Incremental Set Recommendation Based on Class Differences

Yasuyuki Shirai[1], Koji Tsuruma[1], Yuko Sakurai[2], Satoshi Oyama[3], and Shin-ichi Minato[1,3]

[1] JST-ERATO MINATO Discrete Structure Manipulation System Project, Hokkaido University, Sapporo, Japan
{shirai,tsuruma}@erato.ist.hokudai.ac.jp
[2] Graduate School of Information Science and Electrical Engineering, Kyushu University, Fukuoka, Japan
ysakurai@inf.kyushu-u.ac.jp
[3] Graduate School of Information Science and Technology, Hokkaido University, Sapporo, Japan
{oyama,minato}@ist.hokudai.ac.jp

Abstract. In this paper, we present a set recommendation framework that proposes sets of items, whereas conventional recommendation methods recommend each item independently. Our new approach to the set recommendation framework can propose sets of items on the basis on the user's initially chosen set. In this approach, items are added to or deleted from the initial set so that the modified set matches the target classification. Since the data sets created by the latest applications can be quite large, we use ZDD (Zero-suppressed Binary Decision Diagram) to make the searching more efficient. This framework is applicable to a wide range of applications such as advertising on the Internet and healthy life advice based on personal lifelog data.

Keywords: recommendation, classification, collaborative filtering, zero-suppressed binary decision diagram

1 Introduction

Several techniques on information filtering and information recommendation such as collaborative filtering and content-based filtering have been reported [5][1][11]. In conventional collaborative filtering, items are recommended on the basis of their relevance to the user's preferences. Each item is recommended independently of the others; that is, the relationship of a recommended item to the other items is not considered.

In the real world, however, a user is often interested in a combination of items, such as the keywords in an advertisement and the places to be visited during a sightseeing tour. Recently proposed *set recommendation* techniques[12, 10] consider the unit of recommendation to be a set of items and the constraints and requirements among them.

In this paper, we extend this approach to incorporate the use of an algorithm to present recommendations for modifying the user's initially chosen set. In our *incremental set recommendation* framework, it is assumed that each record ("item set") in a database has been classified as a class such as positive/negative, and modifications are recommended that would change the item set so that it matched the target classification.

An example application of our framework is a recommendation system that uses a database in which the action history data for a group of people are stored. The data could be exercise history or dietary behavior, for example. Each person in the database is classified as either a success or failure w.r.t. to some target (e.g., weight loss). Those in the "failure" group could use the system to obtain recommendations for specific behavior improvements that are based on the data for those in the "success" group. The recommendations are made on the basis of the differences between the two groups and should change the user's actions and lifestyle as little as possible. "Behavior improvements" for the exercise history example means the addition and/or deletion of item sets representing the type and amount of exercises performed, while for the dietary behavior example, it means the addition and/or deletion of item sets representing the type and quantity of food eaten. Another example application is a system for describing the items for sale on an Internet shopping site. The descriptions of poorly selling items would be modified on the basis of the descriptions of items that sell well.

The rest of the paper is organized as follows: Section 2 gives the basic definitions. In Section 3, we describe the implementation of our framework using a Zero-suppressed Binary Decision Diagram (ZDD) data structure. We present and discuss the results of its evaluation in Section 4. We conclude in Section 5 with a brief summary, some additional comments, and a mention of future work.

2 Definition

We will provide some definitions and notations as follows :

Definition 1 (item). *An item is an atomic entity that represents a characteristic or feature and is denoted by a lower-case character,* $a, b, c, \ldots$. *A set of all items to be considered is denoted by* Σ.

In the exercise history example, each item could be the name of an exercise.

Definition 2 (data record and class). *A data record is a collection of items that represent the attributes or characteristics of the target object (we use* D *to represent a data record). A class is a name for a set of data records, and is denoted by* α, β, γ, ω *or* ϕ. *Each data record belongs to only one class.*

"Positive" or "Negative" is an example of a class.

Definition 3 (pattern set/class membership). *A pattern set is a set of pairs, each of which consists of an item set and its weight (natural number). If the weight values are all the same, they can be omitted. A pattern set is denoted*

by C_ω where ω is the class identifier ($C_\omega = \{p : w_p | p \in 2^\Sigma, w_p \in \mathbb{N}\}$). If $q : w_q \in C_\omega$ (simply we write $q \in C_\omega$), q is called a pattern of class ω.

