of the mean values.

Furthermore, we investigate experimentally the constant lower bounds that $\delta(T_1, T_2) \leq 3\delta_d(T_1, T_2)$ [5] and $\delta(T_1, T_2) \leq 4\delta_s(T_1, T_2)$ in Theorem 1. Figure 8 (lower) shows the distributions of $\delta_s/4\delta$ (left) and $\delta_d/3\delta$ (right) for tree edit distance δ, respectively, where the y-axis is the values of $\delta_s/4\delta$ (left) and $\delta_d/3\delta$ (right), and the others are the same as in Figure 8 (upper). The solid horizontal lines indicate 1/4 (left) and 1/3 (right) corresponding to the tree edit distance.

Figure 8 (upper) shows that the distributions of δ_s are closer to the values of δ than ones of δ_d. In particular, most of distributions of δ_s are upper than δ, while all distributions of δ_d are lower than δ.

On the other hand, Figure 8 (lower) implies that the mean values of $\delta_s/4\delta$ tend to get close to the values of δ when it increases, while the average value of $\delta_d/3\delta$ is always lower than the value of δ. The reason why $\delta_s/4\delta$ becomes larger than $\delta_d/3\delta$ for the value of a small value of δ is that the value of δ_s increases sensitively by applying all of the edit operations, while the value of δ_d does not change when the relabeling is applied.

We can observe in Figure 8 that the ranges for $\delta \geq 48$ are far from ones for $\delta < 48$. The number of pairs that $\delta \geq 48$ is just 6 and such pairs consist of the N-glycan G03655 (the number of nodes is 34) and one of the N-glycans G04045 (36), G04206 (37), G09054 (31), G10095 (31), G11846 (38) and G11347 (37). Since all the nodes in G03655 are labeled with "Mannose" while the other N-glycans in the pairs contain just 3 nodes with label "Mannose," the values of δ_s and $\delta_s/4\delta$ are large. On the other hand, since the difference between degree histograms of such pairs is not larger than one between the occurrences of labels, the values of δ_d and $\delta_d/3\delta$ are small.

Hence, we can conclude that the sibling distance δ_s is more appropriate to approximate the tree edit distance δ than the degree distance δ_d for N-glycan data, in particular, when the value of δ is large.

6 Conclusion

In this paper, we have introduced a *sibling distance* as the similarity measure for both ordered and unordered trees. Then, we have shown that it gives a constant factor lower bound on the tree edit distance, and can be computed efficiently. Finally, we have given the experimental results that the sibling distance is more appropriate to approximate the tree edit distance than the degree distance.

It is a future work to compare the sibling distance with other similarity measures in Figure 1, in particular, the binary branch distance [10]. Also it is a future work to obtain the experimental advantage of the sibling distance, by applying it to other tree-structured data.

Since the sibling distance is not a metric, it does not have any upper bound on the tree edit distance. Recently, Akutsu [1] has given both lower and upper bounds on the tree edit distance for the string edit distance between *Euler strings* of *ordered trees*, which can be computed in $O(n^2)$ time. It is an important future work to formulate a distance with both lower and upper bounds that can be computed in nearly $O(n)$ time and applied to both ordered and unordered trees.

References

1. Akutsu, T.: A relationship between edit distance for ordered trees and edit distance for Euler strings. Inform. Proc. Let. 100, 105–109 (2006)
2. Augsten, N., Böhlen, M., Gamper, J.: Approximate matching of hierarchical data using pq-grams. In: Proc. VLDB 2005, pp. 301–312 (2005)
3. Bille, P.: A survey on tree edit distance and related problems. Theoret. Comput. Sci. 337, 217–239 (2005)
4. Demaine, E.D., Mozes, S., Rossman, B., Weimann, O.: An optimal decomposition algorithm for tree edit distance. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 146–157. Springer, Heidelberg (2007)
5. Kailing, K., Kriegel, H.-P., Schönauer, S., Seidl, T.: Efficient similarity search for hierarchical data in large databases. In: Bertino, E., Christodoulakis, S., Plexousakis, D., Christophides, V., Koubarakis, M., Böhm, K., Ferrari, E. (eds.) EDBT 2004. LNCS, vol. 2992, pp. 676–693. Springer, Heidelberg (2004)
6. Kuboyama, T., Hirata, K., Aoki-Kinoshita, K.F.: An efficient unordered tree kernel and its application to glycan classification. In: Washio, T., Suzuki, E., Ting, K.M., Inokuchi, A. (eds.) PAKDD 2008. LNCS (LNAI), vol. 5012, pp. 184–195. Springer, Heidelberg (2008)
7. Kuboyama, T., Hirata, K., Ohkura, N., Harao, M.: A q-gram based distance measure for ordered labeled trees. In: Proc. LLLL 2006, pp. 77–83 (2006)
8. Ohkura, N., Hirata, K., Kuboyama, T., Harao, M.: The q-gram distance for ordered unlabeled trees. In: Hoffmann, A., Motoda, H., Scheffer, T. (eds.) DS 2005. LNCS (LNAI), vol. 3735, pp. 189–202. Springer, Heidelberg (2005)
9. Ukkonen, E.: Approximate string-matching with q-grams and maximal matches. Theor. Comput. Sci. 92, 191–211 (1993)
10. Yang, R., Kalnis, P., Tung, A.K.H.: Similarity evaluation on tree-structured data. In: Proc. SIGMOD 2005, pp. 754–765 (2005)
11. Zhang, K., Jiang, T.: Some MAX SNP-hard results concerning unordered labeled trees. Inform. Process. Let. 49, 249–254 (1994)
12. Zhang, K., Shasha, D.: Simple fast algorithms for the editing distance between trees and related problems. SIAM J. Comput. 18, 1245–1262 (1989)
13. Zhang, K., Statman, R., Shasha, D.: On the editing distance between unordered labeled trees. Inform. Process. Let. 42, 133–139 (1992)

Kernel Functions Based on Derivation

Koichiro Doi and Akihiro Yamamoto

Graduate School of Informatics, Kyoto University,
Yoshida-honmachi, Sakyo-ku, Kyoto 606-8501, Japan
{doi,akihiro}@i.kyoto-u.ac.jp

Abstract. In this paper we explain the fundamental idea of designing a class of kernel functions, called the intentional kernel, for structured data. The intentional kernel is designed with the property that every structured data is defined by derivation. Derivation means transforming a data or an expression into another. Typical derivation can be found in the field of formal language theory: A grammar defines a language in the sense that a sequence belongs to a language if it is transformed from a starting symbol by repeated application of the production rules in the grammar. Another example is in mathematical logic: A formula is proved if it is obtained from axioms by repeated application of inference rules. Combining derivation with the kernel-based learning mechanism derives the class of the intentional kernel.

1 Introduction

Kernel functions are very powerful tool for learning mechanisms that construct linear separating functions by using dot-products. Support Vector Machine (SVM) is the most popular in such mechanisms [1,14]. A *kernel function* is defined as

$$K(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\phi}(\mathbf{x}_i) \cdot \boldsymbol{\phi}(\mathbf{x}_j),$$

where $\boldsymbol{\phi}$ maps every data x to a point in $\mathbf{R}^d$ ($d \leq \infty$). The mapping is used for not only constructing non linear separating function but also treating non-numerical data, which we call structured data, such as Boolean valued data, characters, and trees [3]. In the learning mechanism based on dot-products only, what we need is not the mapping $\boldsymbol{\phi}$ itself, but the kernel function K. This is why kernel functions attract much attention.

Various kernel functions were designed. We have already proposed several kernel functions for applying SVM to structured data[2,13]. All of them are in the class that we named the *intentional kernel* [13]. In this paper we precisely explain the fundamental idea of designing the class intentional kernel.

The intentional kernel is defined with the property that every structured data is defined by derivation. Derivation means transforming a data or an expression into another. Typical derivation can be found in the field of formal language theory: A grammar defines a language in the sense that every sequence is in the language if it is obtained from a starting symbol by repeated application of the grammar. Another example is deduction in mathematical logic: A formula is

S. Chawla et al. (Eds.): PAKDD 2008 Workshops, LNAI 5433, pp. 111–122, 2009.

proved if it is obtained from axioms by repeated application of inference rules. Introducing derivation with the kernel-based learning mechanism derives the class of the intentional kernel.

Combining logical deduction with Machine Learning is investigated in the field of Inductive Logic Programming [9,16]. We can find some research of combination of deduction and kernel functions, and some kernel functions invented in the research are intentional kernel functions. Our aim of introducing the intentional kernel is to give a uniform view to combining deduction and kernel-based learning mechanisms.

The rest of this paper is organized as follows: We propose the definition of the intentional kernel in Section 2. Section 3 shows instances of the intentional kernel, and Section 4 shows an intentional kernel for RNA classification. We describe some complexity analyses for intentional kernel in Section 5, and conclusion in Section 6.

2 Intentional Maps and Kernel Functions

2.1 An Introductory Example

We explain the fundamental idea of intentional kernel with a simple example, where we treat sequences of symbols and design a kernel function by using a context free grammar.

Let us treat sequences consisting of two terminal symbols a and b. When a context free grammar (CFG) G is given, we define a kernel function $K_G(x, y)$ for x and $y \in \{a, b\}^*$ as the number of expressions that occur at least one derivation of x and at least one derivation of y, where an expression means a sequences of symbols and non-terminal symbols in this section.

We give a concrete example. Suppose that $x = aaabbb$, $y = aababb$, and the production rules of G are

$$\begin{aligned} S &\to SS\,, \\ S &\to aSb\,, \\ S &\to ab\,, \end{aligned}$$

where S is the only non-terminal symbol in this grammar.

The kernel value $K_G(aaabbb, aababb)$ is computed by using the derivations of $aaabbb$ and $aababb$ from S. A derivation of $aaabbb$ from S is

$$d_1 : S \vdash aSb \vdash aaSbb \vdash aaabbb,$$

and there is no other one. For $aababb$ there are two derivations from S:

$$\begin{aligned} d_2 &: S \vdash aSb \vdash aSSb \vdash aabSb \vdash aababb, \\ d_3 &: S \vdash aSb \vdash aSSb \vdash aSabb \vdash aababb. \end{aligned}$$

Then expressions in $\{a, b, S\}^*$ occurring in d_1 are

$$S,\ aSb,\ aaSbb,\ aaabbb,$$

while expressions in d_2 or d_3 are

$$S,\ aSb,\ aSSb,\ aabSb,\ aSabb,\ aababb.$$

The expressions common to both of the two lists are S and aSb, and therefore we define

$$K_G(aaabbb, aababb) = 2.$$

In order to represent the kernel function $K_G(x, y)$ with the dot product for the two vectors in $\mathbf{R}^\infty$, we define a mapping $\boldsymbol{\phi}_G : \{a, b\}^* \longrightarrow \mathbf{R}^\infty$. At first we give an injective (one-to-one) function $\iota : \{a, b, S\}^* \longrightarrow \mathbf{N}$ which represents each expression with a "code" of a natural number. With this injection the mapping

$$\boldsymbol{\phi}_G(x) = (\phi_1(x), \phi_2(x), \ldots)$$

is defined as

$$\phi_{\iota(e)}(x) = \begin{cases} 1 & \text{if } e \text{ occurs in a derivation of } x \text{ from } S, \text{ and} \\ 0, & \text{otherwise.} \end{cases}$$

For example, $\phi_{\iota(aSb)}(aaabbb) = 1$, $\phi_{\iota(aSSb)}(aaabbb) = 0$, and we obtain the property that

$$\boldsymbol{\phi}_G(x) \cdot \boldsymbol{\phi}_G(y) = \sum_{e \in \{a,b,S\}^*} \phi_{\iota(e)}(x) \cdot \phi_{\iota(e)}(y).$$

2.2 Formal Definitions

As explained in the previous subsection, an intentional kernel function is designed by using derivation as characterization of structured data. A *derivation* of a structured data is represented in the form of

$$F_0 \vdash F_1 \vdash \cdots \vdash F_k \ni x,$$

where F_i is an expression which represents a set $C(F)$ of data, and $F_i \vdash F_j$ denotes that F_j is directly derived from F_i. In this paper a set of data is called a *concept*. In the case of using a CFG G, an expression of a concept is a finite sequence of terminal symbols and non-terminal symbols, and F_j is directly derived from F_i if $F_i = u\alpha w$ and $F_j = u\beta w$ for some rule $\alpha \to \beta$ in G.

The key idea of designing the intentional kernel is interpreting every expression F as a co-ordinate of $\mathbf{R}^d$ $(d \leq \infty)$. Let U be the set of structured data that we are treating, $\mathcal{F}$ be the set of expressions, and every F in $\mathcal{F}$ represent a subset $C(F)$ of U. In the theory of computational learning, every expression in $\mathcal{F}$ is sometimes called a *hypothesis*. We assume that different expressions represent different subsets, that is, the mapping $C : \mathcal{F} \longrightarrow 2^U$ is an injection. We also assume that every $F \in \mathcal{F}$ represents a co-ordinate $\iota(F)$ of $\mathbf{R}^d$ and every co-ordinate is represented by a unique expression, that is, $\iota : \mathcal{F} \longrightarrow \mathbf{N}$ is a bijection. When a mapping $\boldsymbol{\phi} : U \longrightarrow \mathbf{R}^d$ is represented as

$$\boldsymbol{\phi}(x) = (\phi_1(x), \phi_2(x), \ldots),$$

we write $\phi_F(x)$ instead of $\phi_{\iota(F)}(x)$.

Definition 1. A mapping $\boldsymbol{\phi} : U \longrightarrow \mathbf{R}^d$ is called an *intentional mapping* if $\phi_F(x) > 0 \Longleftrightarrow x \in C(F)$. A kernel function $K(x, y)$ $(x,\ y \in U)$ is called an *intentional kernel function* if it is defined with the set

$$E(x, y) = \{F \in \mathcal{F} \,|\, x \in C(F) \text{ and } y \in C(F)\}.$$

Note that for an intentional mapping, it holds that $\phi_F(x) \cdot \phi_F(y) > 0 \Longleftrightarrow x \in C(F)$ and $y \in C(F)$.

For an intentional mapping $\boldsymbol{\phi}$, it is not guaranteed that the series

$$\boldsymbol{\phi}(x) \cdot \boldsymbol{\phi}(y) = \sum_{F \in \mathcal{F}} \phi_F(x) \cdot \phi_F(y)$$

converges. In order to guarantee the convergence, we need some restriction to $\boldsymbol{\phi}$ or modify the definition of dot product. In designing intentional kernel we define a restriction by using the concept introduced in the theory of computational learning.

In computational learning of formal languages, we say that the class $\mathcal{C} = \{C(F) \,|\, F \in \mathcal{F}\}$ has the finite thickness property if, for every data $x \in U$, there exists only finitely many $C(F) \subset U$ such that $x \in C(F)$. Because we are assuming that C is an injection, the set $\mathcal{F}$ has the *finite thickness property* if, for every data $x \in U$, there exists only finitely many $F \in \mathcal{F}$ such that $x \in C(F)$.

Definition 2. If $\mathcal{F}$ has the finite thickness property, $K(x, y) = \sharp E(x, y)$ is an intentional kernel function, where $\sharp E(x, y)$ denotes the number of elements in $E(x, y)$. The kernel function is called a *counting intentional kernel function*.

It is easy to know that a counting intentional kernel function is defined with an intentional mapping $\boldsymbol{\phi}$ such that

$$\phi_F(x) = \begin{cases} 1 & \text{if } x \in C(F), \text{ and} \\ 0, & \text{otherwise.} \end{cases}$$

Now we introduce derivation into designing kernel functions.

Definition 3. A function $r : \mathcal{F} \longrightarrow 2^{\mathcal{F}}$ is a *derivation rule*[1] if

- $r(F)$ is finite for all $F \in \mathcal{F}$, and
- $G \in r(F) \Longrightarrow C(F) \supseteq C(G)$.

An expression $F \in \mathcal{F}$ is *maximal* if there is no G such that $F \in r(G)$. F is *minimal* w.r.t. a data $x \in U$ if $d \in C(F)$ and $d \notin C(G)$ for any $G \in r(F)$.

We illustrate this semantics in Fig. 1. In the figure, every node of the graph represents an expression in F and every directed edge from a node F to G means that $G \in r(F)$. When $\mathcal{F}$ has finite thickness, the nodes surrounded by the bold line square are used for the counting intentional kernel function.

[1] In ILP r is also called a *refinement* [6,15].

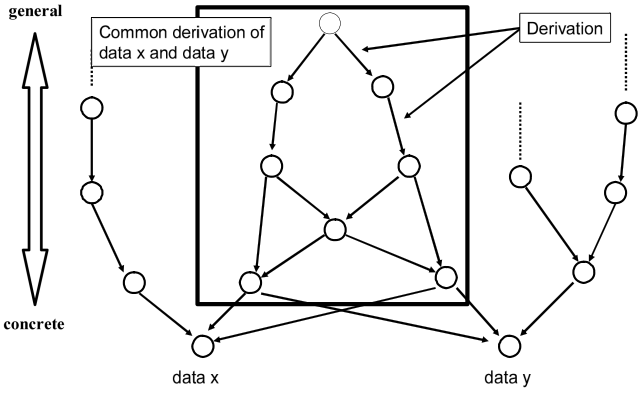


Fig. 1. Derivation and intentional kernel

We give semantics to $E(x, y)$ when $U \subset \mathcal{F}$ and a data $x \in U$ itself is an expression which represents the set $\{x\}$. The example of using a CFG presented in the last subsection is such a case. We define a relation $\succeq'_r$ as

$$x \succeq'_r y \iff y \in r(x)$$

and $\succeq_r$ as the reflexive transitive closure of $\succeq'_r$. Then it holds that

$$E(x, y) = \{z \in U \mid z \succeq_r x \text{ and } z \succeq_r y\}.$$

3 Instances of the Intentional Kernel

In this section we put some kernels proposed in previous works into the class of the intentional kernel.

3.1 Boolean Kernels

Sadohara [11,12] and Khardon et al. [5] independently investigated to apply SVM to learning of Boolean functions. We show that the kernels introduced in their research are intentional kernel functions. We follow the definitions in [11].

Let $B = \{0, 1\}$ and assume that $U = B^n$, and we prepare Boolean variables $x_1, x_2, \ldots, x_n$ so that each variable x_i represents the i-th element of Boolean vectors in B^n. As expressions representing subsets of B^n, we adopt conjunctions of literals in which each variable x_i occurs at most once, and we let $\mathcal{F}$ be the set of such conjunctions. For every conjunction $F \in \mathcal{F}$, we define $C(F) \subset B^n$ as follows:

$\mathbf{x} = (b_1, b_2, \ldots, b_n) \in C(F) \iff$
For every $i = 1, 2, \ldots, n$
$b_i = 1$ if the positive literal x_i occurs in F, and
$b_i = 0$ if the negative literal $\neg x_i$ occurs in F.

For two data $\mathbf{x} = (b_1, b_2, \ldots, b_n)$ and $\mathbf{y} = (c_1, c_2, \ldots, c_n)$ in B^n, we let $s(\mathbf{x}, \mathbf{y})$ denote the number of i such that $b_i = c_i$. Then by using the property that

$$\boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{y}) = \sum_{i=1}^{s(\mathbf{x},\mathbf{y})} \binom{s(\mathbf{x}, \mathbf{y})}{i},$$

we define the kernel function K_{BOOL} as

$$K_{BOOL}(\mathbf{x}, \mathbf{y}) = -1 + 2^{s(\mathbf{x},\mathbf{y})}.$$

This function is called the *DNF kernel* function.

The DNF kernel function is a counting intentional kernel, by defining a derivation rule r as

$$\begin{array}{l} G \in r(F) \Longleftrightarrow \\ G \text{ is a conjunction obtained by adding to } F \\ \text{a literal whose variable does not occur in } F. \end{array}$$

Maximal conjunctions are those consisting of only one literal and the minimal conjunction $F_{\mathbf{x}}$ w.r.t. a data $\mathbf{x} = (b_1, b_2, \ldots, b_n)$ is such ones that

- the positive literal x_i occurs in F if $b_i = 1$, and
- the negative literal $\neg x_i$ occurs in F if $b_i = 0$

for every $i = 1, 2, \ldots, n$. From the definition of $C(F)$, we get

$$\mathbf{x} \cdot \mathbf{y} = \sharp E(\mathbf{x}, \mathbf{y}).$$

Figure 2 illustrates the derivation and computing of K_{DNF} for the case that $\mathbf{x} = (1, 0, 0, 1)$ and $\mathbf{y} = (1, 0, 1, 0)$.

The *monotone DNF kernel* function is defined by using conjunctions of positive literals (Boolean variables). The kernel is defined as

$$K_{mBOOL}(\mathbf{x}, \mathbf{y}) = -1 + 2^{s'(\mathbf{x},\mathbf{y})},$$

where $s'(\mathbf{x}, \mathbf{y})$ represents the number of i such that $b_i = c_i = 1$, for every $\mathbf{x} = (b_1, b_2, \ldots, b_n)$ and $\mathbf{y} = (c_1, c_2, \ldots, c_n)$.

3.2 Kernel Function for Terms in First-Order Logic

We proposed a kernel function for terms in first-order logic as data [2]. Let $\mathcal{T}$ be the set of terms in first-order logic. For the convenience of application and computation we remove singleton variables from $\mathcal{T}$.

By using the least common anti-instances [7,10] of two terms, which is a well-known concept in ILP, we define the kernel function K_{TERM} as

$$K_{TERM}(s, t) = size(lca(s, t)),$$

where $lca(s,t)$ is the least common anti-instances of s and t. The function $size(u)$ denotes the number of such terms that $t = u\theta$ for some substitution θ. This function makes K_{TERM} be a counting intentional kernel function.

We let $U = \mathcal{F} = \mathcal{T}$ with assuming that

$$C(t) = \{s \in U \mid s = t\theta \text{ for some substitution } \theta\}$$

for every $t \in \mathcal{F}$. For a term t let

$$\phi_s(t) = \begin{cases} 1 & \text{if } t = s\theta \text{ for some substitution } \theta, \text{ and} \\ 0, & \text{otherwise.} \end{cases}$$

It has shown that $\mathcal{F}$ has the finite thickness property, and from the property it holds that

$$K_{TERM}(s,t) = \boldsymbol{\phi}(x) \cdot \boldsymbol{\phi}(y).$$

A derivation r for terms is defined as

$$r(t) = \{s \mid s = t\sigma \text{ for some primitive substitution } \sigma \text{ for } t\ \},$$

where a primitive substitution for t is either

- a substitution which replaces x in t with a term $f(x_1, \ldots, x_n)$(f is a function symbol, $x_1, \ldots, x_n$ are mutually distinct variables not occurring in t, $n \geq 0$), or
- a substitution which unifies different variables x and y by replacing y with x.

The maximal term is of the form $f(x_1, \ldots, x_n)$ (f is a function symbol, $x_1, \ldots, x_n$ are distinct variables, $n \geq 0$) and t itself is the minimal term w.r.t. t.

Figure 3 shows the definition of K_{TERM} with the case that $x = f(g(b,a), h(a))$ and $y = f(g(c,a), h(a))$. The terms in the bold line square are used for computing the value of $K_{TERM}(x,y)$.

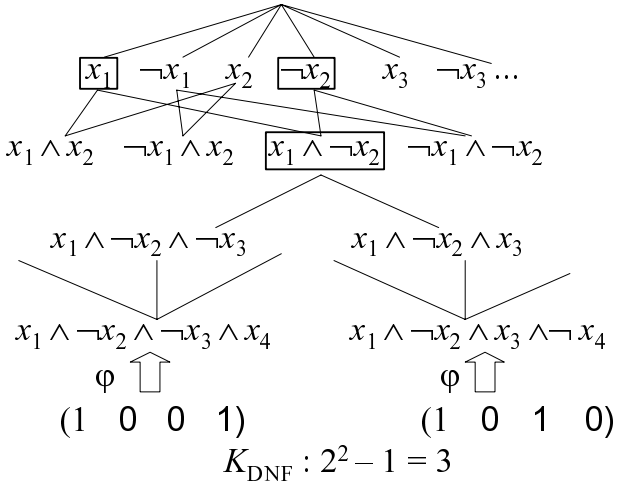


Fig. 2. Derivation of conjunctions of literals and the DNF kernel

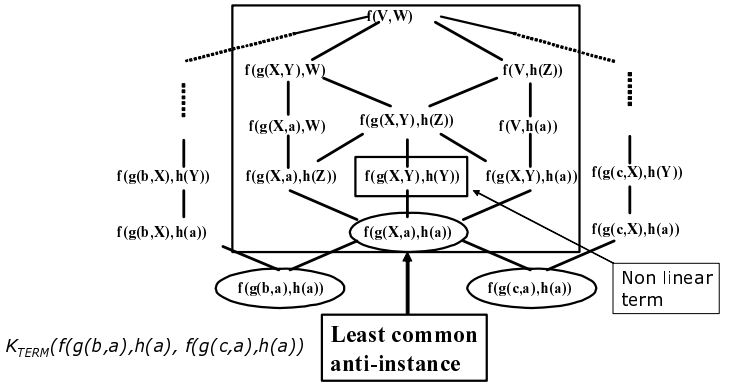


Fig. 3. Counting intentional kernel function K_{TERM} for terms

3.3 Context Sensitive Languages

The simple example in Subsection 2.1 is a counting intentional kernel for a context free languages (CFLs). We can construct the counting intentional kernel for not only CFL but also context sensitive languages (CSLs) based on the same idea.

Let Σ be the set of non-terminal symbols in a grammar G, N be the set of terminal symbols, S be the start symbol. Assume that $U = \mathcal{F} = (\Sigma \cup N)^*$ and

$$C(w) = \{u|\ u \text{ is derived from } w \text{ by } G\}$$

for every w in $\mathcal{F}$. A derivation rule is defined as

$$r(w) = \{u|\ u \text{ is derived from } w \text{ by using a production rule once in } G\}.$$

We consider the restriction for a production rule $\alpha \to \beta$ that β be at least as long as α in derivation rules. The resulting grammar is called a context sensitive grammar (CSG). Each CSL is generated by a CSG. The number of expressions generating a string w is finite because the lengths of the expressions are equal to or less than $|w|$. Therefore we can construct the counting intentional kernel for a CSG.

4 More Complex and Practical Example

We proposed a class K_{RNA} of kernel functions for RNA classification by using a CFG representing secondary structures of RNA sequences [13] as a subclass of the intentional kernel. The function is based on a CFG but the formalization is different from that of the example in Subsection 2.1.

The CFG G_{RNA} for K_{RNA} is defined with terminal symbols a, u, c, and g, and non-terminal symbols P, L, R, S, and E.

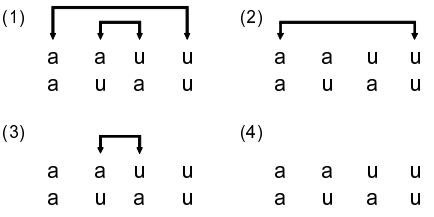


Fig. 4. Common secondary structures of $\sigma_1 = aauu$ and $\sigma_2 = auau$

The production rules are

$$\begin{aligned} &S \to P \mid L \mid R \mid E\,, \\ &P \to xPy \mid xLy \mid xRy \mid xEy\,, \\ &L \to xP \mid xL \mid xR \mid xE\,, \\ &R \to Px \mid Lx \mid Rx \mid Ex\,, \\ &E \to \epsilon\,, \end{aligned}$$

where $x \in \{a, u, c, g\}$ and $(x, y) \in \{(a, u), (u, a), (c, g), (g, c)\}$.

In designing kernel functions in K_{RNA} we let $U = \{a, u, c, g\}^*$. We do not use $\{a, u, c, g, P, L, R, S, E\}^*$ for the set $\mathcal{F}$, because it is not appropriate to representing secondary structures of RNA sequences. Instead we use the set

$$\{e_1 \to e_2 \to \cdots \to e_n \mid n \geq 1 \text{ and } e_i \in \{P, L, R, S, E\}^* \text{ for } i = 1, 2, \ldots, n\}$$

for $\mathcal{F}$. Expressions in $\mathcal{F}$ are similar to derivations of G_{RNA} but lack symbols a, u, c, g in it. For every sequence $\sigma \in \{a, u, c, g\}^*$, $\sigma \in C(F)$ if there is a derivation (in the normal sense) d of σ such that d can be transformed into an expression in F. We do not give the precise definition of the transforming method, but illustrate it as a simple example below. Kernel functions in K_{RNA} are designed just as in the definition in the previous section by using $\mathcal{F}$.

Consider a sequence $\sigma = aauu$. An example of derivation of $aauu$ is

$$d_4 \,:\, S \vdash P \vdash aPu \vdash aaEuu \vdash aauu.$$

Based on this derivation, we define that $aauu$ belongs to $C(S \to P \to P)$, $C(S \to P \to P)$ and $C(S \to P)$. Another derivation of $aauu$ is

$$d_5 \,:\, S \vdash L \vdash aR \vdash aPu \vdash aaEuu \vdash aauu,$$

and this derivation makes $aauu$ belongs to $C(S \to L \to R \to P)$. Each expression in $\mathcal{F}$ corresponds to a secondary structure as illustrated in Fig. 4.

Figure 5 shows the definition of K_{RNA} with the case $K_{RNA}(aauu, auau)$. The expressions in the bold line square are used for computing the value.

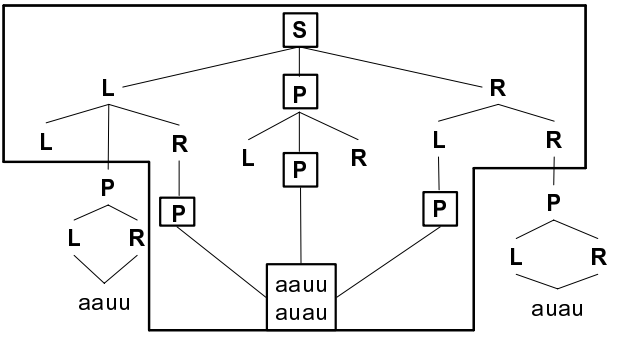


Fig. 5. Common expressions for $\sigma_1 = aauu$ and $\sigma_2 = auau$

5 Computational Complexity

With the two works we also showed some computational aspects of the intentional kernel. The complexity of computing the value of an intentional kernel function K depends on the derivation rules in G with which K is defined. Let $F_i \vdash F_j$ is a fragment of a derivation of x or y by G. If the expression F_j is obtained from F_i by replacing a subexpression of F_i with giving affection to no other subexpressions, just as in the case that G is a context free grammar, we can apply dynamic programming to computing the value of $K(x, y)$.

In our work on K_{RNA}, we showed that dynamic programming makes the computation of $K_{RNA}(x, y)$ efficient, theoretically and practically, with combination of another technique of improving efficiency based on the data structure of sequences. If the derivation rule G is not purely "context free," we need some search mechanism for computing $K(x, y)$. The kernel function K_{TERM} is such a kernel. In computing $K_{TERM}(x, y)$, we divide every common derivation d of

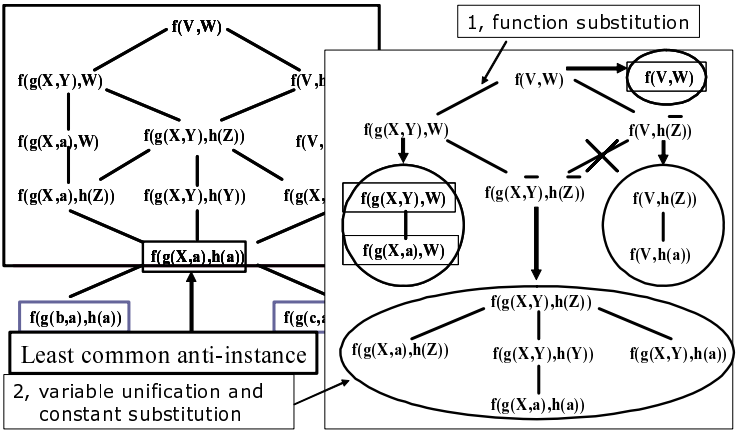


Fig. 6. The computation for K_{TERM}

two terms x and y into two parts d_1 and d_2 so that d_1 is not "context free" and d_2 is "context free." To the part d_2 we apply dynamic programming while we use a depth-first search for d_1. We have proposed an algorithm in [2] (Fig. 6).

6 Concluding Remarks

In this paper we have explained the fundamental idea of designing intentional kernel functions. We have discussed neither their theoretical properties nor experimental results here. In our previous work [2] we provided theoretical properties of the intentional kernel function K_{TERM} for first-order terms. In another previous paper [13] we reported preliminary experimental results of a kernel function in K_{RNA} for RNA sequences, and now preparing more results on them.

The convolution kernel proposed by Haussler [4] is the most popular class of kernels for structured data. The fundamental idea of designing the convolution kernel is that every structured data is composed of small substructures, and that a kernel should be defined with the values for the small substructures. In mathematical expression, the idea is expressed as

$$K(x, y) = \sum_{u \in s(x)} \sum_{v \in s(y)} K^s(u, v),$$

where $s(x)$ and $s(y)$ are respectively the sets of all proper substructures of x and y. Two classes of the convolution kernel and the intention kernel are closely related in the case that the definition of the mapping s is defined with some CFG and dividing a structured data into substructures could be "parsing" with the CFG. The kernel function K_{TERM} would not be regarded as a convolution kernel because the derivation rule for K_{TERM} has unification of two variables and is not context free. If we remove the unification rule, we could not treat terms such as $f(X, X)$ and the value of $K_{TERM}(f(a, a), f(b, b))$ would be 1, which is 2 as an intentional kernel. We are now continuing numerical experiments of the kernel functions, in particular, in K_{RNA} with comparing other kernels. We would like report the results in near future.