Example : Let $C_\alpha = \{\{a, b, c\} : 2, \{a, b, d\} : 1, \{b, c, d\} : 3\}$. $\{a, b, c\}$ is a pattern of class α, whereas $\{a, b\}, \{a, b, c, d\}$ and $\{b, d, e\}$ are not. We sometimes use polynomial notation for pattern set such as $C_\alpha = 2abc + abd + 3bcd$. A difference of two pattern set, such as $C_\alpha - C_\beta$, is defined as a difference of polynomials.

Definition 4 (removable/universal/addable items). *For a data record D, the items in D can be divided into removable items (D^-) and universal items (D^*). The recommendation algorithm can suggest the deletion of items only if they are elements of D^-. A set of addable items, denoted by D^+, is the set of items that can be added to D.*

Universal items intuitively correspond to essential features of the record. If no universal item is specified, $D^* = \emptyset$.

Definition 5 (delete/add-constraints). *The upper bounds on the numbers of items that can be added or deleted for a data record D are denoted by N^D_{add}, N^D_{delete}, respectively.*

Definition 6 (recommendation candidate). *For a data record $D \in C_\phi (D \notin C_\omega)$, if there could be a candidate $D' \in C_\omega (D' \notin C_\phi)$ that is a modification of D by adding and/or deleting items under the conditions of N^D_{add} and N^D_{delete}, and if the weight of D' in $C_\omega - C_\phi$ is equal to or greater than the given natural number M (called weight condition), D' is called a recommendation candidate for class ω that satisfies N^D_{add}, N^D_{delete} and M.*

Example : Let $N^D_{add} = 1$ and $N^D_{delete} = 1$ for $D = \{a, b, f\}$ and weight condition $M = 2$ in the above example, $D' = \{a, b, c\}$ is a recommendation candidate for class C_α.

3 Set Recommendation based on Class Differences

In general, the number of instances of each class could be quite large. For example, the number of articles for sale or the number of customers on a major Internet shopping site could reach several million. As another example, lifelog services using mobile devices generates enormous amount of data in recent years. To handle such huge numbers of data records, we use a ZDD (Zero-Suppressed Binary Decision Diagram) data structure. In this section, we first present an example of our set recommendation framework based on class differences. We then briefly introduce ZDD and our recommendation algorithm which uses the ZDD structure.

3.1 Example

Suppose we have pattern sets for classes α and β as follows:

$$C_\alpha = \{\{a,b,c\}:1, \{b,c\}:2, \{c\}:1, \{d,e\}:2, \{e\}:3\} \quad (1)$$
$$C_\beta = \{\{a,c\}:1, \{a,b,d\}:1\} \quad (2)$$

Suppose further that $\Sigma = \{a,b,c,d,e\}$ and $D = \{a,c\}$ ($D \in C_\beta$). The recommendation for D would consist of the following candidates (D') under the condition that $N^D_{add} = N^D_{delete} = 1$ and the weight condition $M = 1$: "Add b", "Delete a and add b", "Delete a". After those modification, D' would be identified as class α, rather than as class β. If we restrict the recommendation so that $M = 2$, we get only "Delete a and add b".

In this work, we use a VSOP (Valued-Sum Of Products) calculator based on ZDD for calculating the recommended items. We review ZDD and VSOP briefly in the next subsection.

3.2 ZDD and VSOP

Binary decision diagrams[2, 4] (BDDs) are well-known and widely used for efficiently manipulating large-scale Boolean function data. A BDD is a directed graph representation of the Boolean function. The reduction rules in BDD consist of "node deletion rule" (delete all redundant nodes with two edges that point to the same node) and "node sharing rule" (share all equivalent sub-graphs).

ZDDs (Zero-suppressed BDDs) [6, 4] are special type of BDDs which are suitable for implicitly handling large-scale combinatorial item set data. The reduction rules of ZDDs are slightly different from those of BDDs. They are illustrated in Fig. 1 (a).

- Share equivalent nodes as well as ordinary BDDs.
- Delete all nodes whose 1-edge directly points to the 0-terminal node, and jump through to the 0-edge's destination.

ZDDs are especially more effective then BDDs for representing "sparse" combinations such as purchase history data. For instance, sets of combinations selecting 10 out of 1000 items can be represented by ZDDs up to 100 times more compactly than by ordinary BDDs.

VSOP (Valued-Sum-Of-Products Calculator)[7] is a program developed for calculating a combinatorial item set where each product term has a value, specified by symbolic expressions based on ZDD techniques. The value of each product can also be considered as a coefficient or a weight for each term. For example, the formula $(5abc + 3ab + 2bc + c)$ represents a VSOP with four terms abc, ab, bc and c, each of which is valued as 5, 3, 2, and 1, respectively. VSOP supports numerical arithmetic operations based on Valued-Sum-Of-Products algebra, such as addition, subtraction, multiplication, division, and numerical comparison. The details of the algebra and arithmetic operations of a VSOP calculator are described in [6, 7].

When dealing with integer values in binary coding, we have to consider the expression of negative numbers. VSOP adopted another binary coding[8] based on (-2), namely, each bit represents $1(=(-2)^0), -2(=(-2)^1), 4(=(-2)^2), -8(=(-2)^3), 16(=(-2)^4), \ldots$. For example, -12 can be decomposed into $(-2)^5 + (-2)^4 + (-2)^2$. In this encoding, each integer number as a coefficient can be uniquely represented.

Fig. 1 (b) shows the example of the VSOP representation for $abc - ac + 2bc + c + 2de + 3e - abd$. Since ac satisfies the top nodes labeled $+1$ and -2, the coefficient of item ac can be calculated by $+1 - 2 = -1$.

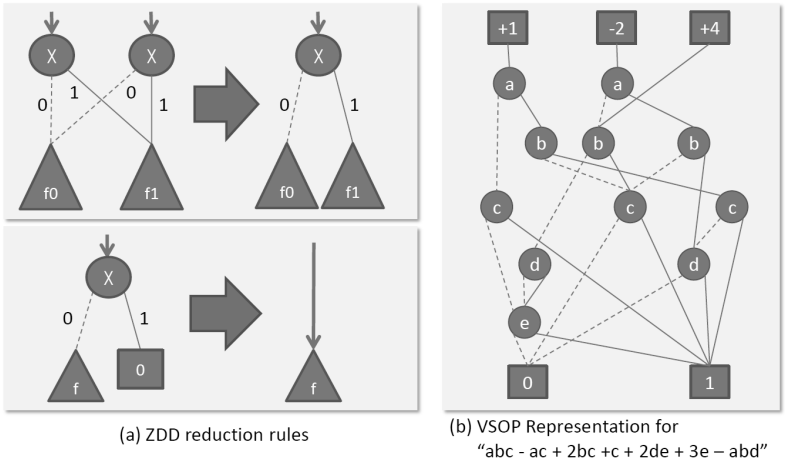

Fig. 1. ZDD Representation

3.3 Set Recommendation with ZDD Structure

In this subsection, we show the method for calculating a set of items to be recommended using ZDD. We first consider the following polynomials (valued sum of products) for (1) and (2) in Section 3.1:

$$C_\alpha - C_\beta = abc - ac + 2bc + c + 2de + 3e - abd$$

For a given D, we have to output a set of terms from $C_\alpha - C_\beta$, that are modification of D under the constraints of N_{add} and N_{delete}, and whose coefficients are equal to or larger than given integer M.