Acknowledgment

This research is partially supported by Grant-in-Aid for Scientific Research (B) 19300046 and Young Scientist (B) 20700135 from MEXT and JSPS.

References

1. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods. Cambridge University Press, Cambridge (2000)

2. Doi, K., Yamashita, T., Yamamoto, A.: An Efficient Algorithm for Computing Kernel Function Defined with Anti-unification. In: Muggleton, S., Otero, R., Tamaddoni-Nezhad, A. (eds.) ILP 2006. LNCS (LNAI), vol. 4455, pp. 139–153. Springer, Heidelberg (2007)
3. Gärtner, T.: A Survey of Kernel for Structured Data. SIGKDD Explorations 5(1), 268–275 (2003)
4. Haussler, D.: Convolution Kernels on Discrete Structures, Technical Report UCS-CRL-99-10, University of California - Santa Cruz (1999)
5. Khardon, R., Roth, D., Servedio, R.: Efficiency versus Convergence of Boolean Kernels for On-Line Learning Algorithms. Journal of Artificial Intelligence Research 24, 341–356 (2005)
6. Laird, P.D.: Learning from Good and Bad Data. Kluwer Academic Publishers, Dordrecht (1988)
7. Lassez, J.-L., Maher, M.J., Marriott, K.: Unification Revisited. In: Minker, J. (ed.) Foundations of Deductive Databases and Logic Programming, pp. 587–626. Morgan-Kaufman, San Francisco (1988)
8. Muggleton, S., Lodhi, H., Amini, A., Sternberg, M.J.E.: Support Vector Inductive Logic Programming. In: Hoffmann, A., Motoda, H., Scheffer, T. (eds.) DS 2005. LNCS (LNAI), vol. 3735, pp. 163–175. Springer, Heidelberg (2005)
9. Nienhuys-Cheng, S.-H., de Wolf, R.: Foundations of Inductive Logic Programming. Springer, Heidelberg (1997)
10. Plotkin, G.D.: A Note on Inductive Generalization. Machine Intelligence 5, 153–163 (1970)
11. Sadohara, K.: Learning of Boolean Functions Using Support Vector Machine. In: Abe, N., Khardon, R., Zeugmann, T. (eds.) ALT 2001. LNCS (LNAI), vol. 2225, pp. 106–118. Springer, Heidelberg (2001)
12. Sadohara, K.: On a Capacity Control Using Boolean Kernels for the Learning of Boolean Functions. In: Proceedings of the IEEE International Conference on Data Mining, pp. 410–417 (2002)
13. Sankoh, H., Doi, K., Yamamoto, A.: An Intentional Kernel Function for RNA Classification. In: Corruble, V., Takeda, M., Suzuki, E. (eds.) DS 2007. LNCS (LNAI), vol. 4755, pp. 281–285. Springer, Heidelberg (2007)
14. Scholkopf, B., Smola, A.J.: Learning With Kernels: Support Vector Machines, Regularization, Optimization and Beyond. MIT Press, Cambridge (2001)
15. Shapiro, E.Y.: Inductive Inference of Theories From Facts, Technical Report 192, Department of Computer Science, Yale University (1981); also in: Lassez, J.-L., Plotkin, G. (eds.) Computational Logic, pp. 199–254.The MIT Press, Cambridge (1991)
16. Yamamoto, A.: Inductive Logic Programming: Yet Another Application of Logic. In: Umeda, M., Wolf, A., Bartenstein, O., Geske, U., Seipel, D., Takata, O. (eds.) INAP 2005. LNCS (LNAI), vol. 4369, pp. 102–116. Springer, Heidelberg (2006)

Dynamic Bayesian Networks for Acquisition Pattern Analysis: A Financial-Services Cross-Sell Application

Anita Prinzie[1, 2] and Dirk Van den Poel[2]

[1] Marketing Group, Manchester Business School, The University of Manchester, Booth Street West, Manchester M15 6PB, UK
Anita.Prinzie@mbs.ac.uk
[2] Department of Marketing, Ghent University, Tweekerkenstraat 2, 9000 Ghent, Belgium
{Anita.Prinzie,Dirk.VandenPoel}@UGent.be

Abstract. Sequence analysis has been employed for the analysis of longitudinal consumer behavior with the aim to support marketing decision making. One of the most popular applications involves Acquisition Pattern Analysis exploiting the existence of typical acquisition patterns to predict customer's most likely next purchase. Typically, these cross-sell models are restricted to the prediction of acquisitions for a limited number of products or within product categories. After all, most authors represent the acquisition process by an extensional, unidimensional sequence taking values from a symbolic alphabet. This sequential information is then modeled by (hidden) Markov models suffering from the state-space explosion problem. This paper advocates the use of intensional state representations exploiting structure and consequently allowing to model complex sequential phenomena like acquisition behavior. Dynamic Bayesian Networks (DBNs) represent the state of the environment (e.g. customer) by a set of variables and model the probabilistic dependencies of the variables within and between time steps. The advantages of this intensional state space representation are demonstrated on a cross-sell application for a financial-services provider. The DBN models multidimensional customer behavior as represented by acquisition, product ownership and covariate sequences. In addition to the ability to model structured multidimensional, potentially coupled, sequences, the DBN exhibits adequate predictive performance to support the financial-services provider's cross-sell strategy.

1 Introduction

Sequence analysis has become common-place in the longitudinal analysis of consumer behavior. One of the most popular applications is Acquisition Pattern Analysis, describing the next logical product/service acquisition for a customer, based on the sequential pattern of customer's acquisition history and on the pattern of other customers. Extant research exploits the order in which household units acquire durable goods [Prinzie and Van den Poel 2007] or financial services [Kamakura et al. 1991, Li et al. 2005, Paas et al. 2007, Prinzie and Van den Poel 2006] to support cross-sell strategies. Typically, the customer's longitudinal acquisition sequence is represented as an unstructured, unidimensional sequence, thereby limiting the practical value of any cross-sell model inferred from it in multiple ways.

S. Chawla et al. (Eds.): PAKDD 2008 Workshops, LNAI 5433, pp. 123–133, 2009.

Firstly, the unidimensional representation impedes capturing the acquisition behavior at a sufficient level of detail or for the full product range. For example, in an *l*th-order Markov model the next acquisition is described by *l*-previous values of one random variable X_t , taking values from a symbolic alphabet N={1, …, M}. This extensional representation of the customer's acquisition state, one in which each state is explicitly named rather than described by variables [Boutilier, Dean and Hanks 1999], rapidly results in an explosion of the state space and as a consequence computational intractability of the methods modelling this information. In practice, the state-space explosion problem forces the researcher either to select a limited set of products or to analyse the acquisition behavior at less detailed level, e.g. product categories. In both scenarios, the cross-sell predictive performance and practical value are limited. In the first scenario, the customer might acquire a product or service *not* included in the acquisition pattern analysis. In the last scenario, marketing communication at the product category level might lack specificity and consequently effectiveness.

Secondly, the analysis of acquisition behavior as a unidimensional process largely neglects that the longitudinal acquisition behavior might be related to other longitudinal behavior like portfolio evolution and other covariates changing over time. Latent Markov analysis has been employed [Paas et al. 2007] to link the acquisition process to covariates changing over time. However, a latent Markov model assumes that, when controlling for covariate values at time *t*, the latent class membership only depends on the previous class membership at time *t-1*. Unlike Dynamic Bayesian Networks (DBNs), a latent Markov model does not allow to model coupled processes like the simultaneous evolution of the acquisition sequence with the evolution of one or more covariates also exhibiting a Markov property.

Thirdly, the adoption of an extensional rather than factored or intensional state space representation largely ignores the structure exhibited by the product/service space and typically results in a simplified representation of the decision environment. However, most marketing problems, including cross-sell problems, exhibit considerable structure and thus can be solved using special-purpose methods that recognize that structure [Boutilier, Dean and Hanks]. Amongst other techniques, Dynamic Bayesian Networks (DBNs) could be employed to exploit the structure of the state. DBNs generalize (hidden) Markov models by allowing states to have internal structure. DBNs represent the state of the environment (e.g. customer) by a set of variables; i.e. intensional state representation as opposed to (hidden) Markov's extensional state representation. The DBN models the probabilistic dependencies of the variables within and between time steps. If the dependency structure is sufficiently sparse, it is possible to analyse real-life problems with much larger state spaces than using Markov models. In addition to reducing computational complexity while maintaining the decision problem's complexity, DBN's intensional state-space representation enables the marketing manager to gain insight in the structure of the problem, in case the customer's acquisition process.

This paper illustrates the advantages of Dynamic Bayesian Networks (DBNs) for acquisition pattern analysis with the aim to support the cross-sell strategy of a financial-services provider. The DBN models multidimensional customer behavior as represented by acquisition, product ownership and covariate sequences. The results convey that, in addition to the ability to model structured multidimensional, potentially coupled,

sequences, the DBN exhibits adequate predictive performance to support the financial-services provider's cross-sell strategy.

The remainder of the paper is structured as follows. In the Methodology Section, we briefly present the Dynamic Bayesian Networks. In Section 3, we describe the cross-sell application demonstrating the advantage of DBNs for acquisition pattern analysis. Section 4 discusses the main findings. Finally, the last section draws conclusions and suggests avenues for further research.

2 Methodology

2.1 Dynamic Bayesian Networks as an Extension of Bayesian Networks

A Bayesian Network encodes the joint probability distribution of a set of variables, $\{Z_1, \ldots, Z_d\}$ as a directed acyclic graph expressing conditional dependencies and a set of conditional probability models. Each node corresponds to a variable, and the model computes the probability of a state of the variable given the state of its parents. The set of parents of Z_i, denoted by $Pa(Z_i)$, is the set of nodes with an arc to Z_i in the graph. The structure of the network encodes that each node is conditionally independent of its non-descendants given its parents. The probability of an arbitrary event $Z=(Z_1, \ldots, Z_d)$ is computed as $P(Z) = \prod_{i=1}^{d} P(Z_i | Pa(Z_i))$.

Dynamic Bayesian Networks (DBNs) [Dean and Kanazawa 1989] extend Bayesian Networks for modelling *dynamic* systems thereby also exploiting conditional independence. In a DBN, a state at time *t* is represented by a set of random variables $Z_t=(Z_{i,t}, \ldots, Z_{d,t})$. In a two-time slice Bayesian Network the state at time *t+1*, Z_{t+1} is only dependent on the immediately preceding state Z_t, i.e. $P(Z_{t+1}|Z_t)$ or first-order Markov property. Typically, the transition models are assumed to be invariant across time slices, i.e. a stationary process. A DBN is a pair of Bayesian networks $(B_0, B_{\rightarrow})$ where B_0 represents the initial distribution $P(Z_0)$ and $B_{\rightarrow}$ is a two-time slice Bayesian Network (2TBN) defining the transition distribution. Studying these initial and transition distributions as embodied by the respective Conditional Probability Tables (CPTs) enable the manager to gain insight into the within and between time-slice dependencies. The set of Z_t could be divided into unobserved state variables X_t and observed state variables Y_t. The joint distribution represented by a DBN is obtained by unrolling the 2TBN (1):

$$P(Z_0,\ldots,Z_T) = P(Z_0)P(Z_0|Pa(Z_0))\prod_{t=1}^{T} P(Z_t|Z_{t-1})P(Z_t|Pa(Z_t)) \quad (1)$$

2.2 Predictive Model Evaluation: Class-Specific PCCs and wPCC

The predictive performance of the Dynamic Bayesian Network is evaluated in terms of class-specific Percentage Correctly Classified (PCCs) and the overall wPCC on a separate validation and test set, i.e. data sets of instances not used for model estimation.

In absence of a specific predictive objective, e.g. predict classes k=1 and k=3 well, we evaluate the DBN in terms of its' ability to correctly classify cases in all classes *K*. Given this objective and the class imbalance of the dependent, it is inappropriate

[Barandela et al. 2003] to express the classification performance in terms of the average accuracy like the Percentage Correctly Classified (PCC) [Morrison 1969]. The predictive evaluation of the models should take the distribution of the multinomial dependent into consideration [Morrison]. Firstly, we will weigh the class-specific PCCs with regard to the prior class distribution. Each class k ($k \in K$) of the dependent has a strict positive weight w_k (2), with f_k referring to the relative frequency of the class on the dependent variable. The class-specific weights sum to one as in (2). Given the weights, the weighted PCC is (3):

$$\boldsymbol{w_k} = \frac{1-\boldsymbol{f_k}}{\sum_{\boldsymbol{k}=1}^{\boldsymbol{K}} 1-\boldsymbol{f_k}} \qquad \text{s.t.} \qquad \sum_{\boldsymbol{k}=1}^{\boldsymbol{K}} \boldsymbol{w_k} = 1 \tag{2}$$

$$wPCC = \sum_{k=1}^{K} wPCC_k \tag{3}$$

with $\boldsymbol{wPCC}_k = \boldsymbol{w}_k * \boldsymbol{PCC}_k$. The weighted PCC is related to the balanced error rate. We penalize models predicting several alternatives by equally dividing the 100% classified over all alternatives predicted. Secondly, we benchmark the model's performance to the proportional chance criterion Cr_{pro} rather than the maximum chance criterion Cr_{max} [Morrison].

3 A Financial-Services Cross-Sell Application

The benefits of Dynamic Bayesian Networks (DBNs) for acquisition pattern analysis are illustrated in a financial-services cross-sell application. From a data warehouse of an international financial-services provider a household's acquisition sequence in eleven service categories (Table 1) is derived. Notice that the acquisition sequences are constructed at the household level, as household units are the principal decision-making unit in the financial-services market [Guiso et al. 2002]. The objective is to extract patterns from the acquisition sequences enabling to predict for each household the next financial service acquisition. However, unlike most previous research in Acquisition Pattern Analysis, the longitudinal acquisition behavior is augmented with other longitudinal behavior like the household's service portfolio sequence at each acquisition event. Furthermore, we address the effect of the age of the head of the family on the household's service portfolio at a given acquisition moment. Table 2 defines the intensional state space representation. The ownership state variables clearly reflect the structure of financial services, which can be partitioned into investments, credits, checking accounts and insurances. The latter reflects how DBNs exploit structure intrinsic to the environment. The age state variable is a proxy for the family-life cycle. As such, the age state variable has been discretized to reflect the notion that households residing in different stages of the family-life cycle typically hold different service portfolios due to household need evolution [Kamakura et al.]. Figure 1 shows the Dynamic Bayesian Network modeling 1) the within-time slice effect of age on a household's service portfolio and 2) the between-time slice effects

Table 1. Financial-Services Product Groups

Services Group	Description
1	Investment: low risk, fixed short term (<=10 years)
2	Investment: limited revenue risks and no capital risks, no duration
3	Investment: limited revenue risks and no capital risks, fixed long duration (>10 years)
4	Investment: some revenue risks and no capital risks
5	Investment: no revenue risks, some to high capital risks, no duration
6	Fire insurance
7	Car insurance
8	Other types of insurance (e.g. health, household, accident and life insurance policies)
9	Short-term credit
10	Mortgage
11	Checking account

Table 2. State Space Definition

Acquisition (Acq)	11	Acquisition of one or more services from service category one to eleven.
Ownership Investment (Own_Inv) Ownership Credit (Own_Loan) Ownership Checking Account (Own_CA) Ownership Insurances (Own_Insur)	4	1: no ownership 2: one or two services owned 3: three or four services owned 4: five or more services owned
Age (Age)	3	1: younger than 35 2: from 35 to 54 years old 3: 55 years old or older

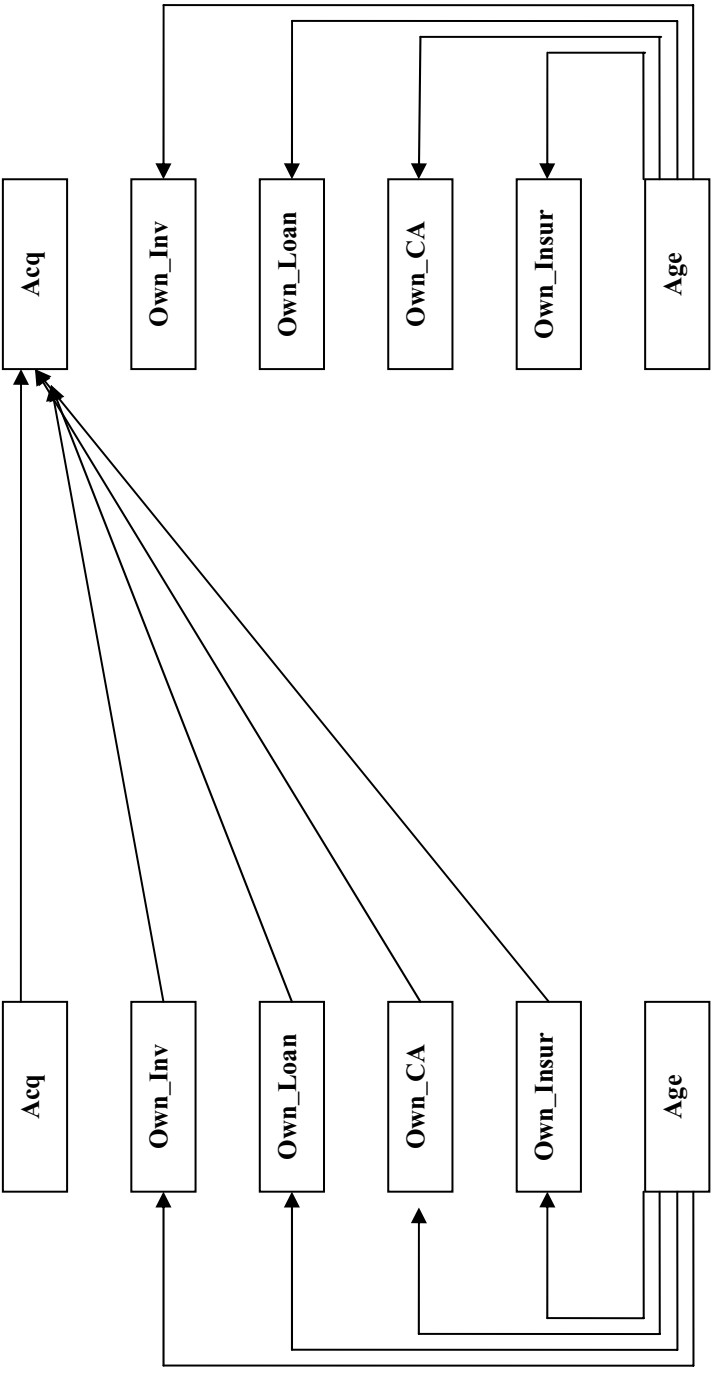


Fig. 1. Dynamic Bayesian Network Modeling Longitudinal Acquisition Behavior for Financial Services

of service portfolio and service acquisition on the household's next service acquisition. The reader should keep in mind that the network structure shown in Figure 1 is only one of the many networks possible. For example, the network could be extended with a first-order Markov effect between service portfolio. Furthermore, extending the 2TBN to a higher-order Dynamic Bayesian Network could control for higher-order acquisition effects. Finally, in the application at hand, the acquisition state variable adopts the services partition as used by the financial-services provider. However, one could decompose the acquisition state variable with as many state variables as there are relevant service features to predict what features the next most likely acquired service would have.

From the original database of the financial-services provider containing information on approximately 860,000 households, households are selected with at least two acquisition dates. Households with an exceptional high number of services acquired or with too many missing values on the service category are deleted. After data preparation, 600,340 households are retained. We randomly assigned 200,113 households to the estimation sample, 200,114 households to the validation sample and the remaining 200,113 households to a test sample.

4 Results

4.1 Predictive Performance

We estimated the DBN illustrated in Figure 1 using the full-length acquisition sequences of the households in the estimation sample. The predictive performance of the DBN indicates how well it is able to predict for all households in a specific sample the 2-nd until last acquisition event. The robustness of the predictive performance of the DBN is assessed by applying the DBN on the validation and test sample. Table 3 reports the predictive performance on the estimation, validation and test samples with respect to the wPCC and the service category-specific PCCs. The results show that the DBN is fairly robust as reflected by similar predictive performance measures across the estimation, validation and test samples. The DBN has a weighted PCC of 34.91% on the test sample, indicating that when correcting for the prior distribution of the financial-services groups, the DBN allows to correctly classify almost 35% of the next acquisitions. The wPCC on all three samples largely outperforms the proportional chance criterion Cr_{pro} of 0.18. The service category specific PCCs are independent of prior class probabilities. The DBN has a high hit rate for car insurances (7), investment products with low risk and fixed short term (1) and for investments with limited revenue risks, without capital risks nor duration (2). The DBN predicts at least 30% of the acquisitions in the other product groups with the exception of other type of insurances (8) and checking accounts (11). The off-diagonal cells of the confusion table in Table 4 provide insight in the pattern of misclassifications. The last row reports the percentage difference between the percentage predicted and the actual percentage of acquisitions in a given service category. For instance, the DBN predicts too many car insurance acquisitions (+14.59). All in all, given DBN's adequate predictive performance, the DBN could be implemented by the financial-services provider to support the cross-selling strategies for all services except for other type of insurances and checking accounts.

Table 3. DBN's Predictive Performance

	wPCC	PCC1	PCC2	PCC3	PCC4	PCC5	PCC6	PCC7	PCC8	PCC9	PCC10	PCC11
Estimation	35.15	73.03	56.40	34.43	33.88	31.28	33.53	75.08	0.93	37.71	19.76	2.74
Validation	34.64	73.44	55.76	34.32	34.30	29.67	33.23	74.91	0.73	34.93	20.04	1.99
Test	34.91	73.00	55.70	34.99	34.48	31.72	33.04	74.90	0.67	36.06	19.46	2.02

Table 4. Confusion Matrix for Test Sample

		Predicted											
Actual		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	
	1	19814	5078.7	49.82	741.07	395.82	328.65	583.65	14.82	105.90	11.07	17.82	27141.31
	2	3839.80	26233	540.96	2573.90	844.63	3630.70	5303.10	155.15	2871.30	661.80	439.93	47094.27
	3	62.27	1056.10	6307.10	669.23	153.44	2845.80	6461.70	38.56	334.98	57.81	39.06	18026.05
	4	1184.80	4879.30	705.60	6476.70	833.17	1328.60	2878.80	41.12	330.95	49.72	75.27	18784.05
	5	494.30	1739	236.64	791.14	2137	353	737.20	17.67	165.54	27.84	37.67	6737
	6	342.61	5139.30	2487.10	1084.90	226.94	34483	57221	311.28	2215.80	608.07	260.05	104380.05
	7	426.47	5114.60	3540.70	1464.40	293.97	25638	121610	118.14	3629.20	258.79	270.82	162365.09
	8	124.82	1795.90	1353.10	357.27	76.99	10708	23979	268.65	843	197.93	120.08	39824.74
	9	139.29	5333.20	494.29	368.69	131.19	3822.30	7506.20	108.96	10847	935.76	389.84	30076.72
	10	37.59	1650.40	262.59	194.35	60.67	2460.80	3019.20	46.85	1377.10	2229.70	115.77	11455.03
	11	670.80	12494	378.05	664.55	221.72	3061.90	4921	98.79	2920.70	483.37	533.14	26448.22
		27136.75	70513.5	16355.95	15386.2	5375.54	88660.75	234220.85	1219.98	25641.67	5521.85	2299.44	492332.48
Predicted %		5.51	14.32	3.32	3.13	1.09	18.01	47.57	0.25	5.21	1.12	0.47	
Actual %		5.51	9.57	3.66	3.82	1.37	21.20	32.98	8.09	6.11	2.33	5.37	
Difference		0.00	4.76	-0.34	-0.69	-0.28	-3.19	14.59	-7.84	-0.90	-1.21	-4.90	

Table 5. Extract from the DBN's Conditional Transition Probabilities

10	2	2	2	1	0.0227	*0.2500*	0.0227	0.0227	0.0455	0.1136	0.0455	0.0455	0.1136	0.1818	0.1364
10	2	2	2	2	0.0000	0.0714	0.0000	0.0238	0.0000	*0.5000*	0.0714	0.0952	0.0476	0.0952	0.0952

Table 6. Inflow into Category 6; Fire Insurances (Transition Probabilities of At Least 0.70)

6	1	4	1	2	0.7453
6	3	4	1	2	0.7000
10	1	4	3	2	0.7692
10	2	4	3	2	0.7273
6	3	4	3	2	0.7143
6	1	4	1	3	0.7447
6	4	4	1	3	0.8000
10	4	2	4	3	0.7500
2	3	4	4	3	0.7500
8	4	4	1	4	0.8000
6	1	4	2	4	0.7000
10	4	2	3	4	0.7500

4.2 Managerial Insights

From a managerial point of view, it is vital to gain insight into the longitudinal acquisition process and its' influential factors. Inspecting the Conditional Probability Tables (CPTs) of a Dynamic Bayesian Network analysis enables this.

The DBN's initial conditional probabilities indicate the effect of state variables *within* a time slice. In the application at hand, the six initial state probability distributions document respectively 1) the initial distribution of acquisitions in the eleven service categories, 2) the initial ownership of investments given the household head's age group, 3) the initial ownership of credits given the household head's age group, 4) the initial ownership of checking accounts given the household head's age group, 5) the initial ownership of insurances given the household head's age group and 6) the initial distribution of households over the three age groups defined. Inspecting these initial state probability distributions reveals that ownership of investment services substantially increases with age. Almost 30% of all households with a household head being 55 years old or older hold at least five investment products in their portfolio, as compared to only 8.65 for households in age group 2 ([35,55[). Furthermore, these older households tend to have fewer insurance policies. The latter might be explained by the investments/cash being the older household's insurance. Finally, the ownership of credits decreases with age.

The DBN's conditional probabilities reflect the effect of state variables between time slices and as such to learn the systems dynamics. In the application at hand, there is one conditional probability table describing the effect of the household's previous service portfolio at time *t*, as expressed by the four service ownership state variables, and the household's previous service acquisition at time *t* on the next acquisition at time *t+1*. Managers and analysts can use these inter-time slice probability distribution(s) to interpret realistic settings. In Table 5 we show an excerpt from the large transition table. The setting "10 2 2 2 1" describes a typical household who, during the previous purchase occasion took out a mortgage. The household's service portfolio at the pervious purchase occasion includes one or two investment products, loan products, checking account but no insurance policies. We observe that this profile of households has the highest probability of acquiring next an investment product (second column: conditional probability of 0.25), followed by another mortgage (0.18). These probabilities differ substantially from the second set "10 2 2 2 2", which represent the transitions for a household owning one or two investment products, loan products, checking account *and* insurance(s). This second household type has a very high probability of acquiring a "fire insurance" policy.

The conditional transition probabilities table also allows managers to analyze the inflow into a particular service category. Let us consider the inflow into category 3; investment services with limited revenue risks but no capital risks for fixed long duration (>10 years). All transitions originate from category 3. A similar analysis is presented in Table 6 for service category 6; fire insurance. In this case, popular transitions into this category not only originate from category 6, but also from 2, 4, 8, and 10. Especially, the transition from mortgage acquisition to fire insurance subscription is very popular, which seems quite logical.

5 Conclusion

One of the major short comings of past research on Acquisition Pattern Analysis is the extensional, one-dimensional sequential representation of the customer's acquisition behavior. This typically results in a simplification of the acquisition process and consequently reduces the practical value of any cross-sell model inferred from it. This paper advocated the use of intensional state representations exploiting structure and consequently allowing to model longitudinal acquisition behavior in its full complexity. The advantages of this intensional state space representation have been demonstrated on a cross-sell application for a financial-services provider. A Dynamic Bayesian Network was developed modeling multidimensional customer behavior as represented by acquisition, product ownership and covariate sequences. The DBN exhibited adequate predictive performance to support the financial-services provider's cross-sell strategy. Furthermore, it has been illustrated how DBN's intensional state-space representation enables the marketing manager to gain insight in the customer's acquisition process by analysing the conditional probability tables. The current application adopted a 2TBN to illustrate the value of DBNs for acquisition pattern analysis. Future research should compare the 2TBN with higher-order DBNs to assess the presence of higher-order acquisition dependencies.

References

1. Barandela, R., Sánchez, J.S., Garcia, V., Rangel, E.: Strategies for learning in class imbalance problems. Pattern Recognition 36(3), 849–851 (2003)
2. Boutilier, C., Dean, T., Hanks, S.: Decision-Theoretic Planning: Structural Assumptions and Computational Leverage. Journal of Artificial Intelligence Research 1, 1–93 (1999)
3. Dean, T., Kanazawa, K.: A model for reasoning about persistence and causation. Computational Intelligence 5(3), 142–150 (1989)
4. Guiso, L., Haliossos, M., Jappelli, T.: Household Portfolios. MIT Press, Cambridge (2002)
5. Kamakura, W.A., Ramaswami, S.N., Srivastava, R.K.: Applying latent trait analysis in the evaluation of prospects for cross-selling of financial services. International Journal of Research in Marketing 8(4), 329–350 (1991)
6. Li, S.B., Sun, B.H., Wilcox, R.T.: Cross-selling sequentially ordered products: an application to consumer banking services. Journal of Marketing Research 42(2), 233–239 (2005)
7. Morrison, D.G.: On the interpretation of discriminant analysis. Journal of Marketing Research 6, 156–163 (1969)
8. Paas, L.J., Vermunt, J.K., Bijmolt, T.H.A.: Discrete time, discrete state latent Markov modelling for assessing and predicint household acquisition of financial products. J. R. Statist. Soc. A 170(4), 955–974 (2007)
9. Prinzie, A., Van den Poel, D.: Investigating purchasing sequence patterns for financial services using Markov, MTD and MTDg models. European Journal of Operational Research 170(3), 710–734 (2006)
10. Prinzie, A., Van den Poel, D.: Incorporating sequential information into traditional classification models by using an element/position-sensitive SAM. Decision Support Systems 42(2), 508–526 (2006)
11. Prinzie, A., Van den Poel, D.: Predicting home-appliance acquisition sequences: Markov/ Markov for Discrimination and survival analysis for modelling sequential information in NPTB models. Decision Support Systems 44(1), 28–45 (2007)

An Automata Based Authorship Identification System

Tsau Young Lin and Shangxuan Zhang

Department of Computer Science, San Jose State University
San Jose, California 95192, USA
z_shangxuan@hotmail.com, tylin@cs.sjsu.edu

Abstract. This paper uses the learning capability of finite automata to develop an authorship identification system. Based on ALERGIA algorithm, we use writing samples of an author to build a stochastic finite automaton. This automaton represents the writing characteristics of the author. This automaton, then, can be used to test whether an anonymous writing piece belongs to this author. Initial tests are quite successful.

1 Introduction

Based on the Kolmogorv complexity K(x) for binary string x, in 1993, Lin proposed to use the opposite of random-ness as the concept of patterns [3], namely, a sequence x has pattern if K(x) < length(x). Obviously, one can conclude that a sequence is said to have pattern if and only if there exists a constant subsequence (Lin stated for infinite sequences; note that finite sequences are automatically included). This could be viewed as the foundation of frequent itemsets (high frequency patterns). In [4], Lin ported the idea to numerical world. In [2], the idea was ported to the world of finite automata, in which the automata were used to detect (learning the patterns) the sequences of system calls in program. Here we switch the applications from the intrusion detection system to authorship identification system, in latter one we use an automaton to detect the string of stop words in a book.

It is well-recognized that every author has some particular writing style; it often can be captured by some statistic characteristics and their hidden relations between the context: average word length, average sentence length in words, word frequency, etc. and their hidden Markov model that reveals the relations among them. Given an anonymous writing piece and possible authors samples, one can investigate these writing characteristics and identify the author of this piece[1].

The aim of this paper is to study authorship identification through function words based on the theory of automata. Function words have long ago been used to identify the writing style. Recently, some interesting work has been done along this direction.

This project apply a similar idea of the work of P.Baliga and T.Y.Lin on the intrusion detection system [2], to text processing. More precisely, writing samples of a prescribed author, instead of programs, are examined by automata. From

S. Chawla et al. (Eds.): PAKDD 2008 Workshops, LNAI 5433, pp. 134–142, 2009.

each sample, we keep the function words for each sentence and wipe out all other information. These sequences of function words are actually the realization of a hidden automaton. Our goal is to use this data and machine learning technique to figure out this automata, which is our representation of the normal writing pattern of the author.

For any other writing sample, our program will test the structure of function words sentence by sentence. We record the proportion of sentences which pass the test. The higher the proportion, the more likely this sample belongs to the author. It is recommended to combine this result with other classical methods of authorship identification to get a more accurate result.

The content of this paper is organized as follows. In section 1, we review stochastic finite automata. In section 2, we describe the ALERGIA algorithm which is used to build an automaton from sample data. In section 3, we handle the data of writing samples, and describe the application of the algorithm to our specific problem. Finally in section 4, we present partial results of the running of our program. S. Zhang would like to thank her advisor T.Y.Lin for his helpful discussion during the research on this topic.

2 Stochastic Finite Automata

In this section we shall review the notion of finite automata and its variation stochastic finite automata [5,6]. In this paper, we shall limit ourselves to deterministic automata.

A deterministic finite automaton(DFA) is a five-tuple $(Q, \mathcal{A}, \delta, q_0, F)$, where $Q = \{q_0, q_1, \cdots, q_n\}$ is its set of states, $\mathcal{A}$ its input symbols, δ its transition function that takes a state and an input symbol as arguments and return a state, q_0 its start state, and F its set of accepting states. One simplest nontrivial DFA is an on/off switch. This device has two states: "on" and "off". The user can press the button to switch one state to another state. For general purpose, one can assign "off" as start state and "on" as accepting state. In reality, a lot of phenomena are actually random. It motivates the following generalization of deterministic finite automata to stochastic finite automata.

A stochastic finite automata(SFA) consists of a DFA $(Q, \mathcal{A}, \delta, q_0, F)$ and a set P of probability matrices $p_{ij}(a)$ for each symbol $a \in \mathcal{A}$. Each $p_{ij}(a)$ gives the probability of the a transition from the state q_i to state q_j led by the symbol a. We let p_{if} be the probability that the string end at state q_i. Then we have the following constraint:

$$p_{if} + \sum_{q_j \in Q} \sum_{a \in \mathcal{A}} p_{ij}(a) = 1.$$

Intuitively, it means that for each state q_i, the sum of the probabilities end at q_i and the probabilities start at q_i should equal to one. Let $\mathcal{A}^*$ be the set of all strings on $\mathcal{A}$. For each string w, one can define the probability $p(w)$ inductively as usual. The language generated by the automaton is defined as:

$$L = \{w \in \mathcal{A}^* : p(w) \neq 0\}.$$

A stochastic regular language(SRL) is defined to be the language generated by an SFA. Two SRLs are said to be equivalent if they contain the same set of strings with the same corresponding probabilities, that is,

$$L_1 \equiv L_2 \Leftrightarrow p_1(w) = p_2(w), \forall w \in \mathcal{A}^*,$$

where L_1 and L_2 are two SRLs, and $p_i(w)$ is the probability of the transition led by w in language L_i.

3 ALERGIA Algorithm

In this section we recall the ALERGIA algorithm to deal with the following problem: Given a fixed SFA, we have a SRL defined by this SFA. Now suppose we are not informed the structure of this SFA, instead we know a large random subset of strings in the SRL generated by this SFA. The goal is to reconstruct the SFA from this given set of strings. For details of the method in this section, please c.f. [6].

Now we describe the approach to solve this problem. First of all, we build a tree from these data. This tree is call a prefix tree adapter(PTA). Each node of the PTA represents a state. For each node of the tree, we assign the frequency of transition led by each symbol. Next, we compare each node of the PTA. We shall define the equivalence of nodes. According to this equivalence, we classify the nodes and merge the equivalent nodes of the PTA. At the end, we recalculate the frequencies and get a SFA which is an approximation of the original SFA.

We start with the definition of PTA. Now suppose the set of sample data is $S = \{s_1, s_2, \cdots, s_m\}$. We describe the PTA inductively. For each string $s_k = a_1 a_2 \cdots a_k$, we begin with the initial node q_0. Suppose there is a transition from q_0 to one of its child node q_i led by a_1, we follow this transition and move to the node q_i. Otherwise, we add a new node to this tree, the transition from q_0 to this new node is thus led by a_1. Either way, we move to a new node, now we look at symbol a_2 and continue this process. In the end, we reach a node that accepting this string. One example of this procedure is given in the next section.

When we run through all the sample data, we can assign the frequency of appearance of each symbol as a transition between nodes, and the number of strings entering each node, the number of string accepting by each node. We denote by n_i the number of strings arriving at node q_i, $f_i(a)$ the number of strings following transition $\delta_i(a)$ and $f_i(\sharp)$ the number of strings ending at node q_i. Obviously, $f_i(a)/n_i$ and $f_i(\sharp)/n_i$ gives estimate of the probabilities $p_i(a)$ and p_{if} respectively.

After we obtain the PTA, we introduce the notion of equivalence between two nodes. Two nodes are said to be equivalent if for all symbols $'a'$ belongs to A, "he associated transition probabilities from the nodes are equal; the termination probabilities for the nodes are equal; and the destination nodes of the two transitions for each symbol are equivalent according to a recursive application of the same criteria." In symbols, we have

$$q_i \equiv q_j \Rightarrow \forall a \in A, \textit{we have } p_i(a) = p_j(a) \textit{ and } \delta_i(a) \equiv \delta_j(a).$$

In the application of this notion, since we seldom have two equal frequencies by statistic fluctuations in experimental results, the equivalence of two nodes must also be accepted within a confidence range. To this end, we call two nodes are compatible if they are equivalent within some pre-described confidence range.

Since for a Bernoulli variable with probability p and frequency f out of n tries, the confidence range is given by the Hoeffding bound as follows:

$$\left|p - \frac{f}{n}\right| < \sqrt{\frac{1}{2n}\log\frac{2}{\alpha}} \textit{ with probability larger than } (1-a).$$

When the two estimated probabilities differ more than the sum of the confidence ranges, the ALERGIA algorithm will reject equivalence.

$$\left|\frac{f}{n} - \frac{f'}{n'}\right| > \sqrt{\frac{1}{2}\log\frac{2}{\alpha}}\left(\frac{1}{\sqrt{n}} + \frac{1}{\sqrt{n'}}\right).$$

Finally, when two nodes are merged, we should recalculate their frequencies and node numbers in order to ensure that the SFA remains deterministic and order-preserving.

4 Automata Based Modeling

In this section we shall describe how to model the authorship identification problem using automata. Our authorship identification approach utilizes function words based automata modeling. In this approach, the first step is to choose an author and collect as many writing samples as possible for use as training data sets that are representative of standard writing style for this author. In the sequel, we shall use the following paragraph as writing sample to illustrate the idea. This piece is cited from the beginning of *Harry Potter And The Prisoner Of Azkaban*.

"Harry Potter was a highly unusual boy in many ways. For one thing, he hated the summer holidays more than any other time of year. For another, he really wanted to do his homework but was forced to do it in secret, in the dead of night. And he also happened to be a wizard." After choosing the sample, we fix the basic unit of training data, which can be one sentence, one paragraph or one whole article, then cut all writing samples into the predetermined units. In this paper, we use one sentence as a unit. The result is finer if the unit is made bigger. However, the running time is longer if we choose larger unit and we need more sample data to keep the number of units large enough to use the ALERGIA algorithms effectively.

In our example, we have four sentences. So we get four units in the sample data. For each unit in the sample, we keep the function words and remove all the other content words. This can be done by choosing a predetermined function words list. We compare each word in the unit according and if the word match a word in the list, we keep it. Applying this to the example, we obtain the following four sequences:

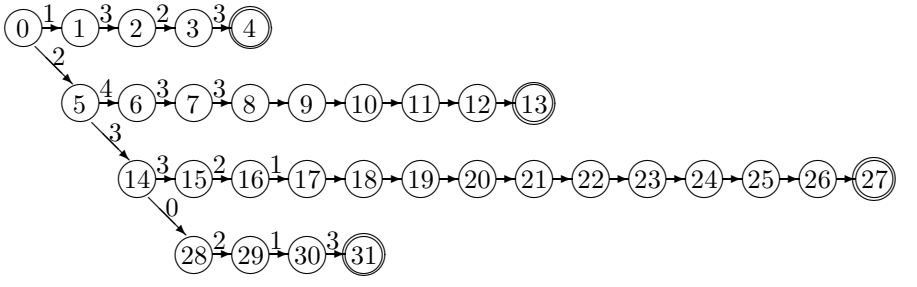
was a in many
for one he the more than any other of
for another he to do his but was to do it in in the of
and he also to be a

Now since the number of function words is around several hundred, to build a tractable automaton, this number is still large as the alphabet of an automaton. The next step is to replace each function word with its part of speech. Usually, we have the following classes of function words: adverb, auxiliary verb, pronoun, preposition, conjunction, interjection and number.

In the following, we use the digits $0, 1, 2, 3, 4$ to represent adverb, auxiliary verb, preposition/conjunction, pronoun and number respectively. This way, we greatly simplify the data of each unit into a sequence of numbers. As an example, we obtain the following sequence of digits.

1 3 2 3
2 4 3 3 3 2 3 3 2
2 3 3 2 1 3 2 1 2 1 3 2 2 3 2
2 3 0 2 1 3

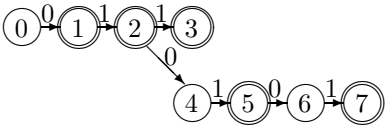
Now from this data we follow the method described in the previous section, we can build the following PTA.



One can calculate the frequency for the transition from each node to it children. Let's take a look at node 5 in our example. We have totally four strings in sample data, out of which the last 3 strings arriving node 5. By our notation in section 2, we have $n_5 = 3$, where the subscript 5 represents node 5. Notice that node 5 has two children, one is node 6, another one is node 14. There is only one string follow the transition symbol 4 from node 5 to node 6, thus $f_5(4) = 1$. Likewise we have $f_5(3) = 2$ and $f_5(a) = 0$ for $a \neq 3, 4$. Since a node with a double circle means there is at least one string ending at this node, we know there is no string ending at node 5, and obtain $f_5(\sharp) = 0$.

In the example, we have insufficiently few data, so the frequency is not accurate as the approximation of probabilities. Ideally, when we go through a large set of sample data, we can get a large PTA which approximate the probabilities quite well. From this PTA, one can merge the compatible nodes to get an SFA. We regard this resulting SFA as an representative of the writing style of the author.

As an example, we look at another set of data as sample. Suppose we have a set of strings:$\{0, 01, 01, 011, 0101, 0101, 0101, 0101, 0101, 010101, 010101\}$, we can build the following PTA according to the method described earlier:



We calculate the values of n_i, $f_i(\sharp)$ and $f_i(a)$ for $a = 0, 1$ and $0 \leq i \leq 7$ in the following table.

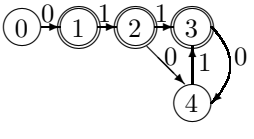
Node i	0	1	2	3	4	5	6	7
n_i	11	11	10	1	7	7	2	2
$f_i(\sharp)$	0	1	2	1	0	5	0	2
$f_i(0)$	11	0	7	0	0	2	0	0
$f_i(1)$	0	10	1	0	7	0	2	0

It is obvious from the table that node 3 and node 7 are equivalent. If we let $\alpha = 0.7$, then one can check that node 5 and node 7(or 3) are compatible because

$$\Big|\frac{f_5(\sharp)}{n_5} - \frac{f_7(\sharp)}{n_7}\Big| = \frac{2}{7} < \sqrt{\frac{1}{2}\log\frac{2}{0.7}}\Big(\frac{1}{\sqrt{n_5}} + \frac{1}{\sqrt{n_7}}\Big).$$

$$\Big|\frac{f_5(0)}{n_5} - \frac{f_7(0)}{n_7}\Big| = \frac{2}{7} < \sqrt{\frac{1}{2}\log\frac{2}{0.7}}\Big(\frac{1}{\sqrt{n_5}} + \frac{1}{\sqrt{n_7}}\Big).$$

Similarly, one can verify that node 4 and node 6 are compatible. For other pair of nodes, this inequality does not hold. So we can merge nodes $3, 5, 7$ and get the following SFA:



Now for any piece of writing, we form the sequences of digits according to the method mention above. Suppose the number of sequences is m. For each sequence, we test if it is accepted by the SFA. The number of accepting sequences is denoted by m_a. Therefore we get a quotient m_a/m which is called the accepting probability.

For instance, if we have a set of 4 strings $\{01010101, 0111, 001, 01010\}$ which are all different from our sample strings. Applying our test program, we see that only the first string 01010101 is accepted by this SFA. The accepting probability is then equal $\frac{1}{4}$. We remark that the accepting probability depends on the

parameter α in our method. This parameter is used to control the accuracy of our merge process. Sometimes it is possible to merge non-equivalent states when α is too small.

5 Results

In this section we present the results of the running of our program. The author we choose is J.K.Rowling and the writing sample is her book *Harry Potter and the Order of the Phoenix*. The test writings are her other three books:

Book 1 : *Harry Potter and the Sorcerer's Stone*

Book 2 : *Harry Potter and the Chamber of Secrets*

Book 3 : *Harry Potter and the Prisoner of Azkaban*

and one book of Gabriel Garcia Marquez:

Solitude : *One Hundred Years Of Solitude*

In our program, we choose a sentence as a unit. One reason is that we already get good results with this choice. Another reason is that if we choose larger unit, the program will run longer. Since our results are good enough to distinguish authors, we don't bother to waste time to get similar results.

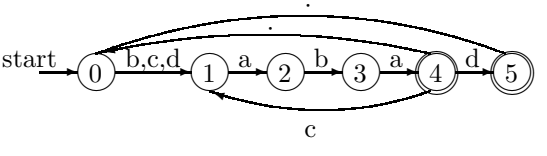
As we use one sentence as unit, the patterns we catch all have size smaller than one sentence. Any larger size pattern can be absorbed in the automaton. Now we give an example to illustrate this situation. The following paragraph consists of five sentences:

`dabad.caba.baba.cabad.cabacaba.`

One pattern is the repeat of string *aba* appeared in every sentence.

`dabad.caba.baba.cabad.cabacaba.`

According to our method, the automaton is



Note that there is another larger pattern *abad.caba* across sentences:

`dabad.caba.baba.cabad.cabacaba.`

This string can be accepted by the previous automaton. If we use two sentences as a unit, we can get a PTA, after merging, we will get the same automaton as

above. However, it takes more time using this algorithm. So it is this technical reason we choose one sentence as a unit. Next, we present our results. First of all, we use the PTA as our SFA, that is, we do not merge the states of the PTA. In this case, the PTA accepts exactly the set of strings of the sample data. The following table gives the result:

	♯ total sentences	♯ accepted sentences	accepting probability
Book 1	6186	3904	0.631102
Book 2	6360	4007	0.630031
Book 3	8425	5554	0.659228
Solitude	5678	1751	0.308383

In this table, one can find a big gap of the accepting probabilities between the book of same author and the book of different author. Next we fix the parameter $\alpha = 0.7$. Then after merging we get an SFA as the writing pattern of the author. The result of accepting probability are given in the following table.

	♯ total sentences	♯ accepted sentences	accepting probability
Book 1	6186	4285	0.692693
Book 2	6360	4390	0.690252
Book 3	8425	6021	0.714659
Solitude	5678	2079	0.36615

The accepting probabilities in this table is greater than the correspondence probabilities in the table before merging. This is because after merging, the new SFA can accept more strings than the one before merging. These new strings cannot be identified by the sample data. We remark that if we take the parameter $\alpha \leq 0.55$ in our program, then a lot of non-equivalent states will merge due to a large error used in the comparison of frequencies. The accepting probability is greater than 0.97 in all four books. This phenomenon does not imply that our method is not effective. It reminds us to pick the parameter appropriately to get the best result. In fact, our first table of accepting probability obtained from the PTA(before merging) has already show the difference between Book 1–3 and Solitude.

We believe that there are tremendous potential generalization of this method. For instance, one can change the size of the segment from one sentence to several sentences, or one can use a finer classification of the set of function words instead of part of speech. Even further, one can also include some type of content words into the sample data instead of the set of function words.

Another direction to refine the result is to combine this method with the traditional statistic methods. The author is working on this direction and obtained partial results.

References

1. Grieve, J.: Quantitative Authorship Attribution: An evaluation of Techniques. Literary and Linguistic Computing 22(3), 251–270 (2007)
2. Baliga, P., Lin, T.Y.: Kolmogorov Complexity Based Automata Modeling for Intrusion Detection. In: Proceeding of the 2005 IEEE International Conference on Granular Computing, Beijing, China, July 25-27, pp. 387–392 (2005)
3. Lin, T.Y.: Rough Patterns in Data-Rough Sets and Foundation of Intrusion Detection Systems. Journal of Foundation of Computer Science and Decision Support 18(3-4), 225–241 (1993)
4. Lin, T.Y.: Patterns in Numerical Data: Practical Approximations to Kolmogorov Complexity. In: Zhong, N., Skowron, A., Ohsuga, S. (eds.) RSFDGrC 1999. LNCS, vol. 1711, pp. 509–513. Springer, Heidelberg (1999)
5. Carraso, R.C., Oncina, J.: Learning stochastic regular grammars by means of a state merging method. In: Carrasco, R.C., Oncina, J. (eds.) ICGI 1994. LNCS (LNAI), vol. 862, pp. 139–152. Springer, Heidelberg (1994)
6. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Language, and Computation. Addison Wesley, Reading (2001)
7. Paek, T., Chandrasekar, R.: Windows as a Second Language: An Overview of the Jargon Project. In: Proceedings of the First International Conference on Augmented Cognition (2005)
8. Koppel, M., Argamon, S., Shimoni, A.: Automatically categorizing written texts by author gender. Literary and Linguistic Computing 17(4), 401–412 (2002)
9. Sekar, R., Bendre, M., Dhurjati, D., Bollineni, P.: A Fast Automaton-Based Method for Detecting Anomalous Program Behaviors. In: Proceedings. 2001 IEEE Symposium on Security and Privacy (2001)
10. Young-Lai, M., Tompa, F.: Stochastic Grammatical Inference of Text Database Structure. Machine Learning, 111–137 (2000)
11. Mosteller, F., Wallace, D.L.: Inference and disputed authorship: The Federalist. Addison-Wesley, Reading (1964)

Detection of Risk Factors as Temporal Data Mining

Shoji Hirano and Shusaku Tsumoto

Department of Medical Informatics, Shimane University, School of Medicine
89-1 Enya-cho, Izumo, Shimane 693-8501, Japan
hirano@ieee.org, tsumoto@computer.org

Abstract. Hosptial information system (HIS) collects all the data from all the branches of departments in a hospital, including laboratory tests,physiological tests, electronic patient records. Thus, HIS can be viewed as a large heterogenous database, which stores chronological changes in patients' status. In this paper, we applied trajectory mining method to the data extracted from HIS. Experimental results demonstrated that the method could find the groups of trajectories which reflects temporal covariance of laboratory examinations.

1 Introduction

It has passed about twenty years since clinical information are stored electronically as a hospital information system since 1980's. Stored data include from accounting information to laboratory data and even patient records are now started to be accumulated: in other words, a hospital cannot function without the information system, where almost all the pieces of medical information are stored as multimedia databases. Especially, if the implementation of electronic patient records is progressed into the improvement on the efficiency of information retrieval, it may not be a dream for each patient to benefit from the personal database with all the healthcare information, "from cradle to tomb". However, although the studies on electronic patient record has been progressed rapidly, reuse of the stored data has not yet been discussed in details, except for laboratory data and accounting information to which OLAP methodologies are applied. Even in these databases, more intelligent techniques for reuse of the data, such as data mining and classical statistical methods has just started to be applied from 1990's[1,2].

In this paper, we applied trajectory mining method to the data extracted from HIS. Experimental results demonstrated that the method could find the groups of trajectories which reflects temporal covariance of laboratory examinations.

The remainder of this paper is organized as follows. In Section 2 we describe the methodoology, including preprocessing of the data. In Section 3 we show experimental results on chronic hepatitis data (albumin-platelet trajectories and cholinesterase-platelet trajectories). Section 4 shows comparison of the clusters obtained and risk analysis of platelet counts. Finally, Section 5 is a conclusion of this paper.

2 Methods: Overview

Figure 1 shows an overview of the whole process of clustering of trajectories. First, we apply preprocessing of a raw temporal sequence for each variable. Secondly, a trajectory

S. Chawla et al. (Eds.): PAKDD 2008 Workshops, LNAI 5433, pp. 143–156, 2009.

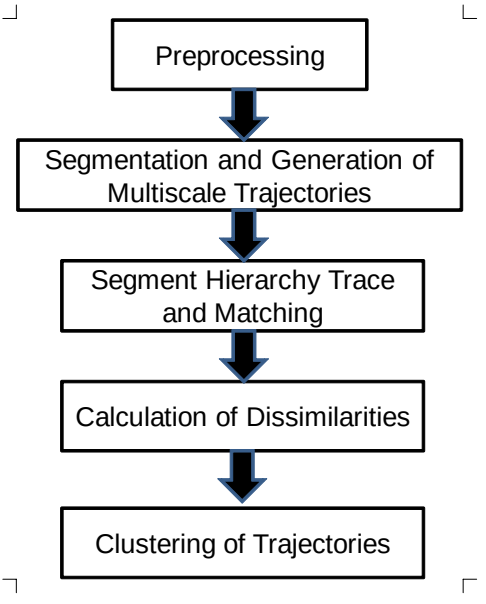


Fig. 1. Overview of Trajectory Clustering

of laboratory tests is calculated for each patient, segmentation technique is applied to each sequence for generation of a segmentation hiearchy. Third, we trace segemented sequences and search for matching between two sequences in a hiearchical way Then, dissimilarities are calculated for matched sequences. Finally, we apply clustering to the dissimilarities obtained.

2.1 Preprocessing

For further details on each methodology, please refer to [3,4,5].

3 Experimental Results

We applied our method to the chronic hepatitis dataset which was a common dataset in ECML/PKDD discovery challenge 2002-2004 [6]. The dataset contained time series laboratory examinations data collected from 771 patients of chronic hepatitis B and C. In this work, we focused on analyzing the temporal relationships between platelet count (PLT), albumin (ALB) and cholinesterase (CHE), that were generally used to examine the status of liver function. Our goals were set to: (1) find groups of trajectories that exhibit interesting patterns, and (2) analyze the relationships between these patterns and the stage of liver fibrosis.

We selected a total of 488 cases which had valid examination results for all of PLT, ALB, CHE and liver biopsy. Constitution of the subjects classified by virus types and administration of interferon (IFN) was as follows. Type B: 193 cases, Type C with IFN:

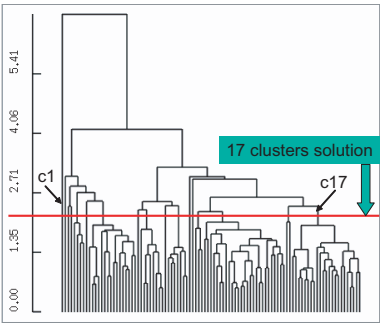


Fig. 2. Dendrogram for ALB-PLT trajectories in Type C without IFN dataset

Table 1. Cluster constitutions of ALB-PLT trajectories, stratified by fibrotic stages. Small clusters of $N < 2$ were omitted.

Cluster	# of Cases / Fibrotic stage				Total
	F0,F1	F2	F3	F4	
5	0	1	0	3	4
7	3	2	2	9	16
9	6	2	0	0	8
11	7	0	0	0	7
14	2	1	0	0	3
15	17	2	7	1	27
16	1	0	1	0	2
17	20	2	1	0	23

296 cases, Type C without IFN: 99 cases. In the following sections, we mainly describe the results about Type C without IFN cases, which contained the natural courses of Type C viral hepatitis.

Experiments were conducted as follows. This procedure was applied separately for ALB-PLT, CHE-PLT and ALB-CHE trajectories.

1. Select a pair of cases (patients) and calculate the dissimilarity by using the proposed method. Apply this procedure for all pairs of cases, and construct a dissimilarity matrix.
2. Create a dendrogram by using conventional hierarchical clustering [4] and the dissimilarity matrix. Then perform cluster analysis.

Parameters for multiscale matching were empirically determined as follows: starting scale = 0.5, scale interval = 0,5, number of scales = 100, weight for segment replacement cost = 1.0. We used group average as a linkage criterion for hierarchical clustering. The experiments were performed on a small PC cluster consisted of 8 DELL PowerEdge 1750 (Intel Xeon 2.4GHz 2way) workstations. It took about three minutes to make the dissimilarity matrix for all cases.

Results on ALB-PLT Trajectories. Figure 2 shows the dendrogram generated from the dataset on Type C without IFN cases. The dendrogram suggested splitting of the data into two or three clusters; however, in order to carefully examine the data structure, we avoided excessive merge of clusters and determined to split it into 17 clusters where dissimilarity increased relatively largely at early stage. For each of the 8 clusters that contained ≥ 2 cases, we classified cases according to the fibrotic stage. Table 1 shows the summary. The leftmost column shows cluster number. The next column shows the number of cases whose fibrotic stages were F0 or F1. The subsequent three columns show the number of F2, F3, and F4 cases respectively. The rightmost column shows the total number of cases in each cluster.

From Table 1, it could be recognized that the clusters can be globally classified into one of the two categories: one containing progressed cases of liver fibrosis (clusters 5 and 7) and another containing un-progressed cases (clusters 9, 11, 14, 15, 16 and 17). This

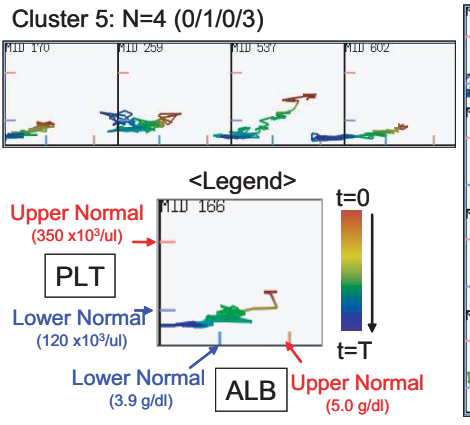


Fig. 3. Trajectories in Cluster 5

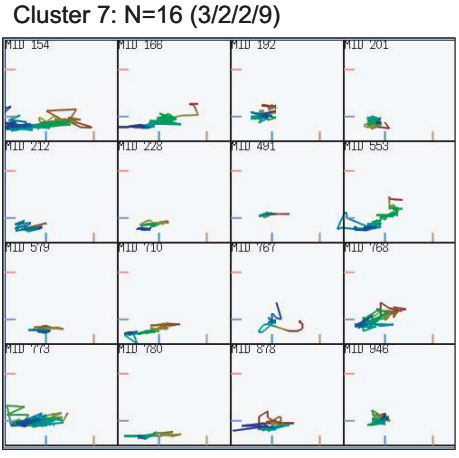


Fig. 4. Trajectories in Cluster 7

can be confirmed from the dendrogram in Figure 2, where these two types of clusters appeared at the second devision from the root. This implied that the difference about ALB and PLT might be related to the fibrotic stages.

In order to recognize the detailed characteristics of 8 clusters, we observed the feature of grouped trajectories. Figures 3-6 show the examples of grouped ALB-PLT trajectories. Each quadrate region contains a trajectory of ALB-PLT values for a patient. If the number of cases in a cluster was larger than 16, the first 16 cases w.r.t. ID number were selected for visualization. The bottom part of Figure 3 provides the legend. The horizontal axis represents ALB value, and the vertical axis represents PLT value. Lower end of the normal range (ALB:$3.9g/dl$, PLT:$120\times10^3/ul$) and Upper end of the normal range (ALB:$5.0g/dl$, PLT:$350\times10^3/ul$) were marked with blue and red short lines on each axis respectively. Time phase on each trajectory was represented by color phase: red represents the start of examination, and it changes toward blue as time proceeds.

Figure 3 shows cases grouped into cluster 5 which contained remarkably many F4 cases (3/4). The skewed trajectory of ALT and PLT clearly demonstrated that both values decreased from the normal range to the lower range as time proceeded, due to the dysfunction of the liver. Cluster 7, shown in Figure 4, also contained similarly large number of progressed cases (F4:9/16, F3:2/16) and exhibited the similar characteristics, though it was relatively weaker than in cluster 5.

On the contrary, clusters that contained many un-progressed cases exhibited different characteristics. Figure 5 shows the trajectories grouped into cluster 17, where the number of F0/F1 cases was large (20/23). Most of the trajectories moved within the normal range, and no clear feature about time-direction dependency was observed. Figure 6 (top) shows the trajectories in cluster 11, where all of 7 cases were F0/F1. They moved within the normal range, but the PLT range was higher than in cluster 17.

Figure 6 (bottom) shows the trajectories in cluster 14, where trajectories exhibited skewed shapes similarly to cluster 5. But this cluster consisted of F0/F1 and F2 cases, whereas cluster 5 contained mainly progressed cases. The reason why these cases were

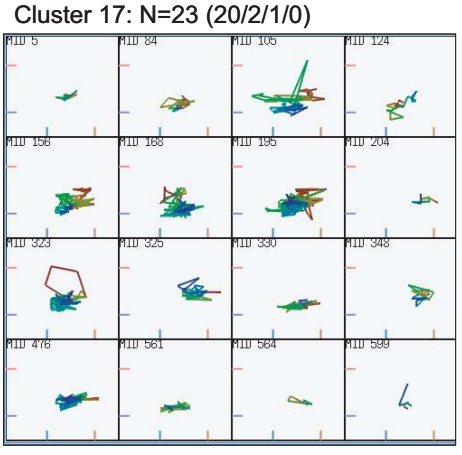


Fig. 5. Trajectories in Cluster 17

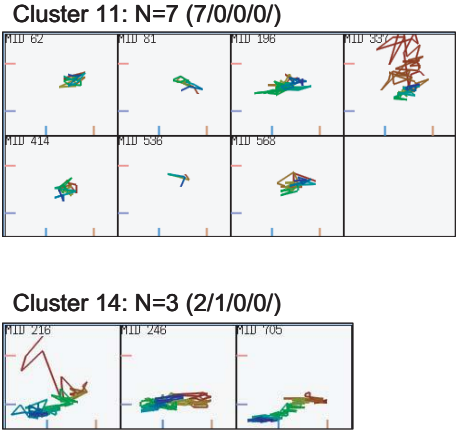


Fig. 6. Trajectories in Cluster 11 and 14

separated into different clusters should be investigated further, but it seemed that the difference of progress speed of liver fibrosis, represented as a velocity term, might be a candidate cause.

Results on CHE-PLT Trajectories. Figure 7 shows the dendrogram generated from CHE-PLT trajectories of 99 Type C without IFN cases. Similarly to the case of ALB-PLT trajectories, we split the data into 15 clusters where dissimilarity increased largely at early stage. Table 2 provides cluster constitution stratified by fibrotic stage. In Table 2, we could observe a clear feature about the distribution of fibrotic stages over clusters. Clusters such as 3, 4, 6, 7 and 8 contained relatively large number of F3/F4 cases, whereas clusters such as 9, 11, 12, 13, 14, 15 contained no F3/F4 cases. These two types of clusters were divided at the second branch on the dendrogram; therefore it implied that, with respect to the similarity of trajectories, the data can be globally split into two categories, one contains the progressed cases and another contained un-progressed cases.

Now let us examine the features of trajectories grouped into each cluster. Figure 8 shows CHE-PLT trajectories grouped into cluster 3. The bottom part of the figure provides the legend. The horizontal axis corresponds to CHE, and the vertical axis corresponds to PLT. This cluster contained four cases: one F3 and three F4. The trajectories settled around the lower bounds of the normal range for PLT ($120 \times 10^3/ul$), and below the lower bounds of CHE (180 IU/l), with global direction toward lower values. This meant that, in these cases, CHE deviated from normal range earlier than PLT.

Figure 9 shows trajectories grouped into cluster 4, which contained nine F3/F4 cases and three other cases. Trajectories in this cluster exhibited interesting characteristics. First, they had very clear descending shapes; in contrast to trajectories in other clusters in which trajectories changed directions frequently and largely, they moved toward the left corner with little directional changes. Second, most of the trajectories settled below the normal bound of PLT whereas their CHE values ranged within normal range at early phase. This meant that, in these cases, CHE deviated from normal range later than PLT.

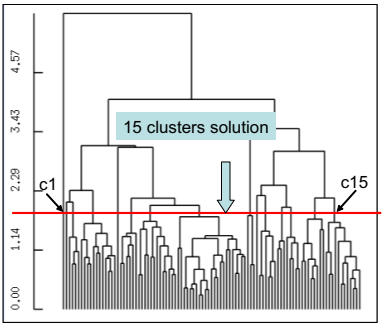


Fig. 7. Dendrogram for Type C without IFN dataset (CHE-PLT trajectories)

Table 2. Cluster constitutions of CHE-PLT trajectories, stratified by fibrotic stages. Small clusters of $N < 2$ were omitted.

Cluster	# of Cases / Fibrotic stage				Total
	F0,F1	F2	F3	F4	
3	0	0	1	3	4
4	2	1	2	7	12
6	3	0	1	2	6
7	5	2	3	3	13
8	9	8	4	2	23
9	1	2	0	0	3
11	4	2	0	0	6
12	2	0	1	0	3
13	5	0	0	0	5
14	8	0	0	0	8
15	12	0	0	0	12

Figure 10 shows trajectories grouped into cluster 6, which contained three F3/F4 cases and three other cases. Trajectories in this cluster exhibited descending shapes similarly to the cases in cluster 4. The average levels of PLT were higher than those in cluster 4, and did not largely deviated from the normal range. CHE remained within the normal range for most of the observations.

Figure 11 shows trajectories grouped into cluster 15, which contained twelve F0/F1 cases and no other cases. In contrast to the high stage cases mentioned above, trajectories settled within the normal ranges for both CHE and PLT and did not exhibit any remarkable features about their directions.

These results suggested the followings about the CHE-PLT trajectories on type C without IFN cases used in this experiment: (1) They could be globally divided into two categories, one containing high-stage cases and another containing low-stage cases, (2) trajectories in some high-stage clusters exhibited very clear descending shapes. (3) in a group containing descending trajectories, PLT deviated from normal range faster than CHE, however, in another group containing descending trajectories, PLT deviated from normal range later than CHE.

Results on ALB-CHE Trajectories. Figure 12 shows the dendrogram generated from the dataset on Type C without IFN cases. The dendrogram suggested splitting of the data into two or three clusters; however, in order to carefully examine the data structure, we avoided excessive merge of clusters and determined to split it into 15 clusters where dissimilarity increased relatively largely at early stage. For each of the 8 clusters that contained ≥ 2 cases, we classified cases according to the fibrotic stage. Table 1 shows the summary. The leftmost column shows cluster number. The next column shows the number of cases whose fibrotic stages were F0 or F1. The subsequent three columns show the number of F2, F3, and F4 cases respectively. The rightmost column shows the total number of cases in each cluster.

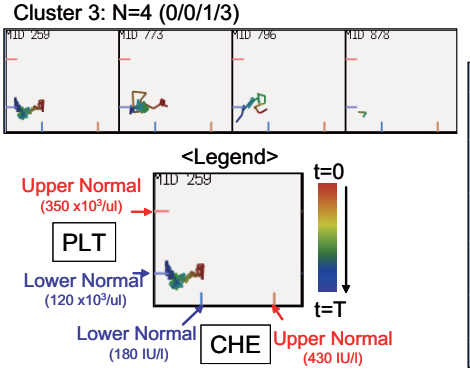


Fig. 8. Trajectories in Cluster 3

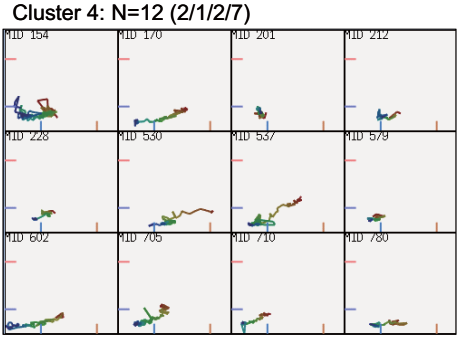


Fig. 9. Trajectories in Cluster 4

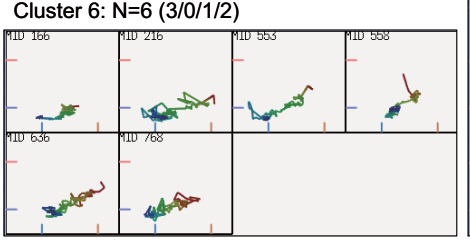


Fig. 10. Trajectories in Cluster 6

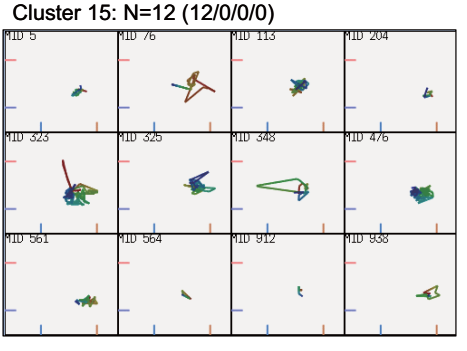


Fig. 11. Trajectories in Cluster 15

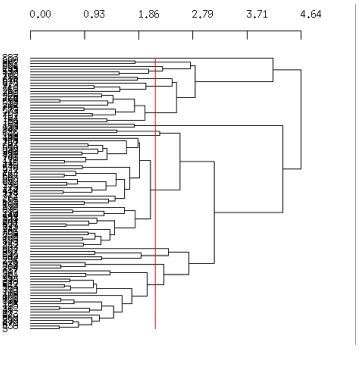


Fig. 12. Dendrogram for ALB-CHE trajectories in Type C without IFN dataset

Table 3. Cluster constitutions of ALB-CHE trajectories, stratified by fibrotic stages. Small clusters of $N < 2$ were omitted.

Cluster	# of Cases / Fibrotic stage				Total
	F0,F1	F2	F3	F4	
2	0	0	0	2	2
4	0	0	0	3	3
5	0	0	1	1	2
6	0	0	0	4	4
7	3	1	2	5	11
8	1	1	0	0	2
9	2	0	0	0	2
11	22	9	8	1	40
13	2	2	0	0	4
14	3	0	0	0	3
15	19	2	0	1	22

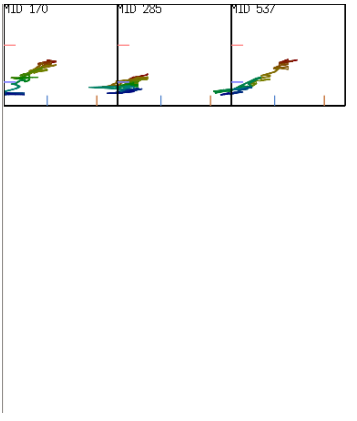


Fig. 13. Trajectories in Cluster 4

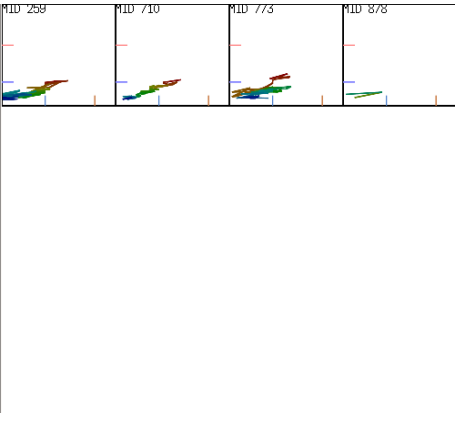


Fig. 14. Trajectories in Cluster 6

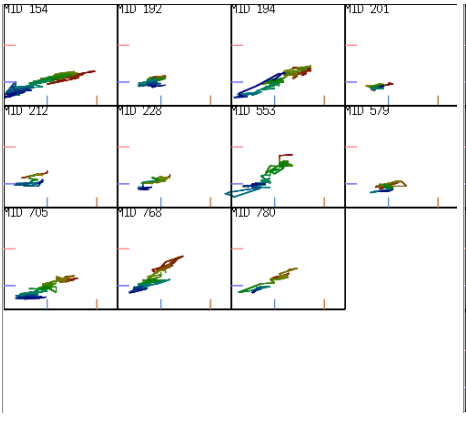


Fig. 15. Trajectories in Cluster 7

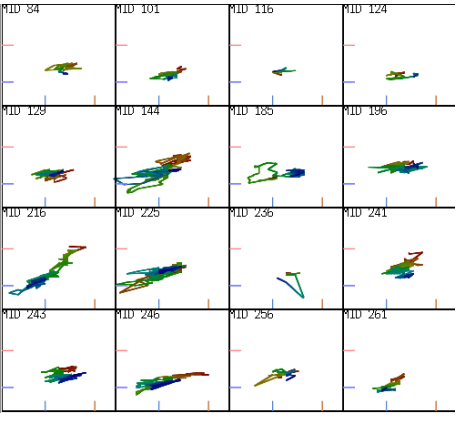


Fig. 16. Trajectories in Cluster 11

From Table 3, it could be recognized that the clusters can be globally classified into one of the two categories: one containing progressed cases of liver fibrosis (clusters 2, 4, 5, 6and 7) and another containing un-progressed cases (clusters 8, 9, 11, 13, 14 and 15). This can be confirmed from the dendrogram in Figure 12, where these two types of clusters appeared at the second devision from the root. This implied that the difference about ALB and PLT might be related to the fibrotic stages.

In order to recognize the detailed characteristics of 8 clusters, we observed the feature of grouped trajectories. Figures 3-6 show the examples of grouped ALB-PLT trajectories. Each quadrate region contains a trajectory of ALB-PLT values for a patient. If the number of cases in a cluster was larger than 16, the first 16 cases w.r.t. ID number were selected for visualization. The bottom part of Figure 3 provides the legend. The horizontal axis represents ALB value, and the vertical axis represents CHE value.

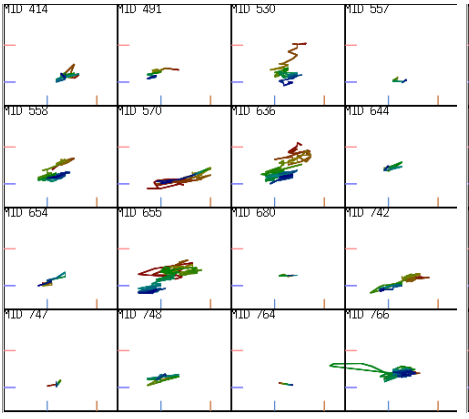


Fig. 17. Trajectories in Cluster 11(2)

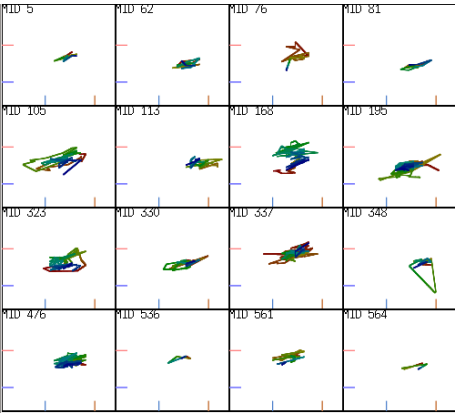


Fig. 18. Trajectories in Cluster 15

Time phase on each trajectory was represented by color phase: red represents the start of examination, and it changes toward blue as time proceeds.

Figures 13 and 14 shows cases grouped into cluster 4 and 6 which contained only F4 cases (3 and 4). The skewed trajectory of ALT and CHE clearly demonstrated that both values decreased from the normal range to the lower range as time proceeded, due to the dysfunction of the liver. Cluster 7, shown in Figure 15, also contained similarly large number of progressed cases (F1: 3, F2: 1, F3: 2, F4: 5) and exhibited the similar characteristics, though it was relatively weaker than in cluster 4 and 6.

On the contrary, clusters that contained many un-progressed cases exhibited different characteristics. Figures 16 and 17 show the trajectories grouped into cluster 11, where the number of F0/F1 cases was large (31/40). Most of the trajectories moved within the normal range, but some decreasing cases were included in this cluster, and no clear feature about time-direction dependency was observed. Figure 18 shows the trajectories in cluster 15, where 19 of 22 cases were F0/F1 and the sequences moved within the normal range.

In summary, the degree of covariance between ALB and CHE is higher than those between ALB and PLT or CHE and PLT. Samples are better split into F4-dominant cases and F0/F1-dominant cases.

4 Discussion: Risk Analysis

4.1 Comparison of Clustering Results

Table 4 compares the characteristics of clustering results. As shown in the table, it seems that a combination of ALB and CHE generates a slightly better results than other pairs with respect to the degree of seperation of fibrotic stages.

Table 5 shows the contingency table between CHE-PLT and ALB-PLT whose examples belong to Cluster No.7 in ALB-CHE. Compared with the results in Table 2 and

Table 4. Comparison of Clustering Results

Pair	#Clusters	$\#Examples$	Most Impurity Clusters				
		> 1	#	F1	F2	F3	F4
ALB-PLT	17	8	16	3	2	2	9
CHE-PLT	15	11	13	5	2	3	3
ALB-CHE	15	11	11	3	1	2	5

Table 5. Contingency Table of Cluster No.7 in ALB-CHE

		ALB-PLT			
		No.7	No.14	No.15	Total
CHE-PLT	No. 4	7	1	0	8
	No. 6	2	0	0	2
	No. 7	0	0	1	1
	Total	9	1	1	11

Table 1, these cases covers impure clusters in CHE-PLT and ALB-PLT. This observation shows that this cluster should be carefully examined by additional information, since the cluster includes several F0/F1 cases whose PLT is decreasing.

4.2 Risk Analysis for Liver Fibrosis Based on the PLT Counts

Determination of the stage of liver fibrosis is usually done with liver biopsy, which is an invasive examination. In recent years, platelet count has been receiving considerable attention as an non-invasive index reflecting the liver dysfunctions, which may be associated with the fibrotic stage in chronic hepatitis. Several researchers has reported the relationships between platelet counts and fibrotic stages. For example, Matsumura et al. [7] showed the data that F1: $20.3 \pm 5.2 (\times 10^4 \mu l)$, F2: 16.0 ± 4.9, F3: 13.0 ± 4.0, F4: 11.8 ± 4.1 and in LC 11.8 ± 4.1. Pohl et al. [8] proposed combination of AST/ALT with PLT as a predictor of fibrotic stages.

Matsumura et al. [7] also reported the progress speed of liver fibrosis examined on the patients of Type C chronic hepatitis in Japan. They used the date of blood transplants, which could be associated with F0, and the date and results of liver biopsy for calculating the progress speed. The results was about 0.12 ± 0.15 stage/year.

The goal of our study is to analyze, without information about blood transplants, the progress speed of liver fibrosis. As a preliminary stage, we attempted to calculate (1) years required for reaching F4 stage, and (2) years elapsed between stages, by combining the fibrotic stages predicted from PLT level and observed by liver biopsy.

Here we made an assumption: "If the PLT level of a patient is continuously lower than the normal range for at least 6 months, and after that never keeps normal range more than 6 months, then the patient is F4." Based on this assumption, we first examined whether and when a patient became F4. Then by subtracting dates and stages from those obtained by biopsy, we calculated the elapsed years.

Table 6. Result of sequence classification. Judging criteria for declination are: (1) PLT becomes continuously lower than the normal range over 6 months, (2) Recovered PLT level cannot continuously maintain the normal range for 6 months. Both criteria should be satisfied.

No biopsy	Short	Inhomogeneous	Available		Total
			Declinated	Normal	
222	82	28	97	291	720

As a pre-process, we selected the cases to analyze according to the following procedure.

1. Exclude cases that meet any of the following three conditions from analysis: (1) No biopsy - biopsy information is not available. (2) Short sequence - the number of examinations is less than 2 or the duration of examination is shorter than 2 years. (3) Inhomogeneous sequence - Deviation of examination intervals is larger than 1 year.
2. Rearrange the sampling intervals of each sequence into one-week. The starting date of re-sampling is selected independently to each case, based on two criteria that (1) it is the day of a week on which the patient most frequently received examinations, and (2) it is the closest date to the first examination. If examination data were missing, we inserted a predicted value by linearly interpolating nearest examination results. In the following procedures we used these rearranged sequences.
3. Smooth each sequence in order to remove short-term changes. We performed convolution with discrete Gaussian kernel with support width of 6 month (26 weeks; $\sigma = 2.8$).
4. From the head of a sequence, search the first point that satisfies both of the following two conditions.
   (a) PLT level becomes continuously lower than the normal range over the next 6 months. Duration of IFN therapy is not included therein as it may induces short-term decrease of PLT.
   (b) Recovered PLT level cannot continuously maintain the normal range for 6 months.

   If found, let the detected point the date of declination from normal range. Otherwise, the case was considered to keep normal PLT range and removed from analysis.

Table 6 shows the result of sequence classification by the above four procedure. A total of 97 cases classified as 'declinated' were the subject of analysis.

Table 7 summarizes calculated years for reaching F4 (first examination date basis), for the 97 declinated cases in Table 6, stratified by the virus types and fibrotic stage. Note that years=0 if the date of declination was earlier than the date of first examination. For each of type B, C with IFN and C without IFN groups, we performed statistical tests (ANOVA) aiming at detecting differences of mean years for reaching F4 with respect to the biopsy-based fibrotic stages. The result of Type C IFN was $p = 0.012 (< 0.05)$, indicating that significant differences of years exist among fibrotic stages. However, this is primarily due to one exceptionally long case in F0; tests after removing this case yielded $p = 0.291$, indicating that there was no significant difference on the years

Table 7. Years for reaching F4 (First-exam basis) stratified by virus types and fibrotic stages. Summary for 97 declination cases in Table 6.*

Type	Fibrotic Stage	Cases	Years for reaching F4, first-exam basis (years)		
			Mean	Median	SD
B	1	5	2.36	0	3.53
	2	9	5.04	1.79	6.29
	3	8	2.44	0.66	3.62
	4	9	1.43	0	2.56
	subtotal	31	2.89	0.56	4.38
C IFN	0	1	13.08	13.08	–
	1	8	4.05	3.87	3.15
	2	6	3.89	3.70	3.90
	3	10	2.26	2.30	2.43
	4	15	1.84	0	3.02
	subtotal	40	2.98	2.32	3.46
C w/o IFN	1	7	4.29	4.50	3.52
	2	2	4.05	4.05	5.72
	3	5	2.49	0.12	3.64
	4	12	2.21	0	3.15
	subtotal	26	2.97	0.65	3.42
Total		97	2.95	1.19	3.73

*Fibrotic stages in the second column are based on biopsy. Years for reaching F4 was years from first exam to the date of declination under assumption that the fibrotic stage at the date of declination was F4. If the date of declination was the same as or before the first exam, years were treated as 0.

for reaching F4 among fibrotic stages. Results for Type B and Type C w/o IFN were $p = 0.357$ and $p = 0.613$ respectively, indicating no significant differences. Kruscal-Wallis tests yielded the same conclusion. Between-group comparison of Type B, Type C with IFN and Type C w/o IFN groups was $p = 0.960$.

Years for reaching F4 in Table 7 were years between the first dates of PLT examinations and the date of PLT declination. Therein we assume that the fibrotic stage at first examination is the same as that at first biopsy. However, the date of first biopsy and the date of first PLT examination are generally different; in some cases they are several years apart. This implies that the stages might also be different. Therefore, we calculated years for reaching F4 biopsy basis, which is years from the date of first biopsy to the date of PLT declination. Additionally, based on the assumption that the stage at PLT declination should be F4, we calculated elapsed years between stages by the following formula: (date of declination - date of first biopsy) / (4 - fibrotic stage at biopsy). If declination occurred before the first biopsy, years were treated as 0. Table 8 summarizes the results.

As we did in the first-exam basis results, for each of type B, C with IFN and C without IFN groups, we performed statistical tests with ANOVA aiming at detecting differences of mean years for reaching F4 w.r.t. the fibrotic stages. The results were $p = 0.421, 0.020(< 0.05), 0.119$ for each group respectively. In Type C IFN there appeared significant difference among stages, however, this was primarily due to one

Table 8. Years for reaching F4 (biopsy basis) and years between stages stratified by virus type and fibrotic stages. Summary for 97 declination cases in Table 6.[**]

Type	Fibrotic Stage	Cases	Years for reaching F4, biopsy basis (years)			Years between stages (years/stage)		
			Mean	Median	SD	Mean	Median	SD
B	1	5	3.70	3.71	3.46	1.23	1.24	1.51
	2	9	4.26	2.85	5.45	2.13	1.42	2.73
	3	8	2.12	0.22	3.72	2.12	0.22	3.72
	4	9	1.31	0	2.54	—	—	—
	subtotal	31	2.77	0.52	4.00	1.92	0.62	2.80
C IFN	0	1	13.08	13.08	—	3.27	3.27	—
	1	8	2.44	2.38	2.36	0.81	0.79	0.79
	2	6	2.54	0.83	3.42	1.27	0.41	1.71
	3	10	1.99	2.14	2.05	1.99	2.14	2.05
	4	15	2.01	0	3.51	—	—	—
	subtotal	40	2.44	1.18	3.30	1.49	1.07	1.66
C w/o IFN	1	7	1.47	1.17	1.76	0.49	0.39	0.58
	2	2	3.70	3.70	5.24	1.85	1.85	2.62
	3	5	2.27	0	3.54	2.27	0	3.54
	4	12	0.26	0	0.54	—	—	—
	subtotal	26	1.23	0	2.27	1.32	0.20	2.30
Total		97	2.22	0.44	3.34	1.6	0.80	2.25

[**]Fibrotic stages in the second column are based on biopsy. Years for reaching F4 were years from first biopsy to the date of declination under assumption that the fibrotic stage at the date of declination was F4. If the date of declination was the same as or before the first biopsy, years were treated as 0. Years between stages were calculated by (years for reaching F4)/(4 − stage at biopsy).

exceptionally long case in F0; tests after removing this case yielded $p = 0.970$, indicating that there was no significant difference on the years for reaching F4 even in the biopsy-date basis measurement. Kruscal-Wallis tests yielded the same conclusion.

Similarity, for each of type B, C with IFN and C without IFN groups, we performed statistical tests with ANOVA aiming at detecting differences of mean elapsed years between stages w.r.t. the fibrotic stages. In this test we removed F4 cases as we could not measure the elapsed years. For the same reason, we excluded F4 cases for calculating values such as mean and SD in Table 8. The results of ANOVA were $p = 0.836, 0.425, 0.340$, indicating that there was no significant differences among stages, including F0, for all of the three groups.

In summary, the results suggests that there were no significant difference of years for reaching F4 and of elapsed years between stages, with respect to fibrotic stages, virus types and administration of IFN. However, it is interesting that the elapsed years between stages were 1-2 years/stage in almost all groups. Let us simply invert it into progress speed for comparison with other resources. The result is about $1/1.32 = 0.76$stage/year for example of Type C w/o IFN cases. This is faster than in [7] (0.12 ± 0.15 stage/year), implying that the liver fibrosis might proceed faster.

It should be noted that the results of analysis should not be generalized because (1) we assume that a patient can be regarded as F4 when PLT level continuously declinates

from the normal range over long time, (2) we selected only exacerbating cases in which PLT continuously decreased, and (3) we did not take into account patient background information such as history of drinking. However, we consider that our approach of measuring elapsed years between stages by combining fibrotic stages obtained from biopsy and inferred from PLT level lead to find interesting results.

5 Conclusions

In this paper we propose a trajectory clustering method as multivariate temporal data mining and shows its application to data on chronic hepatits. Our method consists of a two-stage approach. Firstly, it compares two trajectories based on their structural similarity and determines the best correspondence of partial trajectories. Next, it calculates the value-based dissimilarity for the all pairs of matched segments and outputs the total sum as dissimilarity of the two trajectories.

Clustering experiments on the chronic hepatitis dataset yielded several interesting results. First, the clusters constructed with respect to the similarity of trajectories well matched with the distribution of fibrotic stages, especially with the distribution of high-stage cases and low-stage cases, for ALB-PLT, CHE-PLT and ALB-CHE trajectories. Among three combinations, ALB-CHE shows the highest degree of covariance, which means that CHE can be used to evaluate the trends of ALB.

Our next step is to extend bivariate trajectory analysis into multivariate one. From the viewpoint of medical application, our challenging issue will be to find a variable whose chronological trend is fitted to PLT.

References

1. Tsumoto, S.: Knowledge discovery in clinical databases and evaluation of discovered knowledge in outpatient clinic. Information Sciences, 125–137 (2000)
2. Tsumoto, S.: G5: Data mining in medicine. In: Kloesgen, W., Zytkow, J. (eds.) Handbook of Data Mining and Knowledge Discovery, pp. 798–807. Oxford University Press, Oxford (2001)
3. Hirano, S.: Tsumoto, S.: Multiscale comparison and grouping of trajectories on hospital laboratory examinations. Jounral of Knowledge and Information Systems (submitted, 2009)
4. Everitt, B.S., Landau, S., Leese, M.: Cluster Analysis, 4th edn. Arnold Publishers (2001)
5. Hirano, S., Tsumoto, S.: An indiscernibility-based clustering method with iterative refinement of equivalence relations - rough clustering -. J. Advanced Computational Intelligence and Intelligent Informatics 7, 169–177 (2003)
6. `http://lisp.vse.cz/challenge/` (2004)
7. Matsumura, H., Moriyama, M., Goto, I., Tanaka, N., Okubo, H., Arakawa, Y.: Natural course of progression of liver fibrosis in patients with chronic liver disease type c in japan - a study of 527 patients at one establishment in japan. J. Viral. Hepat. 7, 375–381 (2000)
8. Pohl, A., Behling, C., Oliver, D., Kilani, M., Monson, P., Hassanein, T.: Serum aminotransferase levels and platelet counts as predictors of degree of fibrosis in chronic hepatitis c virus infection. Am. J. Gastroenterol 96, 3142–3146 (2001)

Two-Phased Active Support Kernel Machine Learning

Yasusi Sinohara[1] and Atsuhiro Takasu[2]

[1] Central Research Institute of Electric Power Industry,
2-11-1 Iwado-kita, Komae-shi, Tokyo 201-8511, Japan
sinohara@criepi.denken.or.jp
[2] National Institute of Informatics,
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
takasu@nii.ac.jp

Abstract. Since SVMs have met with significant success in numerous real-world learning, SVM-based active learning has been proposed in the active learning context and it has been successfully applied in the domains like document classification, in which SVMs using linear kernel are known to be effective for the task. However, it is difficult to apply SVM-based active learning to general domains because the kernel used in SVMs should be selected properly before the active learning process but good kernels for the target task is usually unknown. If the pre-selected kernel is inadequate for the target data, both the active learning process and the learned SVM have poor performance. Therefore, new active learning methods are required which effectively find an adequate kernel for the target data as well as the labels of unknown samples in the active learning process.

In this paper, we propose a two-phased SKM-based active learning method for the purpose. By experiments, we show that the proposed SKM-based active learning method has quick response suited to interaction with human experts and can find an appropriate kernel among linear combinations of given multiple kernels.

1 Introduction

Support vector machines (SVMs) are kernel-based learning machines originally developed by Vapnik and co-workers [9]. Since they have met with significant success in batch learning of numerous real-world tasks, several researchers [2,4,8] have independently proposed the use of SVMs in active learning context, that is, situations in which unlabeled data is abundant but labeling data is expensive. The active linear-SVM learning has been successfully applied in the domain of text classification. But it can be an exceptional domain and it is usually difficult to apply active SVM learning to general problems because of the following reason.

In active learning, one kernel (specified by its kernel-type and kernel parameters) should be selected before active learning and if the selected kernel

S. Chawla et al. (Eds.): PAKDD 2008 Workshops, LNAI 5433, pp. 157–168, 2009.

is inadequate for the target problem, the performances of both the active learning process and the learned SVM greatly decrease. However, the kernel good for learning the target problem is usually unknown before active learning. Exceptionally in the domain of text classification, the linear kernel can be selected because the linear kernel is known to be effective.

To overcome the issue of kernel pre-selection, we first propose the use of support kernel machines (SKMs) instead of SVMs in active learning. Because solving SKMs is more time-consuming than solving SVMs, simple use of SKMs takes too much response time, i.e., the time to select an unlabeled sample for next labeling after one labeling. To reduce the response time important to interact with human experts, we further propose a two-phased active SKM learning which has both high responsiveness while active learning and the accurate classification power after learning.

2 Support Kernel Machines

Both SVM[9] and SKM[1] learn a separator $f(x)$ and predict the label $y \in \{\pm 1\}$ of input x by $y = \mathrm{sign}(f(x))$. In this section, we introduce SVMs and SKMs in view of the radius-margin bound R^2/M^2 closely related to the generalization error of SVMs [9].

2.1 Support Vector Machines (SVMs)

For a given set of training samples $\{(x_i, y_i)\}_{i=1,\ldots,N}$ and a given (non-linear) feature mapping $\phi(x)$ or kernel $K(x,z) = \phi(x)^\top \phi(z)$, the SVM finds a large margin separator $f(x) = w^\top \phi(x) + b$ by solving the following optimization problem.

$$\min_{w,b,\xi_i} \ \tfrac{1}{2} R^2 \|w\|^2 + C \textstyle\sum_i \xi_i \ \text{ subject to } y_i(w^\top \phi(x_i) + b) \geq 1 - \xi_i,\ \xi_i \geq 0 \quad (1)$$

R is the radius of the smallest hyper-sphere covering the feature vectors $\{\phi(x_i)\}$ and $M = 1/\|w\|$ is called the margin, which is the minimum of the distances from the consistently labeled samples to the separating plane. The first term is for minimization of the radius-margin bound $R^2/M^2 = R^2\|w\|^2$ and the second term is for minimization of errors ξ_i and the cost parameter C controls their trade-off. We shall henceforth refer to the optimal objective function value (1) as the *error index*. By duality, the following $\max_\alpha S(\alpha)$ equals to the error index.

$$\max_{\substack{\alpha_i \in [A_i, B_i] \\ \sum \alpha_i = 0}} \quad \underbrace{\sum_i \alpha_i y_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j \frac{K(x_i, x_j)}{R^2}}_{S(\alpha)} \qquad (2)$$

$$\text{where } \ [A_i, B_i] = [\min(y_i\, C, 0), \max(y_i\, C, 0)]$$

By the optimal α^*, the optimal $w^* = \sum \alpha_i^* \frac{\phi(x_i)}{R^2}$ and $f(x) = \sum \alpha_i^* \frac{K(x_i,x)}{R^2} + b$. The samples whose $\alpha_i^* \neq 0$ are called support vectors.

2.2 Support Kernel Machines (SKMs)

In contrast to the SVM which uses a given single kernel $K(x,z)$, the SKM finds the SVM with the least error index whose kernel is a linear combination of given M kernels $K(x,z;\beta) = \sum_{k=1}^{M} \beta_k K_k(x,z)/R_k^2$. We refer $K_k(x,z) = \phi_k(x)^\top \phi_k(z)$ as the k-th component kernel and $\phi_k(x)$ as the k-th feature mapping. R_k and c_k is the radius and the center of the smallest sphere covering k-the feature space $\{\phi_k(x_i)\}$ respectively. β_k is the fractional rate for the k-th kernel ($\beta_k \geq 0$, $\sum \beta_k = 1$). We also refer $K(x,z;\beta)$ as a composite kernel and its corresponding feature vector $\phi(x;\beta)$ is $\phi(x;\beta) = (\ldots, \sqrt{\beta_k}\phi_k(x)/R_k, \ldots)$.

$\|\phi(x_i;\beta) - c(\beta)\|^2 = \sum_k \beta_k \frac{\|\phi_k(x) - c_k\|^2}{R_k^2} \leq 1$ when $c(\beta) = (\ldots, \sqrt{\beta_k}\, c_k/R_k, \ldots)$ and the equality holds when β is a unit vector. Hence approximating the radius of the smallest sphere covering $\{\phi(x_i;\beta)\}$ by 1 for all β, the SKM solves the following min-max problem (3) (minimization of the error index of the SVM with kernel $K(\cdot,\cdot;\beta)$) or the dual max-min problem (4) [1].

$$\min_{\substack{\beta_k \geq 0 \\ \sum \beta_k = 1}} \max_{\substack{\alpha_i \in [A_i, B_i] \\ \sum \alpha_i = 0}} S(\alpha;\beta) \tag{3}$$

$$\max_{\substack{\alpha_i \in [A_i, B_i] \\ \sum \alpha_i = 0}} \min_{\substack{\beta_k \geq 0 \\ \sum \beta_k = 1}} S(\alpha;\beta) \tag{4}$$

$$\text{where } S(\alpha;\beta) = \sum_i \alpha_i y_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j K(x_i,x_j;\beta) = \sum_k \beta_k S_k(\alpha)$$

$$S_k(\alpha) = \sum_i \alpha_i y_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j \frac{K_k(x_i,x_j)}{R_k^2}$$

$S_k(\alpha)$ is the objective function of the SVM with the k-th component kernel.

SKM problem (3),(4) is an SVM problem w.r.t. α and a linear programming (LP) w.r.t. β and hence the optimal α^*,β^* are sparse. The samples whose $\alpha_i^* \neq 0$ and the kernels whose $\beta_k^* > 0$ are called the support vectors and the support kernels respectively. The optimal separator $f(x) = \sum_k \beta_k^* \sum_i \alpha_i^* \frac{K_k(x_i,x)}{R_k^2} + b$. Thus, SKM can extract only critical samples and kernels for classification from given samples and component kernels.

Because $S(\alpha^*;\beta^*) = \sum_k \beta_k\, S_k(\alpha^*) = \min_k S_k(\alpha^*)$ [1] and $S_k(\alpha^*) \leq \max_\alpha S_k(\alpha)$, the error index of the SKM is less than that of any SVM with a component kernel. Consequently the SKM can be expected to have higher precision than the SVM with any component kernel but there remain possibilities that the precision of an SVM with some single component kernel exceeds than that of the SKM because the SKM minimizes the error index but not the generalization error itself.

[1] $S(\alpha^*;\beta^*) = \max_\alpha \min_k S_k(\alpha) = \min_k S_k(\alpha^*)$ because the inside of (4) is the LP-formulation of $\min_k S_k(\alpha)$. β^* can be determined because it satisfies $\sum \beta^* S_k(\alpha) \leq S(\alpha^*;\beta^*)$ for all α.

2.3 Batch Solution Algorithm of SKM

SKM problem (3) is equivalent to the following semi-infinite LP (SKM-ILP).

$$\min_{\substack{\theta,\beta\geq 0\\ \sum_k \beta_k=1}} \theta \text{ s.t. } \sum_k \beta_k S_k(\alpha) \leq \theta \text{ for all } \alpha \in \{\alpha | \alpha_i \in [A_i, B_i], \sum_i \alpha_i = 0\} \quad (5)$$

SKM-ILP can be solved by repeating the following steps starting from the initial constraint set $\mathcal{CS} = \{\beta_k \geq 0, \sum_k \beta_k = 1\}$ and some α^0,β^0 [6].

S1. (approximate SVM solution) find α^t s.t. $S(\alpha^t; \beta^{t-1}) > S(\alpha^{t-1}; \beta^{t-1})$.
S2. (update constraints) add constraint $\sum_k \beta_k S_k(\alpha^t) \leq \theta$ to the constraint set $\mathcal{CS}$ of LP.
S3. (LP) get β^t by solving the LP problem $\min_{(\theta,\beta)\in\mathcal{CS}} \theta$.

The tightest constraint $\mathcal{CS}$ for β^t is obtained when α^t is the exact SVM solution $\alpha = \operatorname{argmax} S(\alpha; \beta^{t-1})$. The optimal SKM solution (α^*, β^*) is obtained when $\theta_0^t = \max S(\alpha; \beta^{t-1})$ becomes θ^t.

3 SKM-Based Active Learning Algorithm

3.1 Issue of Kernel Selection in Active SVM Learning

The active SVM learning algorithm is as follows:

1: specify a kernel $K(\cdot, \cdot)$.
2: specify an initial set of labeled samples $\mathcal{L}$ containing at least one positive and one negative sample and an initial set of unlabeled samples $\mathcal{U}$.
3: **while** $|\mathcal{L}| \leq N$ **do**
4: obtain the optimal SVM solution α^* and the optimal $f(x)$ using $\mathcal{L}$.
5: select an unlabeled sample i by sampling strategy $\mathcal{SS}$ based on $f(x)$ and obtain its label y_i from an oracle (human expert).
6: $(\mathcal{L},\mathcal{U}) \leftarrow (\mathcal{L} \cup \{i\}, \mathcal{U} - \{i\})$.
7: **end while**

We use the popular MARGIN strategy ($i = \operatorname{argmin}_{i\in\mathcal{U}} |f(x_i)|$) as sampling strategy $\mathcal{SS}$.

In active SVM learning, the selection of kernel $K(\cdot, \cdot)$ has great impact on the performance of both the active learning process and the learned SVM. For example, consider the case that many samples are uniformly scattered in a square and the samples within a circle covering less than a half of the square are labeled as $+1$ and as -1 otherwise. In the active learning, we initially knows only the all inputs $\{x_i\}$ and the labels $\{y_i\}$ of the small initial $\mathcal{L}$. But i from this knowledge, t is usually impossible to guess the shape of the separating boundary and hence to select a proper kernel. In this example, the non-homogeneous quadratic kernel $(x^\top z + 1)^2$ is a good choice and if we select this kernel, the active SVM proceeds effectively and the learned SVM has an almost exact separating boundary. But if we select a wrong kernel which cannot express the circle, for example, the

linear kernel ($x^\top z$), no consistent separating boundary exists and the optimal separator $f(x) = -1$ whose error rate is the area ratio r of the circle to the square. Thus after several sampling in the active learning, $f(x)$ becomes -1 and the separating boundary vanishes and the error rate stays at the area ratio r after that. Therefore the active learning process as well as the learned SVM has poor performance.

As shown in this illustrative example, it is impossible to select the kernel good for active SVM learning before learning without prior knowledge and the selection of improper kernel causes great decrease of performances in active learning.

3.2 Basic Active SKM Learning Algorithm

To solve the above difficulty of pre-selection of proper kernels, we propose the use of SKMs instead of SVMs in active learning.

The basic active SKM learning algorithm differs only in two points from active SVM learning algorithm in 3.1. In line 1, component kernels $\{K_k(\cdot,\cdot)\}_{k=1,\ldots,M}$ is specified in stead of $K(\cdot,\cdot)$ and in line 4, $f(x)$ is obtained from SKM solution (α^*, β^*) in stead of SVM solution α^*.

This basic active SKM learning algorithm automatically finds a composite kernel $K(\cdot,\cdot;\beta)$ fitting to the current labeled data $\mathcal{L}$ and forms a separator $f(x)$. There remain the probabilities that the separating boundary vanishes in the middle of active learning but such possibilities are greatly reduced because all the SVMs with a component kernel cannot form any separating boundary in such case.

However, it takes much time to obtain the optimal (α^*, β^*) because SKM is solved by the repetition of SVM and LP and the computation of the composite kernel takes M times of the computation of a single kernel as mentioned in 2.3.

Therefore, the basic active SKM learning algorithm takes too much response time to select the next sample for labeling after one labeling and is inadequate for the active learning in interactive environments.

3.3 Twho-Phased Active SKM Learning Algorithm

For the active learning in interactive environments, we propose a two-phased active SKM learning algorithm depicted in Algorithm 1 and 2. We use the following approaches.

- follow the SKM-ILP algorithm in 2.3 basically.
- reduce the number of evaluations of $f(x)$ used in MARGIN strategy by a randomized approximation algorithm for maximization.
- use an approximate solution α^t of the SVM problem $\max_\alpha S(\alpha; \beta^{t-1})$ in active learning phase to achieve responsiveness
- optimize the SKM problem completely after active learning to achieve accurate classification.

Algorithm 1. Update Function of SVM's solution α

function update$(\alpha, g_{[k]}, \beta, \alpha\text{-}mode, \mathcal{L})$
if α-$mode$ = full **then**
  //SVM solution
  repeat
    $(\alpha, g_{[k]}) \leftarrow \text{update}(\alpha, g_{[k]}, \beta, \text{wss2}, \mathcal{L})$.
    $g_s \leftarrow \sum \beta_k \, g_{[k]s}, \forall s \in \mathcal{L}$.
    $(i, j) \leftarrow (\text{argmin}\{g_s | \alpha_s > A_s, s \in \mathcal{L}\}, \text{argmax}\{g_s | \alpha_s < B_s, s \in \mathcal{L}\})$
  until $(g_i - g_j \leq \tau)$
else
  $g_s \leftarrow \sum \beta_k \, g_{[k]s}, \forall s \in \mathcal{L}$.
  $(i, j) \leftarrow (\text{argmin}\{g_s | \alpha_s > A_s, s \in \mathcal{L}\}, \text{argmax}\{g_s | \alpha_s < B_s, s \in \mathcal{L}\})$
  if $g_i - g_j > \tau$ **then**
    if α-$mode$ = wss1 **then**
      $\lambda \leftarrow \text{argmax}\{S(\alpha + \lambda\,(e_i - e_j), \beta) | \alpha + \lambda(e_i - e_j) \in [A, B], \lambda > 0\}$.
    else if α-$mode$ =wss2 **then**
      $(\lambda, j) \leftarrow \text{argmax}\{S(\alpha + \lambda\,(e_i - e_p), \beta) | \alpha + \lambda(e_i - e_p) \in [A, B], \lambda > 0,\, p \in \mathcal{L}\}$.
    end if
    $\alpha \leftarrow \alpha + \lambda(e_i - e_j)$.
    $g_{[k]s} \leftarrow g_{[k]s} - \lambda\,(K_k(x_i, x_s) - K_k(x_j, x_s))$, $\forall s \in \mathcal{L}, k = 1, \ldots, M$.
    $g_s \leftarrow \sum_k \beta_k g_{[k]s} \; \forall s \in \mathcal{L}$.
    $(i, j) \leftarrow (\text{argmin}\{g_s | \alpha_s > A_s, s \in \mathcal{L}\}, \text{argmax}\{g_s | \alpha_s < B_s, s \in \mathcal{L}\})$
  end if
end if
return $(\alpha, g_{[k]})$.

Reduction of the Number of Evaluations of $f(x)$ in Sampling. To select the best sample using MARGIN strategy in the *select* function in line 11 in Algorithm 2, we need to evaluate $f(x_i)$ of all unlabeled samples $x_i \in \mathcal{U}$D However, when the number of unlabeled samples $|\mathcal{U}|$ is large, these evaluations take very long time. To keep the system's responsiveness, we use a randomized approximation algorithm for minimization. We select the best sample in randomly selected n samples instead of all unlabeled samples. The probability whose the minimum of the random n samples is less than the $100 \times p$-percentile of $\{f(x_i)\}_{i \in \mathcal{U}}$ is $1 - (1 - p)^n$. For example, the probability that the minimum under the 5-percentile is about 92% $(= 1 - 0.95^{50})$ and the probability that the minimum under the 10-percentile is about 99.5% $(= 1 - 0.90^{50})$. Thus we can select a nearly optimal sample within a constant time without depending on the number of unlabeled samples $|\mathcal{U}|$.

Approximation Methods of SVM Solutions. The update function (Algorithm. 1) obtains an approximate solution α^{new} of the SVM problem $\max_\alpha S(\alpha; \beta)$ satisfying $S(\alpha^{new}; \beta) > S(\alpha; \beta)$ according to the given approximation mode α-$mode$. There are three approximation modes $full$, $wss1$ and $wss2$. We use wss1- or wss2-mode in active learning phase and use full-mode in complete optimization phase. τ controls the level of suboptimality of the solution in 'full'.

Algorithm 2. Two-pahsed Active SKM (toplevel)

function two-phased-SKM($\mathcal{L}$,$\mathcal{U}$, $\{(x_i, y_i)\}$)
//Initialize t, constraint set $\mathcal{CS}$, kernel weight β
$t \leftarrow 0$. $\mathcal{CS} \leftarrow \{\beta_k \geq 0, \sum \beta_k = 1\}$. $\beta^0 \leftarrow (1/M, \cdots, 1/M)$.
//Learn the initial labeled samples $\mathcal{L}$ using kernel $K(\cdot, \cdot; \beta^0)$.
$\mathcal{L}0 \leftarrow \{\}$.
for all $s \in \mathcal{L}$ **do**
  $\mathcal{L}0 \leftarrow \mathcal{L}0 \cup \{s\}$. $\alpha_s^0 \leftarrow 0$.
  $g_{[k]s}^0 \leftarrow y_s - \sum_{i \in \mathcal{L}0} \alpha_i^0 K(x_i, x_s; \beta^0)$.
  $(\alpha^0, g_{[k]}^0) \leftarrow$ update$(\alpha^0, g_{[k]}^0, \beta^0, \alpha\text{-}mode, \mathcal{L}0)$.
end for
//Active Learning Phase
repeat
  $b^t \leftarrow$ intercept$(g_{[k]}^t, \beta^t, \mathcal{L})$
  $i_{t+1} \leftarrow$ select$(\alpha^t, \beta^t, b^t, \mathcal{U})$. //MARGIN strategy
  $(\mathcal{L}, \mathcal{U}) \leftarrow (\mathcal{L} \cup \{i_{i_{t+1}}\}, \mathcal{U} - \{i_{t+1}\})$.
  $g_{[k]i_{t+1}} \leftarrow y_{i_{t+1}} - \sum_{i \in \mathcal{L}} \alpha_i K(x_i, x_{i_{t+1}}; \beta^t)$.
  repeat
    $t \leftarrow t + 1$.
    //appriximate SVM solution α^t
    $(\alpha^t, g_{[k]}^t) \leftarrow$ update$(\alpha^{t-1}, g_{[k]}^{t-1}, \beta^{t-1}, \alpha\text{-}mode, \mathcal{L})$.
    $S_k^t \leftarrow S_k(\alpha^t), k = 1, \ldots, M$
    $\theta_0^t \leftarrow \sum \beta_k^{t-1} S_k^t$.
    $CS \leftarrow CS \cup \{\sum_k \beta_k S_k^t \leq \theta\}$.
    //LP's solution $\theta^t = S(\alpha^t; \beta^t) \leq \theta_0^t$
    $(\theta^t, \beta^t) \leftarrow$ argmin$\{\theta | (\theta, \beta) \in CS\}$.
  until (β-$mode$ = once) or ($\theta_0^t > 0$ and $|1 - \theta^t/\theta_0^t| \leq \epsilon_1$)
until $|\mathcal{L}| \geq N$
//Complete Optimization Phase
repeat
  $t \leftarrow t + 1$.
  //SVM solution $\alpha^t =$ argmax $S(\alpha; \beta^{t-1})$
  $(\alpha^t, g_{[k]}^t) \leftarrow$ update$(\alpha^{t-1}, g_{[k]}^{t-1}, \beta^{t-1}, \text{full}, \mathcal{L})$.
  $S_k^t \leftarrow S_k(\alpha^t), k = 1, \cdots, M$
  $\theta_0^t \leftarrow \sum \beta_k^{t-1} S_k^t$. //$\theta_0^t = S(\alpha^t; \beta^{t-1})$
  $CS \leftarrow CS \cup \{\sum_k \beta_k S_k^t \leq \theta\}$.
  //LP solution $\theta^t = S(\alpha^t; \beta^t)$
  $(\theta^t, \beta^t) \leftarrow$ argmin$\{\theta | (\theta, \beta) \in CS\}$.
until $\theta_0^t > 0$ and $|1 - \theta^t/\theta_0^t| \leq \epsilon_0$

function intercept$(g_{[k]}, \beta, \mathcal{S})$
$g_s \leftarrow \sum_k \beta_k g_{[k]s}$ $\forall s \in \mathcal{S}$.
$(i, j) \leftarrow$ (argmin$\{g_s | \alpha_s > A_s, s \in \mathcal{S}\}$, argmax$\{g_s | \alpha_s < B_s, s \in \mathcal{S}\}$)
$(b, \delta) \leftarrow ((g_i + g_j)/2, g_i - g_j)$.
return b.

$(i,j) = (\mathrm{argmin}\{g_i|\alpha_i > A_i\}, \mathrm{argmax}\{g_j|\alpha_j < B_j\})$ is called the most violating pair where gradient $g_i = \frac{\partial}{\partial \alpha_i} S(\alpha;\beta) = y_i - \sum \alpha_j K(x_j, x_i; \beta)$. The intercept b of the separator $f(x)$ satisfies $g_j \leq b \leq g_i$ and in the strictly optimal solution, the difference of the gradients of the most violating pair, $\delta = g_i - g_j$, is equal to 0.

In wss1-mode, we improve α by analytically solving the following subproblem in λ [2] for the most violation pair (i,j)[3],

$$\max_{\lambda} S(\alpha + \lambda(e_i - e_j); \beta) \text{ subjective to } \alpha + \lambda(e_i - e_j) \in [A, B] \tag{6}$$

In wss2-mode, we obtain the optimal λ as well as the optimal sample coupling with one of the most violating pair (i,j), which is fixed in case of wss1.

In full-mode, we obtain (sub-)optimal α by repeatedly updating α using wss1 or wss2 until δ becomes less then a given small positive τ like the SMO-type algorithm [3],

Top-level function of two-phased active SKM learning. The top-level function of the two phased active SKM learning algorithm (Algorithm 2) basically follows the solution algorithm of SKM-ILP in 2.3 but it has the active learning phase and complete optimization phase after the initialization phase.

In active learning phase, there are 3 control variables (α-*mode*, β-*mode*, n) to obtain an approximate SVM solution α^t satisfying $S(\alpha^t; \beta^{t-1}) > S(\alpha^{t-1}; \beta^{t-1})$. n is the number of random samples for selection samples, α-*mode* is the approximation mode of α described in the previous subsection. β-*mode* controls whether the update of α and β executes once (once) or repeatedly until satisfying the convergence conditions (full). In the complete optimization phase, we optimize the SKM completely. The different setting of the SKM convergence conditions ϵ_1 and ϵ_0 is allowed because it can take too much response time in active learning when ϵ_1 is set to the value as small as ϵ_0 in case of β-*mode*=full.

The basic active SKM learning algorithm in 3.2 is the case when the mode (α-*mode*, β-*mode*, n)=(full, full, ∞) and it completely optimizes SKM every time even in the active learning phase. Sonnenburg[5,6] proposes a batch algorithm called 'chunked MKL' to solve SKM based on SKM-ILP formulation in which use the optimal solution for the partial data as α^t. This algorithm is similar to the case of mode (full, once, ∞) except that a sampling strategy SS for batch learning is used. In contrast, the proposed two-phased active SKM uses in active learning phase (wss1, once, n) or (wss2, once, n) which gives more rough approximation of the optimal SVM solution to achieve the responsiveness.

4 Experiments

We conducted the following experiments to compare the performances of active SKM learning and active SVM learning, and the effects on error rates and response time of the modes in two-phased active SKM learning algorithm.

[2] Because $S(\alpha + \lambda(e_i - e_j)) = S(\alpha) + (g_i - g_j)\lambda - \frac{1}{2}\lambda^2 D_{i,j}^2$,
$D_{i,j}^2 = K(x_i, x_i) + K(x_j, x_j) - 2K(x_i, x_j)$, $\lambda = \min\{(g_i - g_j)/D_{i,j}^2, \alpha_i - A_i, B_j - \alpha_j\}$.

4.1 Experimental Settings

We use our code written in MATLAB, which solves LP by *linprog* function built in MATLAB and caches the kernel matrix $[K(x_i, x_j)]_{i,j\in\mathcal{L}}$, $[K_k(x_i, x_j)]_{i,j\in\mathcal{L}}$ for the labeled samples $\mathcal{L}$ to avoid the recalculation. $[K(x_i, x_j)]$ is recalculated from $[K_k(x_i, x_j)]$ when β is updated, R_k^2 is calculated before starting active learning by the Support Vector Data Description (SVDD)[7] for all the data $\mathcal{L} \cup \mathcal{U}$.

Table 1 shows the features of the 5 benchmark dataset[3] we used. Each dataset has 100 pairs of training and test data and we use the 1st to 5th pairs for our experiments. The tuned SVM in Table 1 is the average error rate for the 1st to 5th pairs of SVMs with tuned C and γ.

Table 1. Benchmark Datasets

	number of data		error rate(%)	
dataset	learning	test	tuned SVM	active SKM
banana	400	4900	11.6	12.3
breast	200	77	28.0	27.5
flare	666	400	35.9	32.0
image	1300	1010	3.0	2.8
splice	1000	2175	10.8	10.6

Table 2. Modes in two-phased SKM learning algorithm

name	α-mode	β-mode	n
basic	full	full	∞
fullfull	full	full	50
full	full	once	50
wss1	wss1	once	50
wss2	wss2	once	50

In the experiments, we conduct the active SVM or SKM learning for the training data in each pair in a dataset using the 15 RBF kernels $\exp(\gamma \|x - z\|^2)$ where $\gamma \in (2, , 5, 10) \times (10^{-2}, 10^{-1}, 1, 10, 10^2)$ and the cost parameter $C = 1000$, the tolerance $\tau = 10^{-3}$, $\epsilon_1 = 10^{-2}$ in active learning phase and $\epsilon_2 = 10^{-6}$ in the complete optimization phase. We recorded the response time and the error rate for the corresponding test data when the number of labeled samples becomes $1, 2, \ldots, 49, 50, 60, \ldots, 90, 100, 150, \ldots, 450, 500, 600, \ldots, 900, 1000$. The error rate of active SKM when the number of labeled samples is k is that of the completely optimized SKM when the k samples are labeled. In the following evaluation, we use the average of the error rates or the response time of the 1st to 5th pairs in the dataset.

4.2 Active SVM Versus Active SKM

For the 5 datasets, we compared the changes of error rates as the progress of the sampling among the basic active SKM learning algorithm with the 15 RBF kernels and the active SVM learning using each RBF kernel. The first column in Figure 1 shows the result. The solid, dotted, dashed line is the error rate of the active SKM, the best SVM whose final error rate is the smallest and the worst

[3] http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm. The tuned parameters C and γ for each dataset are shown and we recalculate the average error rate of the tuned SVM in Table 1 from the *errors* file in the site.

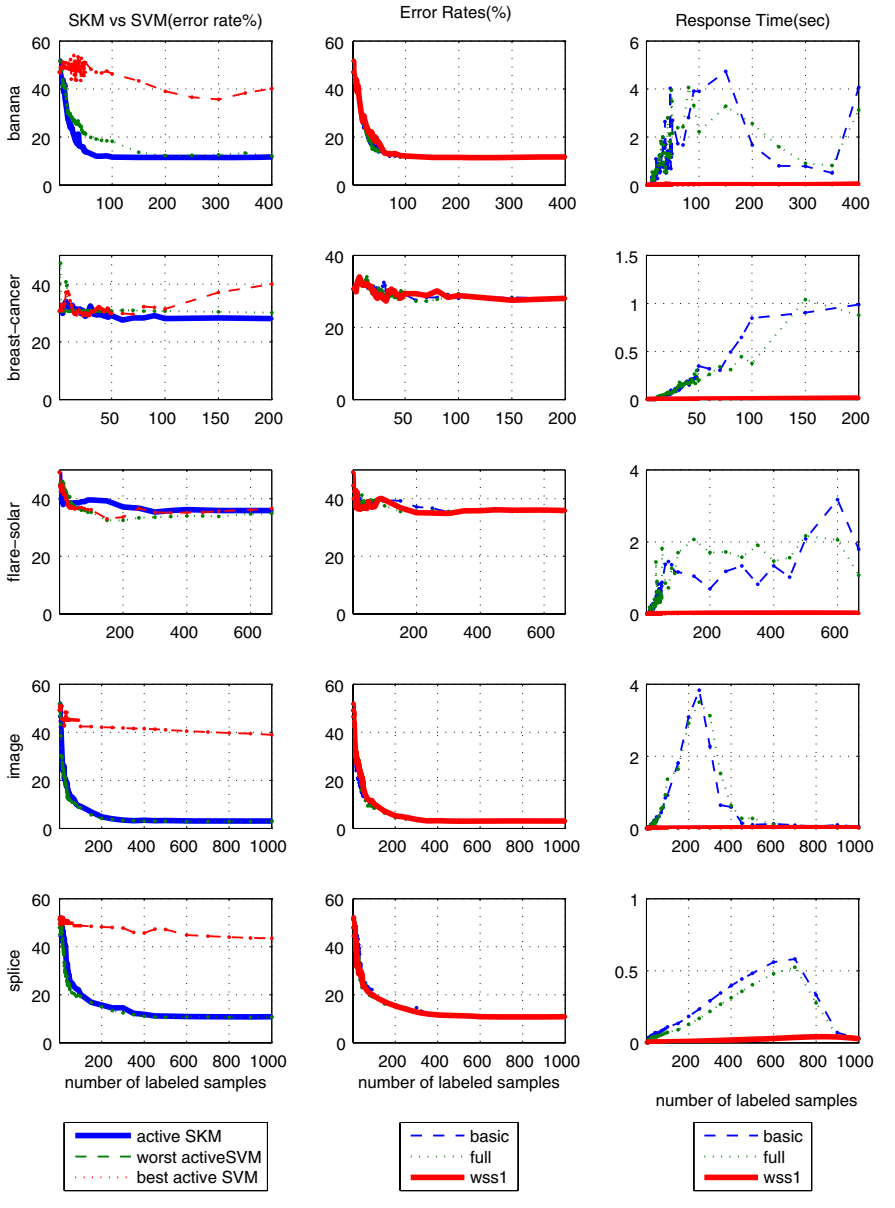


Fig. 1. Experimental Result

SVM whose final error rate is the largest respectively. In the datasets of banana, image, splice, the difference of the error rates of the best SVM and the worst SVM are greater than 20% and hence the selection of kernel greatly affects the performance of active SVM. Each active SKM have the compatible performance with the best SVM for all datasets. We show the final error rates of active SKMs

in Table 1. We fixed the C to 1000 in the experiments but the active SKMs have similar performance to the SVMs with tuned C and γ. For these result, we think the proposed active SKM learning algorithm is a versatile and powerful active learning methods.

4.3 Modes of Two-Phased SKM

In the section, we compare the error rate and the response time of the two-phased SKM with different modes in Table 2. We show only "basic", "full", "wss1" in the 2nd and 3rd column of Figure 1 because the results of fullfull and wss2 are almost same with full and wss1 respectively.

The error rate are almost the same for the modes, while the response time of the wss1 and wss2 is spontaneous (less than 0.2 sec) but those of basic and fullfull, full are over several seconds (8 secs.) in cases.

From these results, the two-phased SKM learning with mode wss1 or wss2 has high responsiveness proper for active learning interacting with human experts and also can learn an accurate classifier.

5 Conclusions

As we showed in the experiments, the performance of the active SVM learning greatly depends on the kernel selected before learning. However, the selection of an effective kernel is impossible before active learning without prior knowledge. Therefore in this paper, we propose the active SKM learning and showed the active SKM with 15 RBF kernels can perform the active learning comparable with the best SVM learning for the popular 5 benchmark datasets. We also propose the two-phased active SKM learning having responsiveness necessary for the interaction with human experts and showed that the response time during the active learning can be greatly reduced without sacrificing the precision of the learned machine. From these result, the proposed two-phased SKM learning is a versatile and powerful active learning method which can learn the highly precise separators and the labels effectively for a wide range of datasets and has the responsiveness necessary for the interaction with human experts.

References

1. Bach, F.R., Lanckriet, G.R.G., Jordan, M.I.: Multiple kernel learning, conic duality, and the smo algorithm. In: Proc. of 21st International Conference of Machine Learning (2004)
2. Campbell, C., Cristianini, N., Smola, A.: Query learning with large margin classifiers. In: Proc. of 17th International Conference on Machine Learning, pp. 111–118 (2000)
3. Platt, J.: Fast training of support vector machines using sequential minimal optimization. In: Advances in Kernel Methods, pp. 185–208. MIT Press, Cambridge (1998)

4. Schohn, G., Cohn, D.: Less is more: Active learning with support vector machines. In: Proc. of 17th International Conference on Machine Learning, pp. 839–846 (2000)
5. Sonnenburg, S., Raetsch, G., Schaefer, C., Schoelkopf, B.: Large scale multiple kernel learning. Journal of Machine Learning Research 7, 1531–1565 (2006)
6. Sonnenburg, S., Ratsch, G., Schafer, C.: A general and efficient multiple kernel learning algorithm. In: Advances in Neural Information Processing Systems, vol. 15. MIT Press, Cambridge (2006)
7. Tax, D.M.J., Duin, R.P.W.: Data domain description using support vectors. In: Proc. of 7th European Symposium on Artificial Neural Networks, pp. 251–256 (1999)
8. Tong, S., Koller, D.: Support vector machine active learning with applications to text classification. Journal of Machine Learning Research 2, 45–66 (2001)
9. Vapnik, V.: Statistical Learning Theory. John Wiley and Sons Inc., Chichester (1998)

Extracting Topic Maps from Web Pages

Motohiro Mase[1], Seiji Yamada[2], and Katsumi Nitta[1]

[1] Tokyo Institute of Technology, Japan
[2] National Institute of Informatics, Japan

Abstract. We propose a framework to extract topic maps from a set of Web pages. We use the clustering method with the Web pages and extract the topic map prototypes. We introduced the following two points to the existing clustering method: The first is merging only the linked Web pages, thus extracting the underlying relationships between the topics. The second is introducing weighting based on similarity from the contents of the Web pages and relevance between topics of pages. The relevance is based on the types of links with directories in Web sites structure and the distance between the directories in which the pages are located. We generate the topic map prototypes from the results of the clustering. Finally, users complete the prototype by labeling the topics and associations and removing the unnecessary items. For this paper, at the first step, we mounted the proposed clustering method and extracted the prototype with the method.

Keywords: Web information extraction, Topic Maps, clustering.

1 Introduction

Information gathering that references a huge amount of Web pages is very useful and essential for Web users. However, finding the necessary information from the Web when users need it and organizing the gathered information are both big problems [6]. There are many works currently underway to solve these problems, and one of them is Topic Maps [7]. Topic Maps are an international standard for organizing and classifying information along with a user's knowledge and concepts. Topic maps are composed of three elements; topics, associations, and occurrences. This standard can connect the various information resources with the knowledge and concepts of the user, represent the relations between the concepts, and help users more easily access the information they need. It takes the information resources and the selection of target domains and topics to build topic maps. The user needs to manually do these tasks, although the user can reduce costs by using editors for the topic maps, such as Ontopoly [1]. By converting the existing metadata, such as XML and RDF, previous studies have been able to automatically generate topic maps [9][12]. Although the amount of structured metadata on the Web is growing, many of the Web pages that users utilize on a daily basis are semi-structured data and HTML files, and using

[1] http://www.ontopia.net/solutions/ontopoly.html

S. Chawla et al. (Eds.): PAKDD 2008 Workshops, LNAI 5433, pp. 169–180, 2009.

the previous method with them is difficult. Therefore, this process requires a method to directly extract topic maps from semi-structured data. However, we aim to extract topic map prototypes from the Web pages by using the clustering method, because it is difficult to automatically extract complete topic maps. Then, users finally complete the topic map prototypes by evaluating the topics and associations, and by adding and removing them.

In this paper, we at first propose a method for extracting a topic map prototype from a set of Web pages and conduct an experiment to evaluate the extracted topic maps. By utilizing the topic map that is extracted from the Web pages in the user's browsing history and neighboring pages, which the user has not visited, a user can easily manage the collected information and then access the information that they have never seen before.

The contents of Web pages and structures of Web graphs that consist of nodes as pages and edges as links contain the underlying topics of pages' contents and relationships between the topics. Our approach is to extract the involved information as topic maps by clustering the Web pages by the contents of their pages and the structures of the Web graphs. In this field, many previous works have studied on the extraction of information from Web communities from the structures of the Web graphs. Broder et al. extracted the communities by searching a complete bipartite graph which is a community signature [2]. Flake et al. found the communities from the Web by using a maximum flow algorithm [3]. Girvan and Newman extracted the structures of the communities within networks by using a clustering method based on the edge betweenness, which is the number of shortest paths between all pairs of vertices that run through it [5]. Newman proposed a hierarchical clustering method that maximizes the modularity[11], which is a function to quantify how good a particular division is: we call the method the Newman's method. These preliminary works focused only on the extraction of structures from the Web graphs. Although our method is based on Newman's method, the method uses the similarity between the contents of the Web pages and the relevance between the topics of pages which is based on the structures of the Web site's directories in addition to the structures of the Web graphs, to extract topic maps.

2 Topic Maps

Topic Maps is an ISO/IEC 13250 standard and a solution for representing a concept and connecting the concept to its related information resources [7]. A topic map is composed of topics, which represent any concept or subject in the real world; associations, which represent the relation between topics; and occurrences which represent the connection between topics and the information resources related to them. Topic maps have a lot of flexibility, and allow their creators to define the types of topics, associations and occurrences, and good for representing various topics and relationships concerning them on the Web. One of the syntaxes of Topic Maps is XML Topic Maps (XTM) [15]. Some information items and named properties are essential for representing topic maps using XTM.

We extract only the items concerning the topics, associations, and occurrences, because it is hard to extract all the items and properties from Web pages. We were able to get three types of items from the results of using our clustering method on a set of Web pages, by assuming the clusters were topics, the edges were associations, the pages related to the clusters were occurrences, and then generate prototypes of the topic maps with the extracted items. Finally, users complete the prototype by labeling the topics and associations and removing the unnecessary items.

3 Framework to Extract a Topic Map from a Set of Web Pages

3.1 Overview

We propose the framework to extract a topic map from a set of Web pages using the following procedure.

1. Collect a set of Web pages from the user's Web history.
2. Use the clustering method on the set of pages.
3. Extract the items of the topic map from the result of clustering and build a topic map prototype
4. Finally, a user completes the extracted prototypes by evaluating, labeling, and modifying the items.

In this paper, we propose a method to extract the topic map prototypes. Topic maps have to show not only the topics that are found in the Web pages, but also the relationships between them. However, existing contents-based clustering methods [8] can measure the similarities between the topics that the clusters represent by using the contents of the Web pages, but not extract what the relationship between the topics is. In contrast, structured-based clustering methods focus only on the structures of networks without any reference to the contents of the pages. We proposed a clustering method based on Newman's method, structure-based clustering methods, regarding both the similarities between the contents of Web pages and the relevance between the topics of pages based on the graph structures of links on Web.

Structure of links between Web pages. In general, site creators manually generate links on the Web, and the linked pages cover the relevant topics [10]. These links have underlying relations between the topics. However, even if a creator link pages that have relevant topics and the links represent some relation to the topics, the contents-based methods can't find the relation for lower value similarities for the pages' contents. We cluster pages by merging only the linked pages and extract the relations from the remaining links, which are represented as the edges between clusters, at the end of clustering.

Types of links with directories in Web sites structure. We build denser clusters, taking into consideration the weights of links between Web pages. Utilizing the similarities between the contents of linked pages is one of the calculation

methods for the weights. In contrast, we use the relevance between the topics of the Web pages based on the types of links.

The relevance between the topics based on the types of links are computed by grouping the links with directories in which linked pages exist on a Web site and using the distance between the directories, without focusing on the contents of the pages. We presume that the creators of Web pages probably make directories on Web sites and locate pages in the directories as follows: 1) Web pages are classified into directories along with the topics of pages. 2) The topics of pages in child directories are a specialization of pages' topics in their parent directories. 3) The topics of pages are similar to them in closer directories than distant directories. We compute the relevance from the previously mentioned estimation. As for related research, Spertus categorized the types of links with hierarchical relationships of directories in which linked pages exist, and introduced heuristics to estimate the meanings of the links [14].

To estimate the relevance between the topics of the Web pages using the types of links, we sorted all the links between pages to three types listed by the directories in which the pages are located, and introduced a measure, the distance of the directories. The measure is the number of directories on the shortest path between any two pages, which we put in a tree structure that consists of pages and directories used as nodes.

1. **upward/downward** is a link between a page and it in a higher or lower directory on the same Web site.
2. **crosswise** is a link between the pages on the same Web site, except for the upward/downward links.
3. **outward** is a link between pages on two different Web sites.

We calculated the relevance by giving priority to the types of links as follows.

- We prioritize by assigning crosswise links over upward/downward ones. Since the creators of pages sort them into directories by content, the topics of pages reached through upward/downward links are specialized or generalized topics and are uncorrelated with the topics of pages in the same directory.
- We assign priority to the outward links over the upward/downward ones, because the outward links represent pointers to related information resources on other sites made by the creators of them, and linked pages with outward links have mutual topics.
- We prioritize links that have lower values for the distances of their directories. Pages in the same directory have similar topics and the topics of the pages are poorly correlated as the distances between the directories in which the pages are involved increase.

With this in mind, we compute the relevance by the types of links. We generate concentrated clusters by weighting based on the similarities between the contents of the Web pages and the relevance between the topics of the pages by the types of links. As for the details, we note them in the next section.

3.2 Weighting Links between Web Pages

For the weight between Web pages p and q, $w(p,q)$ is computed as a weighted liner sum of the similarity between contents of the pages, $s_c(p,q)$, and the relevance between the topics of pages by the types of links, $s_l(p,q)$. $w(p,q)$ is defined as follows:

$$w(p,q) = \alpha s_c(p,q) + (1-\alpha) s_l(p,q) \tag{1}$$

Where, at this time, the value of α is 0.5. We aim to generate dense clusters by using the relevance between the Web pages' topics which is unable to calculate with the similarity between the contents of the pages in addition to the similarity. Then, we evenly use the weighting based on values of the similarities and the relevance.

Similarity between contents of pages. We construct a document vector of a page by applying the TF-IDF method [13] to all the words extracted from the page's HTML file. The document vector v_i is $v_i = (w_{i1}, w_{i2}, ..., w_{in})$, where w_{ij} is the tfidf value of word j in page i. The similarity between pages p and q, $s_c(p,q)$ is defined as follows:

$$s_c(p,q) = \frac{v_p \cdot v_q}{\parallel v_p \parallel \parallel v_q \parallel} \tag{2}$$

Relevance between the topics of pages by types of links. We calculate the relevance between the topics of pages based on the types of links by using the relationships between the directories of the linked pages and the distances of the directories, along with the line of weighting shown in Section 3.1. The relevance between topics of pages p and q, $s_l(p,q)$ is defined as follows:

$$s_l(p,q) = \begin{cases} \dfrac{0.25}{d} C_l & \text{(upward/downward)} \\ \dfrac{0.5}{d} C_l & \text{(crosswise)} \\ 0.4\, C_l & \text{(outward, } \tau_s \leq s(p,q)\text{)} \\ 0 & \text{(outward, } s(p,q) < \tau_s\text{)} \end{cases} \tag{3}$$

where C_l is decided by the direction of the link. When the link between the pages is cross-linked, $C_l = 2$. The value of C_l for a single link is 1, $C_l = 1$. d is the distance between the directories of the pages, as has been previously described. τ is average of all the values of $s_c(p,q)$, which are the similarities between the pages that are connected by an "outward" link.

3.3 Algorithm

To generate topic maps, we use the clustering method for the Web pages and extract the items for the topic maps from the results of the clustering. Our

clustering method is based on Newman's method, although Newman's method only focuses on the structure of the network and we introduce the weighting for the links that were introduced in Section 3.1 to Newman's method. We calculate the value of the weight in the way hereinafter prescribed. Newman's method [11] is the agglomerative hierarchical clustering method and has the modularity Q, which shows the quality of a cluster division. In the clustering in Newman's method, a pair of clusters that provide the maximal value of increment in Q are joined at each step. The definition of Q is as follows:

$$Q = \sum_i (e_{ii} - a_i^2) \tag{4}$$

$$\Delta Q_{ij} = 2(e_{ij} - a_i a_j) \tag{5}$$

where e_{ij} represents the fraction of edges between clusters i and j in the network, and is calculated as the value for the number of edges between clusters i and j divided by the total number of the edges in the network. a_i is the fraction of edges belonging to cluster i and is denoted as $a_i = \sum_i e_{ij}$. Q is the difference between the fraction of the number of edges within all the clusters in the network and the expected value for the same cluster division in a network whose edges are randomly connected. If the number of edges within the clusters is lower than the random connected edges, we will obtain $Q = 0$. The maximal value of Q is 1; a high value represents a good partitioning of the network.

We cluster Web pages using our method which is based on Newman's method, using above mentioned weights, and according to the following procedure.

1. Set a cluster as a page, and an edge as a link in a given set of Web pages.
2. Calculate all the values of the weight for the links, and apply them to the edges. Each weighted value of an edge is normalized by the total value of the edges.
3. Select a pair of linked clusters that have a maximal value of ΔQ. Terminate if the ΔQ of the selected pair has a negative value.
4. Merge the pair of clusters into a new cluster.
5. Recalculate the values of e_{ij} and a_i, which are related to the new cluster and Go to back to Step 3.

Finally, we extract a topic map prototype from the result of the clustering.

3.4 Extracting and Visualizing Topic Map Prototypes

To generate topic maps, we use the clustering method for the Web pages and extract the items for the topic maps from the results of the clustering. Our clustering method is based on Newman's method, although Newman's method only focuses on the structure of the network and we introduce the weighting for the links that were introduced in Section 3.1 to Newman's method. We calculate the value of the weight in the way hereinafter prescribed. Next, we calculate the weights of links and apply the weights to the links in the set of Web pages. Finally, we cluster the set of the Web pages, which have weighted links, by

Newman's method and generate the topic map prototypes by assuming that the clusters are the topics, the edges are the associations, and the Web pages related to the topics are the occurrences from the results of the clustering. We visualize the topic map prototype on a graph using graphviz [4], which is one of the graph visualization tools. On the graph, the topics are represented as nodes and the associations are represented as edges. The area of a node is dependent on the number of pages related to the topic.

4 Experiments

We conducted experiments to evaluate our proposed method by comparing the topic map prototypes that were generated using our method to those using Newman's method. The comparison with the forms generated by the two methods explains the effect of the introduction of the weighting. Sixteen participants took part in this experiment. Each participant supplied two sets of browsing histories, which were used to collect the sets of Web pages, and also evaluated four topic maps generated from the sets of pages using both methods. The order for evaluating the topic maps was random.

4.1 Extracting Topic Map Prototypes

In this experiment, we used the sets of Web pages that were collected using the browsing histories of the participants as seeds. As these sets of pages consisted of pages related to the browsed pages, they easily evaluated the extracted topic map prototypes. The histories are restricted to a sequence of five pages searching for some sort of issue. The pages in the histories have to be connected to each other, because our method is based on the links between pages. We obtained a set of Web pages by following the links four times from the pages of the histories. There were three selected links in each of the pages and the links were selected at random. The sets of pages contained around 600 pages. We extracted topic map prototypes from these sets of pages using both methods, and visualized them on an interface for evaluation. The interface shows the clusters that have pages in the histories as ringed nodes.

4.2 Evaluation

The topic maps had no labels for the topics and associations when they were extracted. The participants gave appropriate names and annotations to the topics and associations. The interface shows the extracted topic map, representing the topics as nodes and the associations as edges between the topics. By clicking the nodes and edges, the interface shows the information window for the topics and associations that correspond to the nodes and edges. The participants get the information concerning the topics and associations through the window.

Evaluating topics. As the topic of the cluster is a concept represented by a set of Web pages within it, the name of the topic is the name of the concept.

The participants judge what the concept of the set of pages is, and appropriately name the topic, referring to the following information:

- Title, URL, and content of the pages related to the topic.
- Three domain-specific terms.

When the cluster has a number of topics of the Web pages, the participants evaluate the cohesiveness of the topics of the Web pages and select a major one and name it. If the topics in the cluster are vary widely, the participants don't give the proper name to the cluster. The participants also evaluate the granularity of the cluster's topic in terms of results. Then, the granularity of the individual topics in the topic maps varies. We call the topics named properly "valid topics". On other hand, the topics that have no name are called "invalid topics".

Evaluating associations. The associations are described as edges between clusters that present topics, and represent some kind of relationship between the topics. The participants judge what relation between the topics is. If feasible, they give proper annotation to the associations, referring to following information:

- The information concerning the two topics that have an association.
- The information concerning the links between the pages that consist of the edges.

As in the case of the topics, we call the associations that are provided with appropriate annotations, "valid associations". The associations having no annotations are called "invalid associations".

4.3 Experimental Results

We evaluate whether the topics and the associations are valid and whether the individual ones are appropriately extracted. We don't evaluate whether they are useful for the participants. Because the problems of evaluating the utility of the topic and the associations depend on the knowledge and information of the participants, it's hard to evaluate it. Thus, in this experiment, we evaluate the validity of the topic map prototypes by the participants' evaluation for the cohesiveness and the granularity of the individual topics and associations in the prototypes.

We extract topic map prototypes from the set of pages, show them to the users. The users finally have to modify and fix the topic map prototypes, so they can be used as the topic maps. It's preferable that the prototype contains valid topics and associations that are extracted broadly. The tasks for adding insufficient topics and associations is more costly than those for reducing the unnecessary items.

To evaluate the fitness of the prototypes, we evaluate the number of valid topics and associations, especially those regarding the topic maps extracted by

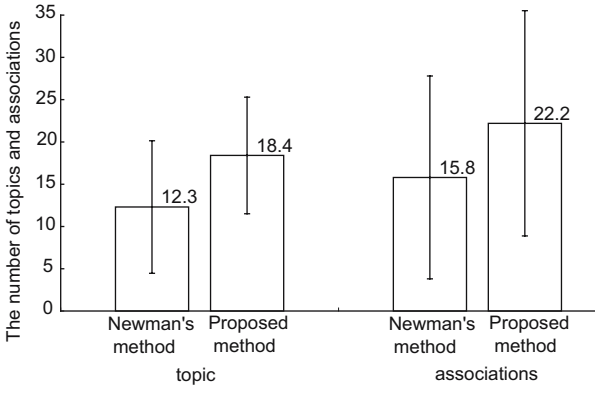


Fig. 1. No. of valid topics and associations

both methods. In this case, recall [1], which is often used to evaluate the performance in the field of IR, is unusable, because the complete topic maps in the sets of pages is not obvious and the topics and associations that are required to be extracted are unclear.

Fig. 1 shows the average number of valid topics and associations. The results showed that the average number of valid topics was 12.3 (standard deviation: 7.84) when using the Newman's method, 18.4 (6.89) for the proposed method, and the same value of valid associations was 15.8 (12) for the Newman's method, but 22.2 (13.22) for our method. Regarding the number of topics, a comparison made using Wilcoxon signed-ranks test showed a significant difference between both methods (p= 0.000012, α= 0.05). The comparison in the number of associations using a paired t-test also showed a significant difference (p= 0.002, α= 0.05). These results show that our proposed method can extract topic maps that have more valid topics and associations than that by the Newman's method.

5 Discussion

5.1 Utilizing Meta Information of Web Pages

The results of the experiments show that our proposed method can extract topic maps that have more valid topics and associations than that by the Newman's method. However, our proposed method weights the links depending on the directories in the Web site structure, so it's hard to accommodate the Web pages in flat directories in the Web site structure and the pages of the news sites are sorted to the directories by date.

To solve the problem, we used tags or categories given to Web pages, in place of the directories sorted by date in the Web sites. We introduced a virtual directory to the URL of the page by using given tags and convert the URL as shown by Fig. 2. We similarly calculated the weights as shown in Section 3 and extracted the topic map prototypes. We experimentally extracted a topic map

Tag : notepc
URL: http://plusd.itmedia.co.jp/pcuser/articles/0711/09/news006.html

Convert URL

URL: http://plusd.itmedia.co.jp/pcuser/articles/notepc/news006.html

Fig. 2. Converting URL

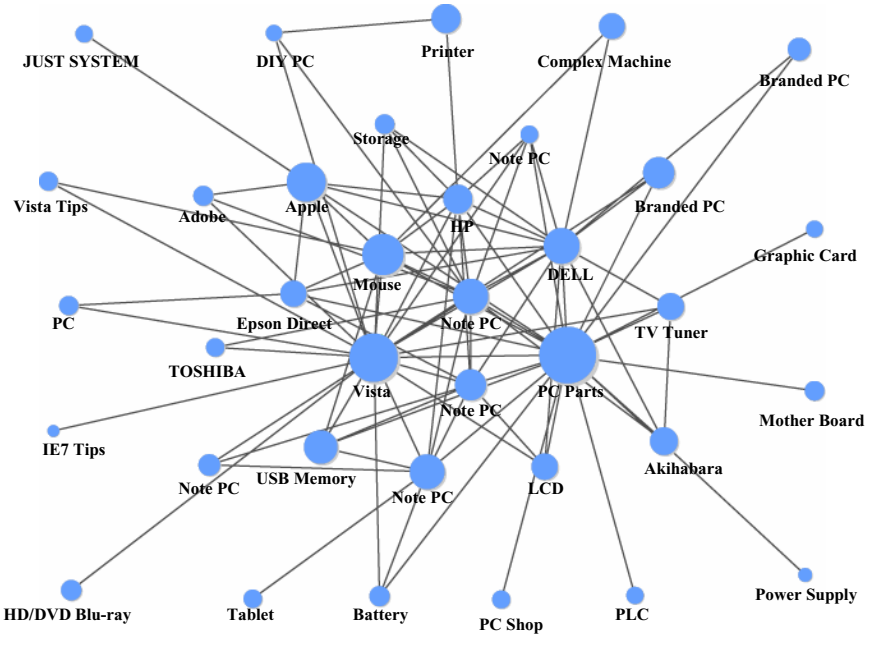


Fig. 3. Topic map from PCUSER

from a set of Web pages that are collected from PCUSER, which is a Japanese news site about PC, by our proposed method with converted URLs. Fig. 3 shows the extracted topic map from the Web pages of PCUSER. The pages of PCUSER have some tags given from 293 tags according to the topics of the pages and are sorted into the some categories. The classification by the tags is too detailed and complicated. We can find the topics presented by the categories in the Web pages of PCUSER, but which the topics are related is unclear.

In contrast, for example, the extracted topic map shown as Fig. 3 shows some topics related to PC and relationships between the topics. We find the topic "PC parts" and the related topics, such as "graphic card", "mother board" and "power supply" around "PC parts". Also, we can find the relationships between "PC parts" and the related topics. The extracted topic map supports a survey of the Web sites.

5.2 Interactive Improvement of Topic Map Prototypes

In this paper, we proposed the method for extracting the topic map prototypes from a set of Web pages. We have to build the interface that enables interactive improvement of the prototypes by user. The interface needs the following functions.

- Editing of topics and associations. User can name the topics and the associations, remove the unnecessary items, add the missing items, and merge the items which have the same name.
- Display of information about the topics and associations. User can name and add the items by referring to the information.
- Support for editing of items. User can spread the changes to the prototypes by an edit once. For example, when user remove an association between topic "A" and "B", the other associations between topic "A" and "B" are automatically deleted.
- Feedback from user's editing to the prototypes. The system regard the Web pages, which are contained in the unnecessary topics removed by the user, as non-relevant documents in document classification and classify the next set of Web . The system previously removes unnecessary Web pages which are classified as the non-relevant documents, and reduce the procedure for removing the unnecessary topics and associations from the user's editing of the prototypes.

In our future work, we will build the interface which has the function to assist user to complete the topic map prototypes.

6 Conclusion

We proposed a framework in this paper to extract topic maps from a set of Web pages. First, we proposed a method to extract the topic map prototypes. Our method was based on Newman's method using a weighting scheme based on the similarities between the contents of pages and the relevance between the topics of the Web pages by the types of links. We conducted experiments to evaluate the proposed method by comparing it to Newman's method. The results of the experiments showed that our method can extract the topic maps that have enough topics and associations with the topic map prototypes. In our future work, we will build the interface for user's interaction to complete the topic map prototypes.

References

1. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley, Reading (1999)
2. Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., Wiener, J.: Graph structure in the web: experiments and models. In: 5th International World Wide Web Conference (2000)

3. Flake, G.W., Lawrence, S., Giles, C.L.: Efficient identification of Web communities. In: KDD 2000: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 150–160 (2000)
4. Gansner, R.E., North, S.C.: An open graph visualization system and its applications to software engineering. Software – Practice and Experience 30(11), 1203–1233 (2000)
5. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. PNAS 99(12), 7821–7826 (2002)
6. GVU's WWW Surveying Team: GVU's 10th WWW User Survey: Problem Using the Web (1998), http://www.gvu.gatech.edu/user_surveys/
7. International Standard Organization: ISO/IEC 13250 Topic Maps: Information Tecknology Document Description and Markup Language (2000)
8. Jain, A.K., Dubes, R.C.: Algorithms for clustering data. Prentice-Hall Inc., Upper Saddle River (1998)
9. Kerk, R., Groschupf, S.: How to Create Topic Maps (2003), http://www.media-style.com/gfx/assets/HowtoCreateTopicMaps.pdf
10. Menczer, F.: Lexical and semantic clustering by web links. Journal of American Society Information Science and Technology 55(14), 1261–1269 (2004)
11. Newman, M.E.J.: Fast algorithm for detecting community structure in networks. Physical Review E 69, 066133 (2004)
12. Reynolds, J., Kimber, W.E.: Topic Map Authoring With Reusable Ontologies and Automated Knowledge Mining. In: XML 2002 Conference (2002)
13. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Information Processing & Management 24(5), 513–523 (1988)
14. Spertus, E.: ParaSite: mining structural information on the Web. In: The 6th International World Wide Web Conference, pp. 1205–1215 (1997)
15. TopicMaps.Org: XML Topic Maps 1.0 (2001), http://www.topicmaps.org/xtm/1.0/

Interactive Abnormal Condition Sign Discovery for Hydroelectric Power Plants

Norihiko Ito[1], Takashi Onoda[1], and Hironobu Yamasaki[2]

[1] System Engineering Research Laboratory,
Central Research Institute of Electric Power Industry,
2-11-1, Iwado Kita, Komae-shi, Tokyo 201-8511, Japan
{norihiko,onoda}@criepi.denken.or.jp
[2] The Power System Engineering Department,
Kyushu Electric Power Co.,Inc.,
2-1-82, Watanabe-Dori, Chuo-ku, Fukuoka 810-8720, Japan
Hironobu_Yamasaki@kyuden.co.jp

Abstract. Kyushu Electric Power Co.,Inc. collects various sensor data and weather information to maintain hydroelectric power plants while the plants are running. However, it is very rare to occur abnormal and trouble condition data in power equipments. And in order to collect the abnormal and trouble condition data, it is hard to construct an experimental hydroelectric power plant. Because its cost is very high. In this situation, we have to find abnormal condition data as a risk management. In this paper, we consider that the abnormal condition sign may be unusual condition data. This paper shows results of unusual condition data of bearing vibration detected from the collected various sensor data and weather information by using one class support vector machine. The result shows that our approach may be useful for unusual condition data detection and maintaining hydroelectric power plants. Therefore, the proposed method is one of risk management for hydroelectric power plants.

Keywords: Data Mining, Abnormal Condition Detection, Support Vector Machine, Hydroelectric Power Plant.

1 Introduction

Recently, electric power companies have begun to try to shift a Time Based Maintenance (hereafter, we use TBM) to a Condition Based Maintenance (hereafter, we use CBM) for electric equipment management to realize an efficient maintenance and reduce the cost of maintenance [1,2]. TBM is to check and change the equipment based on the guaranteed term recommended by makers. And CBM is to check, repair and change the equipment based on the state of equipment [3]. The state consists of the present state of equipment, the operation term of equipment, the load in an operation, and etc [3]. Therefore, this CBM is a kind of risk management for electric power companies' management.

S. Chawla et al. (Eds.): PAKDD 2008 Workshops, LNAI 5433, pp. 181–192, 2009.

In order to realize the CBM, it is important for electric power companies to collect the data of equipment. The data consist of normal condition data of equipment, abnormal condition sign of equipment, and etc. Especially, to reduce the maintenance cost by the improvement of the equipment management and maintenance based on CBM, it is the most important to collect much data utilized to make management and maintenance.

It is necessary to collect and analyze the past abnormal condition sign and the past trouble condition data, in order to discover an abnormal condition sign of power equipment [4,5]. For instance, there are the discovery of an abnormal condition sign from sensor data of the hydroelectric power plant. However, it is very rare to occur abnormal and trouble condition in the power equipments in Japan. Because power companies have changed the power plant equipment in certified term by makers for the conventional maintenance. And in order to collect the abnormal and trouble condition sign, it is hard to construct an experimental hydroelectric power plant. Because its cost is very high.

In the above situation, Kyushu Electric Power Co.,Inc. has been analyzing the causal relation between the bearing vibration and various sensor data to establish the detection method of an abnormal condition sign for the bearing vibration in the hydroelectric power plants. However, it is hard to discover a causal relation between the bearing vibration and various sensor data. Because the relation is too complex. And recently, the power plant equipment has not experienced an abnormal and trouble condition in the conventional maintenance. So, it is impossible to acquire the sensor data in the abnormal and trouble condition and we can measure the normal condition only.

In order to discover abnormal condition sign of hydroelectric power plants, Kyushu Electric Power Co.,Inc. and Central Research Institute of Electric Power Industry are researching the discovery of abnormal condition sign based on the unusual condition detection using various sensor data, now. In our research, we consider that the increase of generation of some unusual condition data coincide with the abnormal condition sign nearly, because we can measure the normal condition data only from a regular operation hydroelectric power plant. In fact, the abnormal condition is very rare case and very unusual condition. And we developed the discovery method based on the unusual condition detection from regular condition data in the hydroelectric power plant. This method based on the interaction between human expertise and the unusual condition detection system.

In this paper, we describe the measured sensor data briefly in the next section. In the third section, we briefly explain our proposed approach to discover abnormal condition sign from regular operation condition data. And our proposed method is also an interactive data mining method between human expertise and systems. The experimental results are shown in the forth section. Finally, we conclude the concept of the abnormal sign discovery of the hydroelectric power plant bearing vibration based on the detection method of the unusual data.

Table 1. Outline of a targeted Hydroelectric Power Plant

Generated Output		18,000kW
Working Water		$45m^3/s$
Effective Head		46.3m
Turbine Type		Vertical Shaft Francis Turbine
Rated Revolutions Per Minute		240rpm
Bearing Type	Upper Bearing	Oil Self Contained Type Segment Bearing (Natural Cooling)
	Bottom Bearing	Oil Self Contained Type Segment Bearing (Natural Cooling)
	Turbine Bearing	Oil Self Contained Type Cylindrical Bearing (Natural Cooling)
	Thrust Bearing	Oil Self Contained Type Pivot Spring Bearing (Natural Cooling)
Operation Pattern		Process Control Operation (An operation pattern is generated at everyday.)

2 Measurement Data

Table 1 shows the outline of a hydroelectric power plant. The hydroelectric power plant has various sensors to measure data related to bearing vibration. In this paper, the measurement data is collected from the hydroelectric power plant and analyzed by our proposed method. The measurement data, which is related to bearing vibration, is collected from March 16, 2004 to November 23, 2004.

One data has been composed of the sensor and weather information on 38 measurement items for five seconds the measurement interval. All measurement data is normal condition data and does not include an abnormal condition such as accidental condition, trouble condition, and etc.

3 Abnormal Condition Sign Detection

In this section, we describe the outline of the approach of detecting the abnormal condition sign using the unusual condition data.

Generally, the discovery of an abnormal sign is to detect a peculiar case that appears only before an existing abnormal condition by comparing between normal conditions and abnormal conditions. However, it is a fact that there is few data of abnormal conditions in the electric power equipment, because the electric power plants is designed with the high safety factor and maintained appropriately. Currently, our bearing vibration data of the hydroelectric power plant also does not have abnormal conditions and accidental conditions. Therefore, it is impossible to detect a peculiar case before abnormal conditions and accidental conditions happen, because it is hard to take the abnormal or accidental conditions and it is impossible to compare normal conditions with abnormal or accidental conditions. Then, we think the relation between a peculiar condition before an abnormal condition happen (hereafter, we call it the abnormal condition sign) and unusual conditions as the following relation.

$$\textit{The abnormal condition sign} \approx \textit{The increase of generation rate of selected unusual condition.}$$

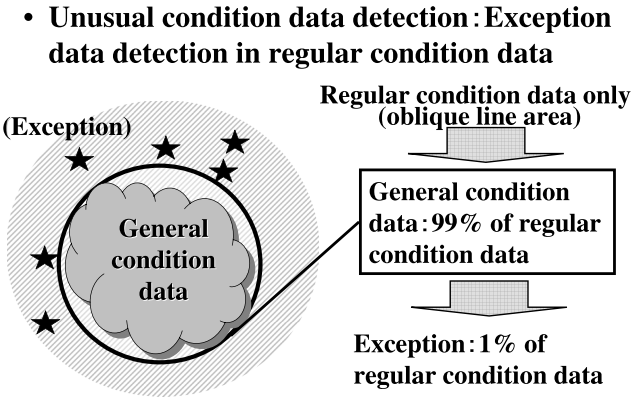


Fig. 1. Image of Unusual Condition Detection

It is possible to change the discovery of the abnormal sign to monitor the generation rate of selected unusual conditions in the normal conditions. In other words, it is think that the unusual condition data with low probability of existing in the normal condition data have the relation with abnormal condition signs.

The figure 1 shows a concept of detection of unusual condition data in the normal condition data. In this figure, the gray oblique line area denotes the normal condition data area. In this research, the unusual condition data are detected from this normal condition data. From the figure 1, if we can find a hypersphere, which can cover the 99% of the normal condition data, we can think that the other 1% data unusual condition data. This 99% normal condition data are called "general condition data". In the figure 1, the inside of a circle shown a black solid line is general condition data area, and black stars denote unusual condition data. Therefore, if we can find a boundary of α% area in the normal condition area correctly, it is possible to detect unusual condition data which do not belong the α% area of normal condition data. We adopt One Class Support Vector Machine (hereafter One Class SVM) to find the boundary correctly [6]. After extracting unusual condition data, our proposed method shows the unusual condition data to human expertise, and human expertise evaluates them and selects abnormal condition data. And our method monitor the generation rate of selected abnormal condition data in new data.

4 One-Class SVM

Schölkopf et al. suggested a method of adapting the SVM methodology to one class classification problem [6]. Essentially, after transforming the feature via a kernel, they treat the origin as the only member of the second class. The using "relaxation parameters" they separate the image of the one class from the origin. Then the standard two class SVM techniques are employed [7,8,9].

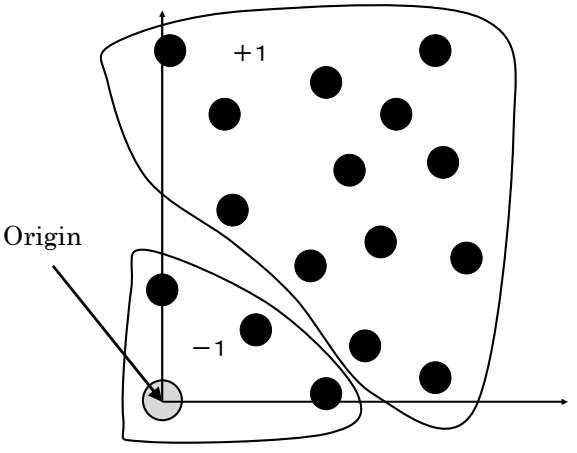


Fig. 2. One Class SVM Classifier: the origin is the only original member of the second class

One Class SVM [6] returns a function f that takes the value +1 in a "small" region capturing most of the training data points, and -1 elsewhere.

The algorithm can be summarized as mapping the data into a feature space H using an appropriate kernel function, and then trying to separate the mapped vectors from the origin with maximum margin(see Figure 2).

Let the training data be $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$ belonging to one class X, where X is a compact subset of R^N and ℓ is the number of observations. Let $\Phi : X \to H$ be a kernel map which transforms the training examples to feature space. The dot product in the image of Φ can be computed by evaluating some simple kernels

$$k(\mathbf{x}, \mathbf{y}) = (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})) \tag{1}$$

such as the Gaussian kernel

$$k(\mathbf{x}, \mathbf{y}) = \exp\left(\frac{\|\mathbf{x} - \mathbf{y}\|^2}{c}\right). \tag{2}$$

The strategy is to map the data into the feature space corresponding to the kernel, and to separate them from the origin with maximum margin. Then, To separate the data set from the origin, one needs to solve the following quadratic program:

$$\begin{aligned} \min_{\mathbf{w} \in H, \xi \in R^\ell \rho \in R^N} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{\nu\ell}\sum_i \xi_i - \rho \\ \text{subject to} \quad & (\mathbf{w} \cdot \Phi(\mathbf{x}_i)) \geq \rho - \xi_i, \\ & \xi_i \geq 0. \end{aligned} \tag{3}$$

Here, $\nu \in (0, 1)$ is an upper bound on the fraction of outliers, and a lower bound on the fraction of Support Vectors.

Since nonzero slack variables ξ_i are penalized in the objective function, we can expect that if $\mathbf{w}$ and ρ solve this problem, then the decision function

$$f(\mathbf{x}) = \mathrm{sgn}\left((\mathbf{w} \cdot \Phi(\mathbf{x})) - \rho\right) \tag{4}$$

will be positive for most examples $\mathbf{x}_i$ contained in the training set, while the SV type regularization term $\|w\|$ will still be small. The actual trade-off between these two is controlled by ν. For a new point $\mathbf{x}$, the value $f(\mathbf{x})$ is determined by evaluating which side of the hyper-plane it falls on, in feature space.

Using multipliers $\alpha_i, \beta_i \geq 0$, we introduce a Lagrangian

$$\begin{aligned} L(\mathbf{w}, \boldsymbol{\xi}, \rho, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{\nu\ell}\sum_i \xi_i - \rho \\ - \sum_i \alpha_i((\mathbf{w} \cdot \mathbf{x}_i) - \rho + \xi_i) - \sum_i \beta_i \xi_i \end{aligned} \tag{5}$$

and set the derivatives with respect to the primal variables $\mathbf{w}, \xi_i, \rho$ equal to zero, yielding

$$\mathbf{w} = \sum_i \alpha_i \mathbf{x}_i, \tag{6}$$

$$\alpha_i = \frac{1}{\nu\ell} - \beta_i \leq \frac{1}{\nu\ell}, \quad \sum_i \alpha_i = 1. \tag{7}$$

In Eqn. (6), all patterns $\{\mathbf{x}_i : i \in [\ell], \alpha_i > 0\}$ are called Support Vectors. Using Eqn. (1), the SV expansion transforms the decision function Eqn. (4)

$$f(\mathbf{x}) = \mathrm{sgn}\left(\sum_i \alpha_i k(\mathbf{x}_i, \mathbf{x}) - \rho\right). \tag{8}$$

Substituting Eqn. (6) and Eqn. (7) into Eqn. (5), we obtain the dual problem:

$$\min_{\boldsymbol{\alpha}} \ \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \tag{9}$$

$$\text{subject to } \ 0 \leq \alpha_i \leq \frac{1}{\nu\ell}, \quad \sum_i \alpha_i = 1. \tag{10}$$

One can show that at the optimum, the two inequality constraints Eqn. (3) become equalities if α_i and β_i are nonzero, i.e. if $0 < \alpha \leq 1/(\nu\ell)$. Therefore, we can recover ρ by exploiting that for any such α_i, the corresponding pattern $\mathbf{x}_i$ satisfies

$$\rho = (\mathbf{w} \cdot \mathbf{x}_i) = \sum_j \alpha_j \mathbf{x}_j \cdot \mathbf{x}_i. \tag{11}$$

In our research, we used the LIBSVM. This is an integrated tool for support vector classification and regression which can handle One Class SVM using the Schölkopf etc algorithms. The LIBSVM is available at *http://www.csie.ntu.edu.tw/˜cjlin/libsvm.*

5 Unusual Data Detection Experiment

In this section, we describe our experiment by using the measurement data, which is explained in section 2. Especially, we briefly introduce our experimental setup, how to verify our experimental results, experimental results, and the evaluation.

5.1 Experimental Setup

Our experiment analyzed the measurement data, which is explained in section 2. The measurement data is composed of 38 measurement items. However, in order to detect the unusual condition data, we extracted the related measurement items to the bearing vibration from all measurement items. So, 16 measurement items were selected by the expertise of the bearing vibration of the experts to analyze the unusual condition data. Table 2 shows these selected 16 measurement items.

The starting condition data and the parallel off condition data are very few in our dataset relatively. The parallel operation condition data are very large. If we analyze the all measurement data to detect the unusual condition data, the detected condition data are the starting condition data or the parallel off condition data. This is not good situation for our analysis. So, the all measurement data is divided into the following four groups by expertise of experts. In practice, these operations should be defined by the operation record of the hydroelectric power plant. However, in order to verify our proposed approach, we defined the four groups by using the expertise of the experts intentionally.

Starting condition:
Generator Voltage(V-W) $<$ 10kV and Guide Vane Opening $\geq$ 10% and Revolutions Per Minute $\geq$ 200 rpm.

Parallel operation condition:
Generator Voltage(V-W) $\geq$ 10kV and Revolutions Per Minute $\geq$ 200 rpm.

Parallel off condition:
Generator Voltage(V-W) $<$ 10kV and Guide Vane Opening $<$ 10% and Revolutions Per Minute $\geq$ 200 rpm.

Stopping condition:
Otherwise.

Table 2. Measurement Items

Measurement Items for Detection Analysis	
A. Generated Output(MW)	B. Revolutions Per Minute
C. Room Tempe.(℃)	D. Water Tempe.(℃)
E. Oil Cooler Inlet Air Tempe.(℃)	F. Oil Cooler Outlet Air Tempe.(℃)
G. Upper Bearing Tempe.(℃)	H. Bottom Bearing Tempe.(℃)
I. Upper Bearing Oil Tempe.e(℃)	J. Turbine Bearing Tempe.(℃)
K. Turbine Bearing Oil Tempe.(℃)	L. Thrust Bearing Tempe.(℃)
M. Bottom Oil Tank Oil Tempe.(℃)	N. Bottom Bearing Inlet Air Tempe.(℃)
O. Generator Shaft Vibration (X axis)(μm)	P. Turbine Shaft Vibration (X axis)(μm)

Table 3. The typical unusual condition data in the starting condition data

Measurement Item	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8	Case 9
A.Generated Output(MW)	-0.192	-0.192	-0.192	*-0.192*	*-0.192*	-0.192	*-0.162*	-0.192	-0.18
B.Revolutions Per Minute	239	224	238	*243*	*243*	238	*330*	233	224
C.Room Tempe.(℃)	15.0	16.1	15.2	*23.1*	*23.1*	29.8	*28.8*	27.4	26.8
D.Water Tempe.(℃)	10.1	10.9	10.4	*21.1*	*21.1*	27.9	*27.4*	25.8	24.5
E.Oil Cooler Inlet Air Tempe.(℃)	20.3	22.9	18.0	*25.9*	*25.9*	36.6	*30.9*	28.6	27.9
F.Oil Cooler Outlet Air Tempe.(℃)	25.4	27.6	21.8	*35.0*	*34.8*	40.9	*40.5*	29.9	28.5
G.Upper Bearing Tempe.(℃)	40.0	41.0	36.2	*46.4*	*46.5*	50.3	*50.8*	32.2	30.6
H.Bttm Bearing Tempe.(℃)	36.3	35.1	32.9	*44.3*	*44.3*	44.5	*50.9*	31.2	29.6
I.Upper Bearing Oil Tempe.(℃)	35.9	36.8	32.1	*42.9*	*42.8*	47.7	*47.5*	31.6	29.5
J.Turbine Bearing Tempe.(℃)	34.1	31.7	30.8	*41.1*	*41.1*	40.8	*43.8*	24.6	22.5
K.Turbine Bearing Oil Tempe.(℃)	32.3	30.0	29.2	*33.1*	*33.1*	38.2	*37.0*	24.8	22.5
L.Thrust Bearing Tempe.(℃)	40.4	41.3	36.5	*54.6*	*54.8*	51.3	*59.7*	32.1	30.4
M.Bottom Oil Tank Oil Tempe.(℃)	33.3	32.5	29.7	*40.5*	*40.4*	43.3	*47.8*	30.1	28.6
N.Bottom Bearing Inlet Air Tempe.(℃)	18.8	21.0	18.1	*24.3*	*24.5*	33.5	*29.2*	29.1	27.8
O.Generator Shaft Vibration(X axis)(μm)	11	22	16	*12*	*8*	15	*23*	11	14
P.Turbine Shaft Vibration(X axis)(μm)	21	33	31	*34*	*27*	35	*33*	20	22

Table 4. No. of data in Each Condition

Group	The number of data
Stopping condition	433,935
Starting condition	120
Parallel operation condition	2,368,756
Parallel off condition	132
Total	2,804,113

These groups defined by the expertise of the experts. Table 4 shows the number of data in each group.

In the stopping condition group, the bearing does not rotate. This group data were omitted from the analyzed data. In other words, the unusual condition data were detected in each group, which is starting condition or parallel operation condition or parallel off condition. In order to ignore the different measurement units, the measurement data is normalized into the average 0 and the variance 1 at each measurement item.

5.2 Detection Results and Evaluation

The unusual condition data were detected in each group data of the starting condition, the parallel operation condition and the parallel off condition by applying One Class SVM, which is introduced in section 3. Our experiments made the trial that changes the detected number of unusual condition data. In this section, we report the analysis result to each group data in the number of unusual condition data. The experts judged that the number was appropriate.

Detection of the unusual condition data in the starting condition data. The starting condition data were applied One Class SVM to detect unusual condition data in these data. The parameter ν, which can determine the rate of the unusual condition data in the analyzed data, was set 0.05. This analysis can detect the unusual condition data, whose existence probabilities are smaller than 0.05. We could detect 9 unusual condition data by the analysis. Table 3 shows the typical unusual condition data, which are detected by our experiment.

Table 5. Data Before and After Case 4 and 5

Measurement Items	-15 sec.	-10 sec.	-5 sec.	Case 4	Case 5	+10 sec.	+15 sec.
A.Generated Output(MW)	4.272	4.32	4.452	*-0.192*	*-0.192*	4.26	2.352
B.Revolutions Par Minute	243	243	243	*243*	*243*	243	243
C.Room Tempe.(℃)	23.2	23.1	23.1	*23.1*	*23.1*	23.1	23.2
D.Water Tempe.(℃)	21.2	21.0	21.1	*21.1*	*21.1*	21.2	21.2
E.Oil Cooler Inlet Air Tempe.(℃)	25.9	26.1	25.9	*25.9*	*25.9*	26.1	26.1
F.Oil Cooler Outlet Air Tempe.(℃)	34.8	35.0	34.9	*35.0*	*34.8*	34.9	34.8
G.Upper Bearing Tempe.(℃)	46.4	46.6	46.5	*46.4*	*46.5*	46.3	46.6
H.Bottom Bearing Tempe.(℃)	44.3	44.3	44.3	*44.3*	*44.3*	44.5	44.2
I.Upper Bearing Oil Tempe.(℃)	42.9	42.8	42.8	*42.9*	*42.8*	43.0	43.0
J.Turbine Bearing Tempe.(℃)	41.1	41.1	41.1	*41.1*	*41.1*	41.1	41.1
K.Turbine Bearing Oil Tempe.(℃)	33.1	32.9	33.2	*33.1*	*33.1*	33.0	33.2
L.Thrust Bearing Tempe.(℃)	54.8	54.8	54.8	*54.6*	*54.8*	54.6	54.8
M.Bottom Oil Tank Oil Tempe.(℃)	40.4	40.6	40.4	*40.5*	*40.4*	40.6	40.3
N.Bottom Bearing Inlet Air Tempe(℃)	24.5	24.4	24.5	*24.3*	*24.5*	24.6	24.4
O.Generator Shaft Vibration(X axis)(μm)	14	11	9	*12*	*8*	11	9
P.Turbine Shaft Vibration(X axis)(μm)	40	25	26	*34*	*27*	31	34

Table 6. The typical unusual condition data in the parallel operation condition data (1)

Measurement Item	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7
A.Generated Output(MW)	1.602	2.112	1.542	2.622	1.122	16.68	16.26
B.Revolutions Per Minute	242	242	242	242	242	243	242
C.Room Tempe.(℃)	15.0	14.8	15.0	15.0	12.1	23.3	28.6
D.Water Tempe.(℃)	10.2	10.3	10.7	10.3	12.2	20.8	27.4
E.Oil Cooler Inlet Air Tempe.(℃)	18.4	18.3	16.9	16.7	11.1	25.9	31.2
F.Oil Cooler Outlet Air Tempe.(℃)	24.2	24.4	22.9	23.2	22.0	34.3	40.6
G.Upper Bearing Tempe.(℃)	37.9	38.1	37.0	37.3	39.6	45.8	50.7
H.Bottom Bearing Tempe.(℃)	35.0	35.2	34.6	35.6	44.3	42.6	51.0
I.Upper Bearing Oil Tempe.(℃)	34.1	34.2	33.0	33.2	35.1	42.2	47.6
J.Turbine Bearing Tempe.(℃)	32.5	32.9	32.9	33.5	43.3	39.7	44.1
K.Turbine Bearing Oil Tempe.(℃)	29.4	29.3	29.1	29.0	35.5	31.9	37.4
L.Thrust Bearing Tempe.(℃)	39.6	40.3	39.1	40.6	46.9	53.9	59.8
M.Bottom Oil Tank Oil Tempe.(℃)	32.1	32.4	31.6	32.0	39.2	38.7	47.8
N.Bottom Bearing Inlet Air Tempe.(℃)	16.3	16.3	15.9	15.7	12.1	24.5	29.0
O.Generator Shaft Vibration(X axis)(μm)	18	21	21	19	24	17	22
P.Turbine Shaft Vibration(X axis)(μm)	40	35	42	44	87	26	51

In the table 3, the case 4 and 5 should belong to the parallel operation condition actually. Table 5 shows the data before and after 15 seconds of the case 4. In this table, we can recognize that the case 4 and 5 should belong to the parallel operation condition. This situation shows that the expertise of experts could not describe the starting condition completely. However, our proposed approach can detect the case 4 and 5, which should belong to the parallel operation condition, as the unusual condition data. This fact shows that our approach can find the appropriate unusual condition data, which should be detected in the starting condition. In the table 3, the case 7 is an accidental interruption. This fact was researched by using an operation record, an daily report, and etc. The accidental interruption denotes a power plant parallel off when the power system has an accident, and is very rare case.

Detection of the unusual condition data in the parallel operation condition data. The parallel operation condition data were applied One Class

Table 7. The typical unusual condition data in the parallel operation condition data (2)

Measurement Item	Case 8	Case 9	Case 10	Case 11	Case 12	Case 13	Case 14
A.Generated Output(MW)	15.792	14.64	*-0.192*	*-0.192*	*-0.192*	-0.18	1.632
B.Revolutions Per Minute	242	243	*232*	*231*	*229*	233	242
C.Room Tempe.(℃)	28.6	28.7	*26.8*	*26.8*	*26.7*	27.6	12.4
D.Water Tempe.(℃)	27.4	27.6	*25.6*	*25.6*	*25.6*	25.7	13.1
E.Oil Cooler Inlet Air Tempe.(℃)	30.8	30.6	*28.1*	*28.0*	*28.0*	28.6	12.8
F.Oil Cooler Outlet Air Tempe.(℃)	40.3	40.0	*35.3*	*35.3*	*35.3*	30.6	22.5
G.Upper Bearing Tempe.(℃)	50.5	50.7	*45.6*	*45.6*	*45.8*	36.8	39.2
H.Bottom Bearing Tempe.(℃)	50.7	50.9	*43.5*	*43.7*	*43.8*	32.8	44.2
I.Upper Bearing Oil Tempe.(℃)	47.3	47.4	*42.4*	*42.6*	*42.4*	34.7	35.1
J.Turbine Bearing Tempe.(℃)	43.8	43.7	*39.9*	*40.0*	*39.9*	27.4	43.8
K.Turbine Bearing Oil Tempe.(℃)	36.5	36.0	*31.1*	*31.1*	*31.1*	25.2	36.3
L.Thrust Bearing Tempe.(℃)	59.7	59.4	*52.9*	*52.9*	*53.1*	38.1	46.8
M.Bottom Oil Tank Oil Tempe.(℃)	47.8	48.1	*40.3*	*40.2*	*40.2*	30.8	39.0
N.Bottom Bearing Inlet Air Tempe.(℃)	29.2	29.1	*26.8*	*26.7*	*26.8*	28.5	11.7
O.Generator Shaft Vibration(X axis)(μm)	20	22	*23*	*18*	*12*	10	23
P.Turbine Shaft Vibration(X axis)(μm)	54	63	*70*	*54*	*47*	24	87

Table 8. The typical unusual condition data in the parallel off operation condition data

Measurement Item	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8	Case 9	Case 10	Case 11
A.Generated Output(MW)	-0.21	-0.192	-0.192	-0.192	-0.192	-0.192	-0.192	*-0.18*	*-0.192*	*-0.192*	*-0.192*
B.Revolutions Per Minute	227	208	239	233	240	208	242	*261*	*208*	*225*	*236*
C.Room Tempe.(℃)	15.6	16.4	13.8	16.9	22.8	22.7	29.9	*28.8*	*28.8*	*26.7*	*27.6*
D.Water Tempe.(℃)	9.4	9.6	13.2	14.7	20.6	20.4	28.7	*27.6*	*27.6*	*25.7*	*25.6*
E.Oil Cooler Inlet Air Tempe.(℃)	17.6	18.6	13.9	20.1	26.3	26.3	32.3	*31.1*	*31.1*	*28.0*	*28.6*
F.Oil Cooler Outlet Air Tempe.(℃)	27.4	28.0	24.1	29.1	32.0	31.9	41.1	*40.6*	*40.5*	*35.3*	*30.8*
G.Upper Bearing Tempe. (℃)	42.4	42.3	40.3	43.0	42.6	42.8	50.8	*50.6*	*50.7*	*45.6*	*36.6*
H.Bottom Bearing Tempe. (℃)	45.6	45.6	44.7	44.7	38.1	38.1	51.6	*50.7*	*50.9*	*43.7*	*32.5*
I.Upper Bearing Oil Tempe.(℃)	38.3	38.3	36.2	39.0	39.0	39.3	48.1	*47.6*	*47.7*	*42.6*	*34.6*
J.Turbine Bearing Tempe. (℃)	42.4	42.6	42.6	41.5	35.5	35.6	48.1	*44.0*	*43.9*	*40.0*	*26.8*
K.Turbine Bearing Oil Tempe.(℃)	35.5	35.0	34.8	34.0	31.9	31.9	43.0	*37.0*	*37.0*	*31.1*	*25.3*
L.Thrust Bearing Tempe. (℃)	50.2	49.9	47.9	50.9	45.6	45.8	59.4	*59.7*	*59.7*	*53.1*	*37.2*
M.Bottom Oil Tank Oil Tempe.(℃)	41.0	41.1	39.7	40.2	35.4	35.4	48.8	*47.7*	*47.9*	*40.3*	*30.9*
N.Bottom Bearing Inlet Air Tempe.(℃)	16.4	17.1	14.1	17.9	24.6	24.6	31.0	*29.2*	*29.4*	*26.6*	*28.2*
O.Generator Shaft Vibration (X axis)(μm)	11	11	12	12	13	13	12	*17*	*10*	*12*	*9*
P.Turbine Shaft Vibration (X axis)(μm)	35	33	34	33	28	23	46	*37*	*37*	*34*	*23*

SVM to detect unusual condition data in these data. The parameter ν, which can determine the rate of the unusual condition data in the analyzed data, was set 5×10^{-6}.

We could detect 14 unusual condition data by the analysis. Table 6 and 7 shows the typical unusual condition data, which are detected by our experiment. In the table 7, the case 10, 11 and 12 of these unusual condition data were test operation. This fact was researched by using an operation record, an daily report, and etc. The test operation denotes a short term operation to check the electric power plant, and is very rare case. This fact shows that our approach can find the appropriate unusual condition data, which should be detected in the parallel operation condition.

Detection of the unusual condition data in the parallel off condition data. The parallel off condition data were applied One Class SVM to detect unusual condition data in these data. The parameter ν, which can determine the rate of the unusual condition data in the analyzed data, was set 0.05.

We could detect 11 unusual condition data by the analysis. Table 8 shows the typical unusual condition data, which are detected by our experiment. In the table 8, the case 8 and 9 of these unusual condition data were an accidental interruption. The other two cases (the case 10 and 11) of the unusual condition data were a test operation. These facts were researched by using an operation record, an daily report, and etc. The test operation and the accidental interruption are very rare case. These facts show that our approach can find the appropriate unusual condition data, which should be detected in the parallel off condition.

5.3 Inexplainable Unusual Condition Data

In our experiment, there are some inexplainable unusual condition data in the above described unusual condition data. Our approach could detect nine unusual condition data in the starting condition. The two data should belong to the parallel operation condition. The other one pattern is the accidental interruption. However, the rest six data are inexplainable unusual condition data. This thing occurs in the parallel operation condition data and the parallel off condition data. These inexplainable unusual condition data are very important for the discovery of the abnormal condition sign.

It is easy for experts to manage the explainable unusual condition data, because the experts understand the evidence of generation of the unusual condition data. However, it is difficult to manage the inexplainable unusual condition data, because the experts can not understand the evidence of generation of the unusual condition data without the experience of the different abnormal conditions. Therefore, the following two parts are important for the discovery of abnormal condition signs and the risk management of hydroelectric power plants.

1. A system discovers the unusual condition data, and then human experts evaluate the discovered data and select the inexplainable data. We call the inexplainable unusual condition data “abnormal condition data”.
2. A system classifies the inexplainable data as abnormal condition data in unseen data and monitors the trend of generation of the inexplainable data.

6 Conclusion

In this paper, we described the discovery method of abnormal condition signs based on the unusual condition detection data in hydroelectric power plants and the interactive method between human expertise and systems to select abnormal condition data. Our proposed approach is a kind of the interactive data mining for hydroelectric power plants. We proposed an unusual condition detection of bearing vibration in hydroelectric power plants base on the One Class SVM. And

we show that our proposed approach could select some unusual condition data as abnormal condition data in hydroelectric power plants by using expertise of experts.

References

1. Yamana, M., Murata, H., Onoda, T., Oohashi, T., Kato, S.: Comparison of pattern classification methods in system for crossarm reuse judgment on the basis of rust images. In: Proceedings of Artificial Intelligence and Applications 2005, pp. 439–444 (2005)
2. Jardine, A.K.S.: Repairable system reliability: Recent developments in CBM optimization. In: 19th International Congress and Exhibition on Condition Monitoring and Diagnostic Engineering Management (COMADEM), Luleä, Sweden, June 13-15 (2006)
3. Tsang, A.H.C., Yeung, W.K., Jardine, A.K.S., Leung, P.K.: Data management for CBM optimization. Journal of Quality in Maintenance Engineering 12, 37–51 (2006)
4. Lin, D., Banjevic, D., Jardine, A.K.S.: Using principal components in a proportional hazards model with applications in condition-based maintenance. The Journal of the Operational Researche Sciety 57, 910–919 (2006)
5. Zuashkiani, A., Banjevic, D., Jardine, A.K.S.: Incorporating expert knowledge when estimating parameters of the proportional hazards model. In: Proceedings of Reliability and Maintainability Symposium, Newport Beach, CA, January 23-26 (2006)
6. Schölkopf, B., Smola, A., Williamson, R., Bartlett, P.: New support vector algorithms. Neural Computation 12, 1083–1121 (2000)
7. Cortes, C., Vapnik, V.: Support Vector Networks. Machine Learning 20, 273–297 (1995)
8. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer, Heidelberg (1995)
9. Vapnik, V.N.: Statistical Learning Theory. Wiley, New York (1998)

Interactive Visualization System for Decision Making Support in Online Shopping

Tomoki Kajinami, Takashi Makihara, and Yasufumi Takama

Tokyo Metropolitan University,
6-6 Asahigaoka, Hino, Tokyo, 191-0065 Japan

Abstract. This paper proposes an interactive visualization based on decision making approach, which helps a user to make a comparison between items and to select certain item in online shopping. This paper employs an interactive visualization system *Keyword Map*, which is designed for general-purpose visualization of graph structure. The system has several functions for emphasizing user's intentions and can reflect those to a map. In this paper, improved Keyword Map is proposed and applied to interactive retrieval process of online shopping. Experiments with test subjects are performed to compare user's behaviors in online shopping between the proposed system and ordinary Web browser. The results show that subjects using proposed system tend to analyze relationship between items, which leads to the acceleration of concept articulation of online shopping.

Keywords: Interactive Retrieval, Decision Making, Information Visualization, Relevance Feedback.

1 Introduction

Recently, we can receive several services through many online stores on the Web. Many methodologies of the site design have been studied to win customers and to increase sales [3]. In online shopping, a user tends to compares items in order to buy a better one among several candidates. For example, many people want to buy good items at popular prices. They would analyze those from various viewpoints (e.g. price, color, design, brand, and so on.), using manufacturer sites and/or price comparison sites (e.g. Kakaku.com[1]). In any case, a user decides which item to buy through analyzing items from several viewpoints. Therefore, acceleration of concept articulation is also important in online shopping. Concept articulation means that when a user initially has only vague requirements for items, he/she gradually clarifies requirements for those through interactive analysis from various viewpoints [4].

Information visualization techniques have been studied to help a user's perception about multi-dimensional data, complicated relationship between objects, and so on [1]. We regard the comparison of items by a user as the perception

[1] Kakaku.com — http://kakaku.com/ (Japanese site).

S. Chawla et al. (Eds.): PAKDD 2008 Workshops, LNAI 5433, pp. 193–202, 2009.

of relationship between objects, and apply information visualization techniques for supporting online shopping. A Keyword Map has been studied for general-purpose visualization of relationship between objects (keywords) [2]. However, the Keyword Map treats only one kind of relationships between keywords, which means a user has to perceive relationships between keywords from one viewpoint.

In this paper, we extend existing Keyword Map so that a user can analyze/view relationship between keywords from various viewpoints, and propose the Keyword Map-based online shopping support system. We show through experiments with test subjects that proposed system can offer various viewpoints to a user, which leads to support his/her decision making in online shopping.

In section 2, we first describe basic algorithm of the Keyword Map and summarize its features. And we introduce *Multiple Relevance Controller* as the new feature of the Keyword Map. The Multiple Relevance Controller enables a user to analyze/view relationship between keywords from various viewpoints. In section 3, we perform a preparatory experiment with test subjects for confirming the effectiveness of the Multiple Relevance Controller. In section 4, we propose the Keyword Map-based online shopping support system. In section 5, we perform experiments with test subjects for evaluating the effectiveness of proposed system.

2 Keyword Map: Interactive Information Visualization System

2.1 Overview of Keyword Map

Keyword Map is an interactive information visualization system designed for general-purpose visualization of graph structure. The system employs the spring model which is a kind of the force-directed graph drawing methods to arrange keywords on 2D space. The basic algorithm of Keyword Map is as follows.

1. Define the distance l_{ij} between keyword i and j based on their degrees of relationship $R_{ij} \in [-1, 1]$ by Eq.(1) (m is positive constant).

$$l_{ij} = m(1 - R_{ij}). \tag{1}$$

2. The moving distance of keyword i in each step, $(\delta_{xi}, \delta_{yi})$, is calculated by Eq.(2).

$$(\delta_{xi}, \delta_{yi}) = \left(c\frac{\partial E_{ij}}{\partial x_i}, c\frac{\partial E_{ij}}{\partial y_i} \right), \tag{2}$$

$$E_{ij} = \sum_i \sum_j \tfrac{1}{2} k_{ij} (d_{ij} - l_{ij})^2, \tag{3}$$

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}. \tag{4}$$

3. By repeating above calculation and update of coordinates, a keyword arrangement is closing to one of local optimal arrangements.

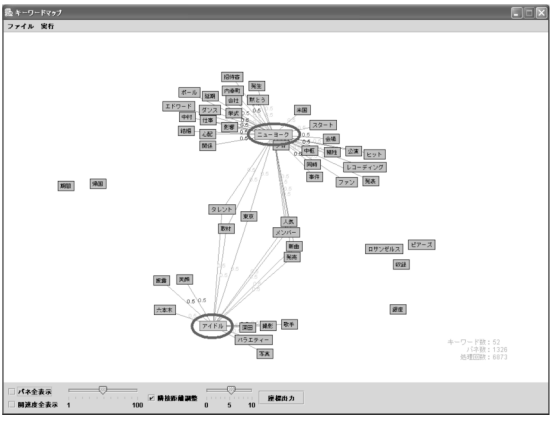


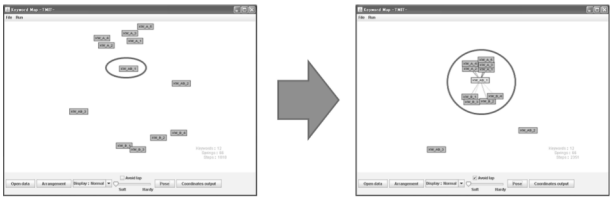
Fig. 1. Screenshot of Keyword Map

The keywords with the high R_{ij} will be arranged close to each other, while those with low or negative R_{ij} will be arranged far away from each other.

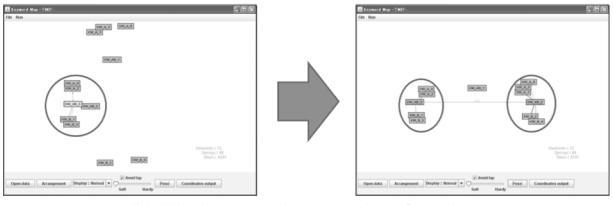
Figure 1 shows the screenshot of Keyword Map. Keyword Map receives a set of keywords and their degrees of relationship as an input and draws the graph structure on 2D space with an animation. A keyword is represented as a rectangle. A link is displayed between keywords when its degree of relationship is positive.

Keyword Map provides several interactive functions. A user can drag arbitrary keywords for changing positions (enclosed with circles in Fig.1). When a position of a keyword is changed, the positions of its related keywords are also changed in real-time. A user can also *disable* arbitrary keywords, which are ignored by the real-time calculation. Keyword Map also provides high level of interactive functions such as *keyword concentration*, *keyword separation*, and *keyword formation*.

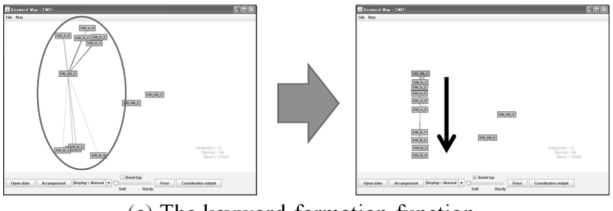
Figure 2(a) shows an example of how the keyword concentration function is used. This function makes a keyword cluster, in which a user is interested (enclosed with a circle in left of Fig.2(a)). By using this function, a user can gather relevant keywords related with an arbitrary keyword (enclosed with a circle in right of Fig.2(a)). Figure 2(b) shows an example of how the keyword separation function is used. This function divides a keyword cluster (enclosed with a circle in left of Fig.2(b)) into two or more clusters (enclosed with two circles in right of Fig.2(b)). By using this function, a user can arrange keywords clusters separately as different topics. Figure 2(c) shows an example of how the keyword formation function is used. This function lines up keywords or keyword clusters downward/rightward. By using this function, a user can align keywords or keyword clusters in order of their importance. These functions can be used for emphasizing/reflecting user's intention and viewpoint [2].



(a) The keyword concentration function



(b) The keyword separation function



(c) The keyword formation function

Fig. 2. High level interactive functions

2.2 Introduction of Multiple Relevance Controller

Previous study has investigated the effectiveness of the system in terms of user's concept articulation in his/her decision making process (e.g. tourist route decision, college course decision) [2]. The experimental result showed that users can reflect his/her own intention in keywords arrangement using the interactive functions and make his/her own concept clear. However, when initial keywords arrangement made by the system is similar to what a user expected, his/her motivation to edit a map tends to decrease, which makes it difficult to find new interpretation. This means a user interprets a map only from one viewpoint if the system arranges keywords according to single relevance type. A relevance type is one of factors of relationships between keywords. When keywords are arranged according to different relevance type, a user can analyze/view a map from different viewpoint.

In online shopping, alike shopping in the real world, a user tends to examine various attributes items, such as brands, colors, shapes, in addition to price. That is, a user analyzes items from various viewpoints. Therefore, The keyword Map needs a new function to offer multiple viewpoints to a user based on the combination of multiple relevance types. In this section, we introduce new function "Multiple Relevance Controller" into existing Keyword Map, so that it can combine multiple relevance types by changing their weights. By introduction the

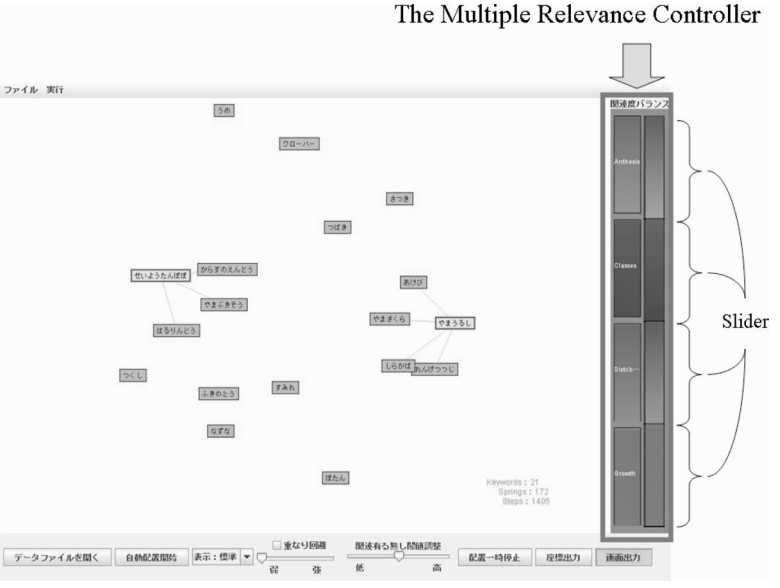


Fig. 3. The Keyword Map equipped with the Multiple Relevance Controller

Multiple Relevance Controller, it is expected that a user can analyze/view a data set from various viewpoints.

The controller is realized by extending definition of R_{ij} in existing Keyword Map. When relationship between keyword i and j has n relevance types, the r_{mij} is relevance value of $m(\in N)$ relevance type. The $g_m(\geq 0)$ is its weight. The ${R^s}_{ij}$ is weighted sum of relevance values (Eq.(6)). The extended $R_{ij}(-1 \leq R_{ij} \leq 1)$ is defined by Eq.(5). Where, ${R^s}_{\max}$ is maximal value of $|{R^s}_{ij}|$.

$$R_{ij} = \frac{{R^s}_{ij}}{{R^s}_{\max}}, \tag{5}$$

$${R^s}_{ij} = \sum_{m=1}^{n} g_m r_{mij}. \tag{6}$$

A user can change g_m in order to reflect his/her thought about the importance degree of each relevance type to R_{ij}.

Figure 3 shows the screenshot of Keyword Map equipped with Multiple Relevance Controller. Not only a single relevance type, but also the combination of multiple relevance types is available. The selection/combination of relevance types are performed with sliders on the right side of the window in Fig.3. Each relevance type is assigned to a different slider, on which using a wheel of a mouse adjusts its degree of influence on visualization. Hereinafter, Keyword Map equipped with Multiple Relevance Controller is simply called Keyword Map.

3 Effectiveness of the Multiple Relevance Controller

In this section, we confirm effectiveness of the Multiple Relevance Controller through a preparatory experiment with test subjects. The purpose of the experiment is to confirm difference of the users' interpretation according to different weight settings of relevance types. If subjects make different interpretation depending on the weight settings of relevance type, it can be said that the Multiple Relevance Controller can offer subjects various viewpoints to analyze a map.

First, we prepared a data set that has thirty keywords and three relevance types. All keywords' labels are drawn as "keyword and keyword's number" on the map, i.e., "keyword 1," "keyword 2," and "keyword 30." Relevance values of three relevance types are defined at random, each of which is expected to

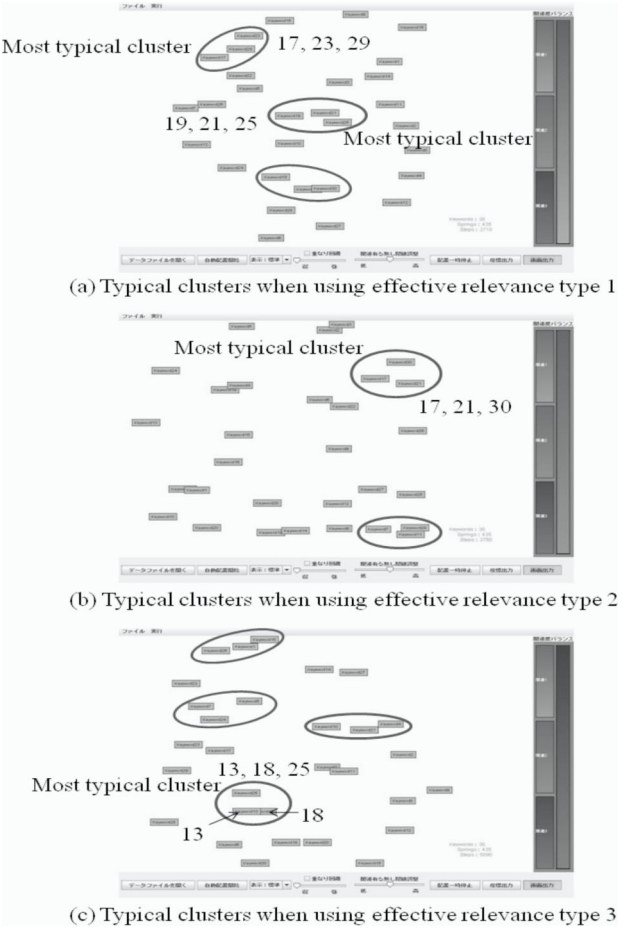


Fig. 4. Typical clusters found in each effective relevance type

give users different viewpoints. Second, we presented a map to test subjects. Each weight of two relevance types in a map is set to zero. That means subjects view/analyze a map depending on only one of relevance types—in this section, it is called an *effective relevance type*. Next, subjects were asked to cluster keywords that represent features of the map. Their task is to find at most three clusters in the map. When performing the task, a condition that each cluster contains three to six keywords is imposed on test subjects. We presented three maps in turn, each of which is drawn with one of relevance types. This experiment was performed by 10 test subjects.

Figure 4 shows that keyword clusters found by test subjects in each effective relevance type. The clusters enclosed with circles are found by more than 6 subjects. In Figure 4(a), the most typical cluster includes keyword 17, 23, and 29, which was found by 9 subjects. Another one cluster was also found by 9 subjects, which includes keyword 19, 21, and 25. In Figure 4(b), the most typical cluster includes keyword 17, 21, and 30, which was found by all subjects. In Figure 4(c), the most typical cluster includes keyword 13, 18, and 25, which was found by 9 subjects.

From the results, it can be said that test subjects make different interpretation of the map that consists of the same keyword set, when the map is drawn according to the effective relevance types. That means subjects viewed the map from different viewpoints. Therefore, it is expected the introduction of Multiple Relevance Controller makes it possible for users to view the same map from multiple viewpoints.

4 Keyword Map-Based Online Shopping Support System

We propose a Keyword Map-based online shopping support system. Figure 5 shows the system architecture. The system combines the Keyword Map with backend system. The backend system makes data set for the Keyword Map from Amazon.co.jp (using AmazonAPI[2]). The Keyword Map can read data set from the backend system, and outputs an edit log of a map by a user to the backend system. The backend system extracts user's intention from the log and makes new data set. This process is called Keyword Map-based relevance feedback [5].

5 Experiment

5.1 Purpose of Experiment and Preparation

The purpose of the experiment is to confirm difference of users' behaviors in online shopping, by comparing the behaviors when using the proposed system with that when using ordinary Web browser (e.g. Internet Explorer). It can be said that the proposed system can support decision making through users' trial and errors, if they can find new conceptions and search items more actively compared with when they use ordinary Web browser. The subject's task is to select

[2] Amazon.co.jp — http://www.amazon.co.jp (Japanese site).

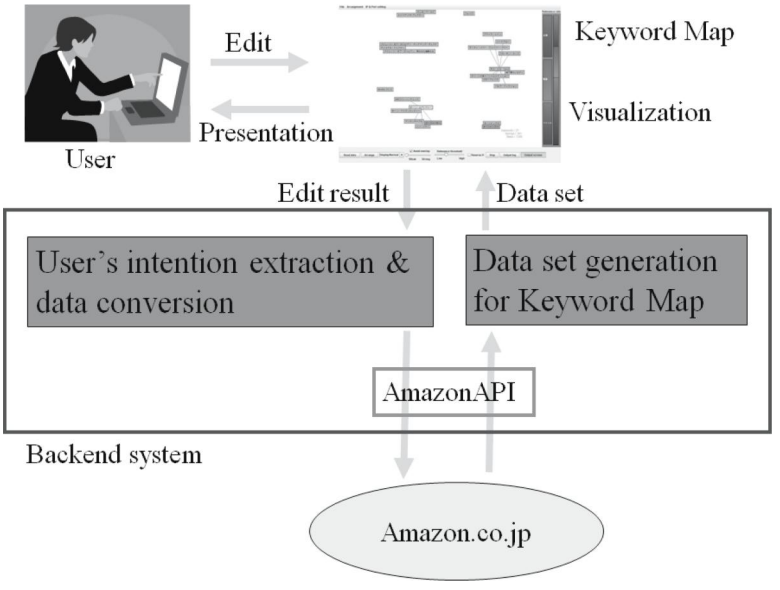


Fig. 5. Architecture of the Keyword Map-based online shopping support system

a movie's DVD item for a gift to his/her friend and parents. Actual searching process using the system is as follows.

1. A subject views a map constructed from a seed (initial query) item.
2. He/she edits the map, during which he/she analyzes relationship between items using the interactive functions including the Multiple Relevance Controller.
3. From the edited map, the backend system including the AmazonAPI searches movie DVDs, and makes new data set for the keyword map.
4. Step2 and 3 are repeated until he/she makes a decision on buying a certain item.

Data set for the Keyword Map includes three relevance types; actors, director, and genre. And Keyword labels are movies' titles. Therefore, test subjects can view/analyze relationship between movies with the combination of these relevance types. By selecting an arbitrary movie as a seed, a subject starts searching process either using the proposed system or accessing Amazon.co.jp with an ordinary Web browser. We compare subjects' behaviors by the proposed system with that by an ordinary Web browser. This experiment was performed by six test subjects.

5.2 Experimental Results and Discussion

Table 1 summarizes results obtained from questionnaire and test subjects' behavior logs. Average score of amusement level and satisfaction level are evaluated by

Table 1. Comparing proposed search process with ordinary one

	Ordinary Web browser	Proposed system
Avg. score of amusement level	2.83	4.00
Avg. score of satisfaction level	2.83	2.67
Avg. # of repeat search	6.33	8.17
Mean task time (minutes)	7.17	10.67

questionnaire. Their evaluations are given with five scale (1-5, 5 is the highest). Furthermore, they are asked to give as many comments as possible.

From Table 1, when the proposed system is used, amusement level is higher than the ordinary Web browser (significant, $|t| = 2.44, P = 0.03 < 0.05$), and the average number of repeated search and mean task time increase. From the result, it can be said that the proposed system encourages test subjects to repeatedly search items while enjoying the process itself. On the other hand, as for using the ordinary Web browser, test subjects did not search items so actively, compared to when test subjects used the proposed system. It is also observed that test subjects using the proposed system tended to use one or two relevance types at the same time, which means they can analyze items' relationships from multiple viewpoints for finding new conceptions. Moreover, obtained typical comments about the reason for choosing movies, and used interactive functions are as follows.

- Test subjects using ordinary Web browser
  - He/she wants to find famous movies.
  - He/she is interested in favorite movies.
  - He/she is interested in known movies.
- Test subjects using proposed system
  - He/she wants to examine relationships between movies.
  - He/she is interested in particuler relevance type and selects it using the Multiple Relevance Controller.
  - He/she is interested in common relationship between unknowing movies.

From the result, characteristic of test subjects' behaviors in search are summarized as follows: test subjects using the ordinary Web browser tend to search items based on their already-known information. But whose who use the proposed system tend to search items based on interactive data analysis of maps from various viewpoints. It means that they make their concept articulation on purchase through trial and errors.

From the results, it can be said that the proposed Keyword Map-based online shopping support system can support a user's decision making on online shopping, by accelerating trigger of concept articulation based on interactive data analysis from various viewpoints.

6 Conclusions

The Keyword Map-based online shopping support system is proposed for supporting decision making in online shopping. The proposed system makes it

possible for users to analyze relationships between items from various viewpoints. We perform an experiment with test subjects for investigating the effectiveness of the proposed system. Experimental results show that subjects can analyze items from various viewpoint using the proposed system, which means the proposed system can support a user's decision making by accelerating their concept articulation through interactive item analysis from various viewpoints. Future studies include the application of the Keyword Map to various fields including visual data mining and chance discovery, as well as the detailed analysis of users' exploratory activities on Keyword Map.

References

1. Information Visualization. In: Card, S.T., Mackinlay, J.D., Shneiderman, B. (eds.) Reading in Information Visualization: Using Vision to Think, ch. 1, pp. 1–34. Morgan Kaufman, San Francisco (1999)
2. Kajinami, T., Takama, Y.: Examination of Arrangement Support Functions for Emphasizing User's Intentions on Keyword Map. In: ICARCV 2006, pp. 1667–1671 (2006)
3. Lohse, G.L., Spiller, P.: Electronic Shopping. Communications of the ACM 41(7), 81–88 (1998)
4. Shoji, H., Hori, K.: S-Conart an interaction method that facilitates concept articulation in shopping online. AI&Society 19(1), 65–83 (2005)
5. Takama, Y., Kajinami, T.: Keyword Map-based Relevance Feedback for Web Information Retrieval. In: 2nd Pan-Pacific Symposium on Information Technology, pp.41–44 (2004)

A Method to Recognize and Count Leaves on the Surface of a River Using User's Knowledge about Color of Leaves

Fujio Tsutsumi[1] and Yutaka Tateda[2]

[1] System Engineering Research Laboratory,
Central Research Institute of Electric Power Industry,
2-11-1, Iwado kita, Komae-shi, Tokyo 201-8511, Japan
[2] Environmental Science Research Laboratory,
Central Research Institute of Electric Power Industry,
1646 Abiko, Abiko-shi, Chiba 270-1194, Japan

Abstract. This paper proposes a recognition and count method of leaves on the surface of a river to be used in mangrove ecosystem monitoring. Conventionally counting leaves required considerable manual labor for precise monitoring of material flow in the ecosystem. Therefore an efficient counting method was needed. Our method automatically recognizes and counts the number of floating leaves in recorded video using color and motion features. The color feature is represented by 3 dimensional histogram of a color space. We have developed a user interface based on the interactive machine learning model to acquire the color feature from video images. The user can easily produce a huge number of sample data to extract the color feature by the user interface in the same way as coloring a picture. For the motion feature, speed and acceleration of the targets are used. The counting method proposed in this paper has been applied to three videos (total five hours) which recorded about 20,000 leaves, and high recall and precision rates of 96% and 94%, respectively, have been achieved.

Keywords: Video Image Processing, Color Image Processing, Environmental Monitoring, Interactive Machine Learning.

1 Introduction

In the mangrove ecosystem, by photosynthesis of trees, CO_2 is absorbed from the atmosphere and mangrove forests store carbon. Moreover, it is known that organic matters of the mangrove origin, such as fallen leaves, turn into a sediment, accumulate them into mangrove forests, and they store carbon as a peat layer. However, it turns out that some fallen leaves flow out of a river into the coast, and it is decomposed by work of bacteria etc., and returns to inorganic carbon. Therefore, measurement of the number of mangrove leaves which flow out of a river is one of the important factors for grasping the final amount of CO_2 fixation of mangrove forests [1](fig. 1). Since conventional counting methods (such as the litter trap method) require considerable manual labor for monitoring, an efficient counting method was needed.

S. Chawla et al. (Eds.): PAKDD 2008 Workshops, LNAI 5433, pp. 203–212, 2009.

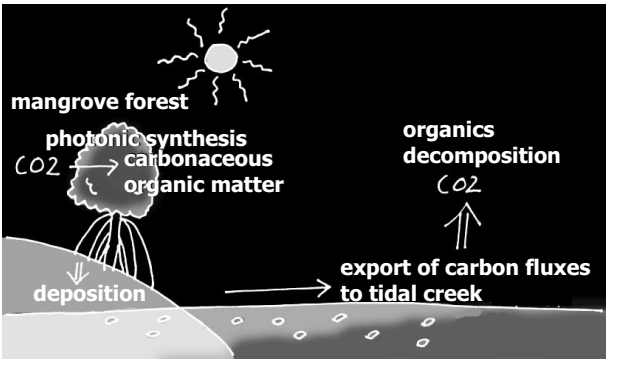


Fig. 1. Fringe forest carbon budget

We propose a count method for leaves on the surface of a river to be used in ecosystem monitoring. The method automatically counts the number of floating leaves in recorded video images using color and motion features. Conventionally, it was difficult to carry out automatic measurement of the number of natural objects using color information. The difficulty is caused by the fact that natural objects have various colors, and also because the colors change frequently with available light. In order to use the various colors of the targets, a structure which can memorize the various colors correctly is required. Furthermore, a support tool with which a human can easily teach the colors of the targets to a computer is also required.

2 Related Works

Numerous object tracking and recognition algorithms in outdoors have been proposed in the computer vision community. Especially many tracking methods for cars and pedestrians in outdoor environments have been proposed in recent years for the needs of security and ITS (intelligent transport systems) [2,3,4,5,6]. However, the algorithm which can track arbitrary objects in arbitrary environments does not exist, because it is difficult to find out the appearance invariant of target objects in outdoors.

For the robust tracking during the change of the lighting environment, many conventional techniques adopt motion, shape or texture feature for the pedestrian [2,3,4], and the edge or texture feature for the automobile [5,6]. On the other hand, in the tracking of floating leaves, although many motion features of leaves are common, shape and texture of them have few common features. Furthermore, since the river always changes its surface under the influence of wind and reflection, the conventional tracking techniques for cars or pedestrians cannot distinguish the leaves from the surface of a river and tracking cannot be carried out correctly.

Since there is a common color feature in the floating leaves, it is possible to apply many tracking methods based on color features to detect human's faces

and hands [7,8]. However, the methods are premised on indoor environment, and there is a limitation of the number of the tracking objects. On the other hand, the color of a leaf changes with the differences in change of the weather, and the angle of a leaf frequently. Moreover, since there are many color variations of leaves, it is not easy to find out the common color feature of the huge number of leaves.

As an application in the problem similar to the proposal technique, the technique of counting the number of fruits is proposed [9]. The system can automatically count the number of oranges in farm from the images captured with the panning camera. In the system, common color feature of oranges is manually defined and used in a partial space separated with several discriminative plains in a color space. Since the definition of color is too simple, by this technique, only about 76% of counting accuracy has been attained to the only 59 images.

Several techniques of recognizing and retrieving kinds of leaf are also proposed [10,11]. Since those techniques aim at the recognition of a leaf in closeup pictures, and they use the shape of the leaf as an important feature, the techniques are not applicable to the floating leaf detection.

3 A Method to Recognize and Count Leaves on the Surface of a River

3.1 Extracting Candidates of Leaves Using Color Filter

Our count method extracts the candidate of leaves from video images using the color characteristic of leaves at first. However, the filtering does not necessarily extract the whole pixels of the leaves. The leaves contain the same color as floating non-targets, reflection of the water surface, etc. Therefore, if it is going to extract the whole pixels of the leaves, many things which are not leaves will be also extracted and it will become a cause of incorrect counting.

Our method uses the color filter using three-dimensional color space, for example RGB color space. A three-dimensional color space is classified in the form of lattices, and the frequency (n_{pos}) where it appears as color of leaves, and the frequency (n_{neg}) where it appears as color which is not leaves are totaled for every lattices. Therefore, color values corresponding to the lattice with big ($n_{pos} - n_{neg}$) express color characteristics of the leaves.

The color filter is applied to video images as follows (fig. 2). At first, all pixels of the color value corresponding to the positive bins ($n_{pos} - n_{neg} > 0$) pass the filter and the connected pixels are grouped as a candidate area. Because a leaf area is generally divided into two or more scattered partial areas, the areas within a fixed distance are grouped. Calculating the minimum rectangle which includes the combined area inside, the method extracts a leaf candidate. Fig.3 is a processed screen shot of a video. Leaves extracted with the color filter are indicated by small rectangles.

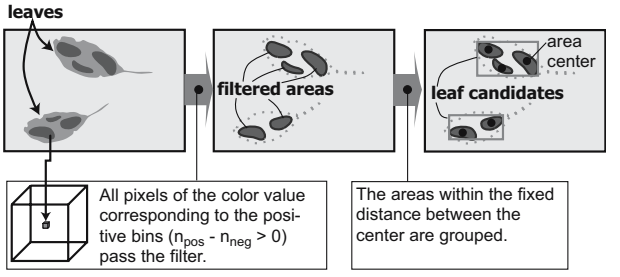


Fig. 2. Process of filtering and detecting candidates

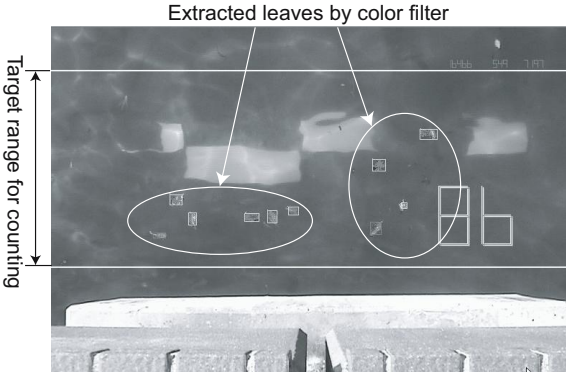


Fig. 3. A screen shot of extracting leaves by color filter

3.2 User Interface to Acquire the Color Information of Leaves

We have developed a user interface based on the Interactive Machine Learning model (IML)[12]. IML prepares the software environment where the user can label data easily (indicating training samples and instructing correct answers). Moreover, if the user changes the labels, the recognition result based on the labels is immediately visualized. The user can detect suitable training data which should be labeled by using the direct manipulation style user interface.

The mechanism of the developed user interface is shown in Fig.4. (1) First, the user indicates a leaf on the screen in the same way as coloring a picture with a mouse-dragging operation. The mouse-dragging operation has two modes, painting mode and elimination mode. The user can indicate positive examples in the painting mode, and negative examples in the elimination mode. (2) The system increments the histogram bin corresponding to the color of the pixel which the user indicated as a positive example, and decrements for a negative one. (3) In the target range of the screen, all pixels of the color values corresponding to positive bins pass the filter. The filtered pixels are painted in red on the screen as soon as the histogram is changed. The user can easily check whether the system

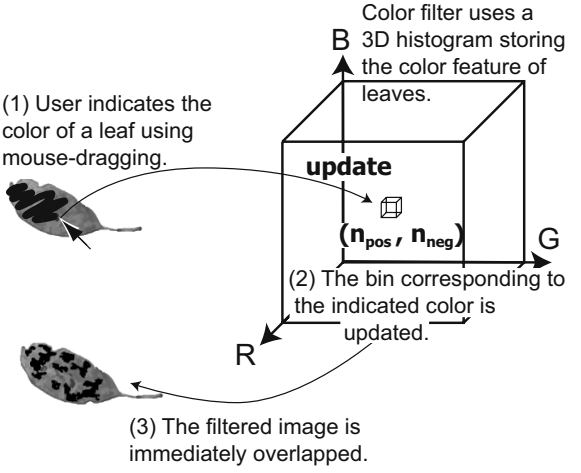


Fig. 4. Interaction mechanism of the user interface to indicate color information

can distinguish leaves appropriately, by looking at the screen which changes at every moment during the dragging operation.

Our user interface resembles a conventional color range tool which is adopted in commercial photo retouch softwares. However, since the tool has not memorized frequency, if user make a mistake in the specification at least one point, it exerts a big change on the filter. In our user interface, in order for a system to become wise gradually, color memory of the system is stabilized after the sufficient specification. Therefore, influence of mistakes decreases in the memory of the system, and the user comes to be able to indicate color information comfortably.

3.3 Counting Leaves Using Motion Feature

The proposal method tracks the extracted leaf candidates over two or more frames, in order to distinguish leaves from reflections of the sunlight in a water surface etc. Since the reflections similar to the color of leaves do not move synchronously with the flow of a river, our method can avoid incorrect calculation by the tracking.

The outline of tracking and counting leaves is shown in Fig.5. The center of a leaf candidate at frame t is (x_t, y_t). The method looks for the nearest candidate within v_y in the direction of the lower stream at frame $t + 1$. The v_y value is the upper limit of the flow velocity for one frame (1/30 seconds). The system decides that the candidate which is possible for tracking in n frames is a leaf as the target of measurement. The values of v_y and n are set up by the user in advance.

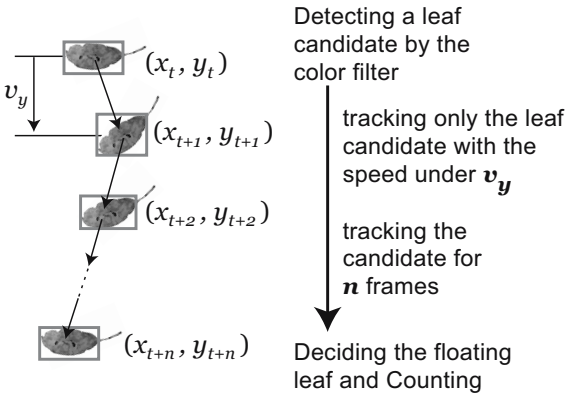


Fig. 5. Tracking and counting of leaves

4 Evaluation Using Five Hours Video Which Recorded 20,000 Leaves

The counting method proposed has been applied to three videos (total five hours) which recorded about 20,000 leaves. The videos were recorded at Fukido river, Okinawa prefecture, in Japan.

We used RGB color space as the color space for the experimental, since the RGB color space is the most commonly used color space. The each number of quantization for R, G and B axes was set to 20. Besides RGB, many color spaces, such as Yxy, HSV, L*a*b*, and L*u*v* have been proposed. The performance evaluation of the technique by the difference in color space is a future subject. The number of quantization influences the speed of response of the color specification user interface. The quantization number (= 20) was set up so that the direct manipulation feature of the user interface might not be lost. If the number of quantization is increased, color filter can distinguish delicate differences in color, but the color information which user should specify increases. Evaluation of influence of the quantization number is also a future subject.

The parameters in the tracking step were determined from the preliminary experiment using a part of sample video. The maximum move pixel (v_y) to the direction of the lower stream in tracking of leaves was 12 pixels. The number of tracking frames (n) was set to 7. In the case of the computer used for the experiment[1], processing speed of automatic counting was about 30 minutes to the examination video for 120 minutes.

Fig.6 illustrates a three-dimensional histogram created using the developed user interface. This histogram was created by the user's indication operation to 4 minutes video (1 minute every 30 minutes from a two hour video). The colors

[1] OS: Windows XP Professional sp2, CPU: Intel Core2Duo E6600 2.4GHz, Memory: 2GB.

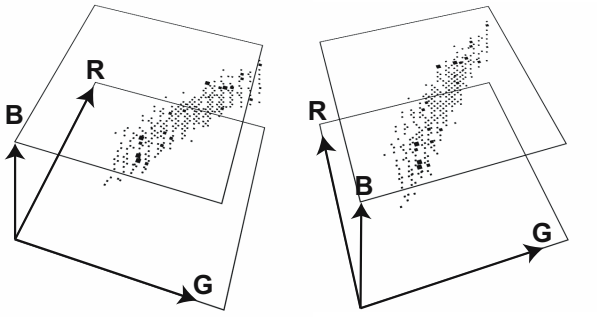


Fig. 6. Color distribution of mangrove leaves in the RGB color space

Table 1. Precision and recall rates of automatic counting

Date	Precision	Recall
2004/7/30	80%	91%
2004/7/31	98%	97%
2004/8/1	96%	97%

of the mangrove leaves are intricately and discontinuously distributed from dark brown to bright yellow in the color space.

For comparison, two human subjects counted the leaves manually using slow playback and stop motion function, taking a long time[2].

Fig.7 illustrates the automatic counting results using color indication of only 1 minute video. These figures show that correct counting is difficult when the system is trained by the color indication based on each 1 minute video. The cause of the difference between the result of July 30 and the other result is the change of situation of a river. 82 minutes just before tide pulls are recorded on the video on July 30. Although there is some depth of water and the water is flowing in the first half of the video, the flow of the river stopped in the second half of the video, and it has resulted in the state where it is exposed of a part of river bottom for a short time.

From the result of other videos in Fig.7, we can find that the exact counting is possible using only one minutes indication, if we can choose the appropriate time zone to indicate the color of leaves. However, we can not find out the appropriate time zone before crawling the whole of video.

Fig.8 is the experimental results of automatic count with the color indication using several minutes of video. The automatic count method showed results mostly in agreement with the manual counting results. We calculated precision and recall rates by considering the manual measurement as the correct answer. Table 1 shows the precision and recall rates for each example videos. The total precision and recall rates for the five hour video are 96% and 94%, respectively.

[2] Counting by the subjects took several weeks for the five hour video.

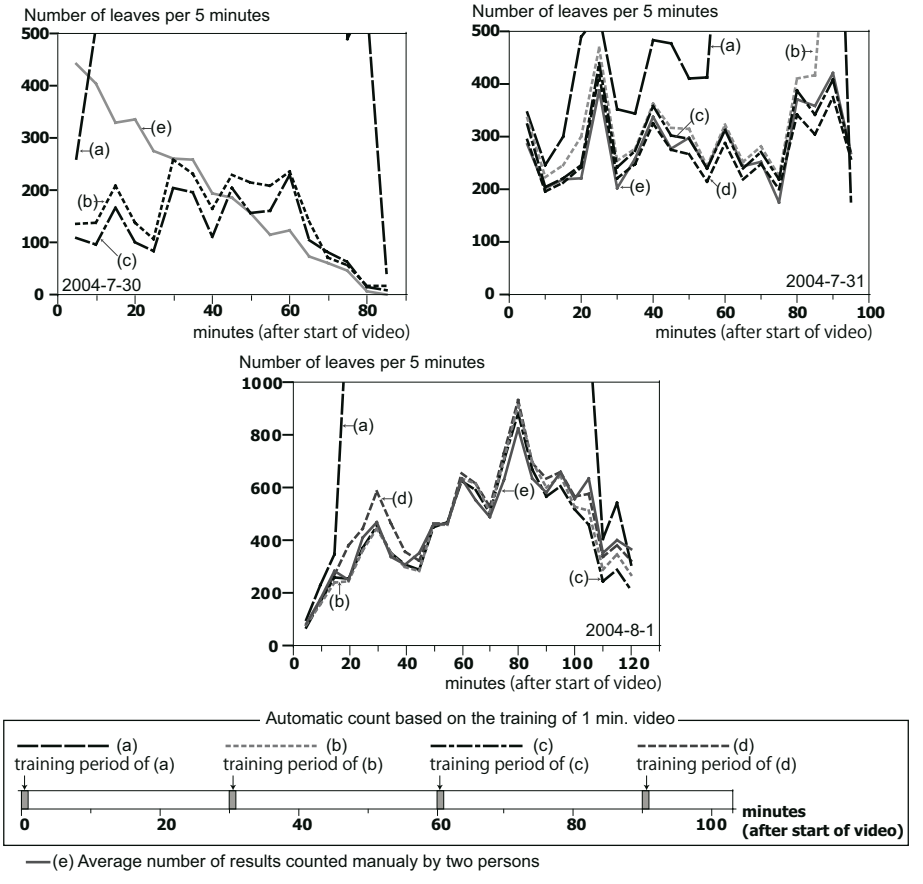


Fig. 7. Results of automatic count for video (case of 1 min. training)

Through the user interface, the average number of registered pixels is 23,190 pixels per one minute video clip, and the color instruction operation for a 1 minute video took an average of 7 minutes. This means that the proposed user interface is efficient for data input, as a large number of color samples could be registered in a short time. Moreover, since the information resulted in correct counting, the user interface is applicable when building effective color samples for video.

In the tracking phase, when training of color is not enough, the counting system has to track a lot of wrong candidates. In such case, the tracking error occurred frequently. Since these errors are not reduced even if parameters of the tracking are changed, these errors cannot be avoided by the simple tracking algorithm that we used. On the other hand, there were few tracing errors in the stage with sufficient training (such as 4 min.). However, when two or more leaves overlapped and separated, some tracking errors occurred. If pattern matching technology is added to the tracking algorithm, tracking errors may be able to be

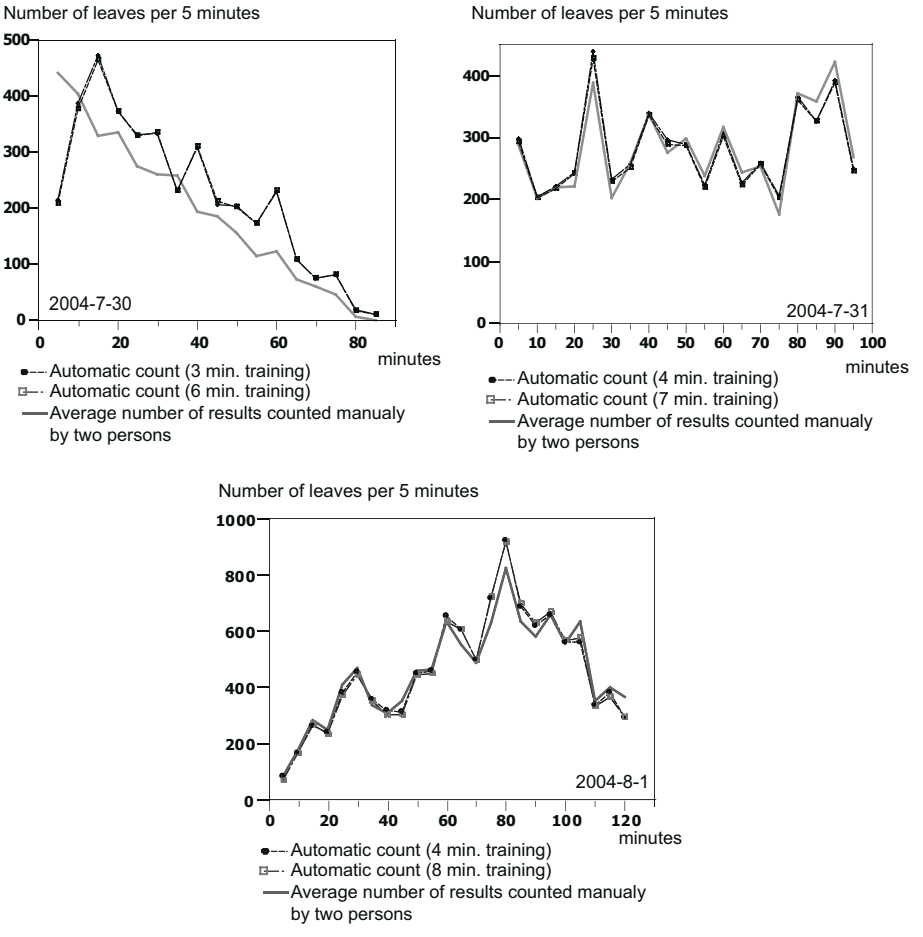


Fig. 8. Results of automatic count for video (case of multiple minutes training)

reduced even when training is not enough. Moreover, the method of optimizing tracking parameters (v_y and n) suitable for a new video automatically is required practically. We are planning to tackle these as future subjects.

5 Conclusion

In this paper, an automatic video counting method of the leaves on the surface of a river for an ecosystem monitoring was proposed. The method uses the color and motion feature of the leaves to detect them. The system using the method makes user specify the color feature of the leaves by using a user interface based on the Interactive Machine Learning model. User can easily indicate the color information to the system by using the interactive software tools. Based on the color information which the user specified, a color filter is constructed and

leaf candidates are extracted by the filter from video data. Furthermore, after tracking the motion of the candidates, the method counts the leaves.

The method proposed in this paper has been applied to three videos (total five hours) which recorded about 20,000 leaves, and high recall and precision rates of 96% and 94%, respectively, have been achieved.

Although the proposed method achieved high accuracy as a whole, changes in weather and river flow causing rapid visual changes could result in deviating from the built color information. This shows the necessity for a mechanism which can automatically catch changes in the video, and request the user's input for a new color set.

References

1. Ong, J.E.: Mangroves: a carbon source and sink. Chemosphere 27(6), 1097–1107 (1993)
2. Lanz, O.: Approximate Bayesian multibody tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence 28(9), 1436–1449 (2006)
3. Viola, P., Jones, M.J., Snow, D.: Detecting Pedestrians Using Patterns of Motion and Appearance. International Journal of Computer Vision 63(2), 153–161 (2005)
4. Wu, B., Nevatia, R.: Detection and Tracking of Multiple, Partially Occluded Humans by Bayesian Combination of Edgelet based Part Detectors. International Journal of Computer Vision 75(2), 247–266 (2007)
5. Dahlkamp, H., Nagel, H.H., Ottlik, A., Reuter, P.: A Framework for Model-Based Tracking Experiments in Image Sequences. International Journal of Computer Vision 73(2), 139–157 (2007)
6. Kamijo, S., Nishida, T., Sakauchi, M.: Occlusion Robust and Illumination Invariant Vehicle Tracking for Acquiring Detailed Statistics from Traffic Images. IEICE TRANSACTIONS on Information and Systems E85-D(11), 1753–1766 (2002)
7. Medionia, G., Francoisa, A.R.J., Siddiquia, M., Kima, K., Yoonb, H.: Robust real-time vision for a personal service robot. Computer Vision and Image Understanding 108(1-2), 196–203 (2007)
8. Polat, E., Yeasin, M., Sharma, R.: Robust tracking of human body parts for collaborative human computer interaction. Computer Vision and Image Understanding 89(1), 44–69 (2003)
9. Annamalai, P., Lee, W.S., Burks, T.F.: Color vision system for estimating citrus yield in real-time. In: ASAE Annual International Meeting, Paper Number: 043054 (2004)
10. Im, C., Nishida, H., Kunii, T.L.: Recognizing Plant Species by Leaf Shapes - A CaseStudy of the Acer Family. In: Proc. of ICPR 1998, pp. 1171–1173 (1998)
11. Wang, Z., Chi, Z., Feng, D.: Shape based leaf image retrieval. Proc. IEE Vision Image and Signal Processing 150(1), 34–43 (2003)
12. Fails, J.A., Olsen Jr., D.R.: Interactive machine learning. In: Proc. of the 8th International Conference on Intelligent User Interfaces, pp. 39–45 (2003)

Author Index