Fig. 2 shows an example search process on the ZDD structure for $C_\alpha - C_\beta$, where $N_{add} = N_{delete} = 1$ and $D = ac$. In this figure, the search process starts with each top node $+1, -2, +4$ respectively and then item sets satisfying the constraints are extracted for each top node $+1, -2, +4$. The pair of numbers

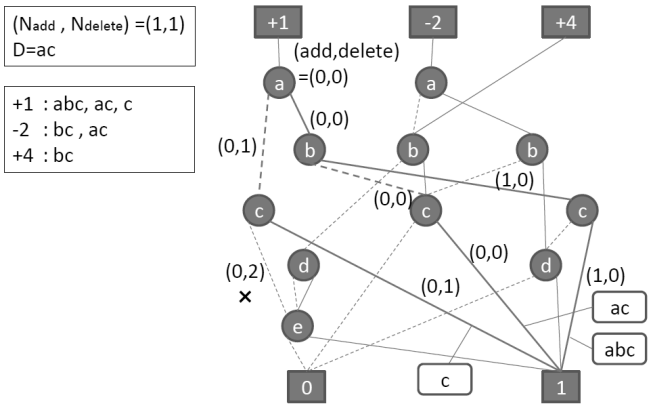

Fig. 2. Search on ZDD Structure

for item addition and deletion is attached to each edge, as shown in Fig. 2. If the pair does not satisfy condition N_{add} or N_{delete}, searching along that path is terminated. For example, since the pair on the edge from c (left side in Fig. 2) is $(0, 2)$, which does not satisfy the N_{delete} condition, searching along the path below that node is terminated.

Item sets that need to be found under the condition of $M = 2$ must satisfy one of $(0, 0, 1), (1, 0, 1), (0, 1, 1)$, or $(1, 1, 1)$ for the top nodes $(+1, -2, +4)$ in Fig. 2. For example, suppose $D = ac, N_{add} = 1$ and $N_{delete} = 1$. Since the numbers of added items and deleted items w.r.t bc are 1 and 1 respectively, and since bc satisfies $(0, 1, 1)$ for the top nodes $(+1, -2, +4)$, bc is a recommendation candidate under the condition of $M = 2$. As another example, since the numbers of added items and deleted items w.r.t abc are 1 and 0 respectively, and since the candidates abc satisfies $(1, 0, 0)$ for the top nodes $(+1, -2, +4)$, abc could be a recommendation candidate under the condition of $M = 1$ as well as bc described above. By the same way, c is also a recommendation candidate under the condition of $M = 1$

The naive search algorithm on a ZDD structure is shown in Algorithm 1.

4 Experiments

We first evaluate the efficiency of our approach based on ZDD, using artificial data, and then we show the examples using actual Internet application data.

4.1 Performance Evaluation

The problem we provided for performance evaluation in this experiment consists of 170 items in total ($|\Sigma| = 170$), and each record contains 5 items. There are two classes: positive and negative.

Algorithm 1 Naive Search Algorithm on ZDD structure

Given D (target items), N_{add}, N_{delete} (upper limits of number of "add" items, "delete" items), $M(\geq 1)$ (weight condition), ZDD (ZDD structure whose top nodes are $1, \ldots, N$ (ex. -1,-2,4,...)).

for $i = 1$ to N **do**
  initialize $(path_i = \{\}, AddList = \{\}, DeleteList = \{\})$
  $L = L + get\_candidate(path_i, i, AddList, DeleteList)$
end for
$merge\_output(L, M)$ (merge the results for each node $(1, \ldots, n)$ and output the results whose coefficients $\geq M$)

Function $get\_candidate(path, n, AddList, DeleteList)$ {n is a node of ZDD}
if $|DeleteList| > N_{delete}$ or $|AddList| > N_{add}$ **then**
  return *null*
else if n is a terminal node 1 **then**
  return *path*
else if n is a terminal node 0 **then**
  return *null*
else
  let n_0, n_1 : node which is connected by 0 edge, 1 edge of node n, respectively.
  if $n \in D$ **then**
    $get\_candidate(path, n_0, AddList, DeleteList + \{n\})$
    $get\_candidate(path + \{n\}, n_1, AddList, DeleteList)$
  else
    $get\_candidate(path, n_0, AddList, DeleteList)$
    $get\_candidate(path + \{n\}, n_1, AddList + \{n\}, DeleteList)$
  end if
end if
End Function

There are two execution scenarios: one used a random data set, and the other used a fixed pattern data set. In the random data set, items occurs randomly in each record; in the fixed pattern data set, fixed positive and negative patterns were prepared (20 patterns for each class), and each pattern consisted of three items.

In the fixed pattern data set, three items in each record are taken from the fixed patterns, and two are taken from the random patterns. In actual applications, such as an Internet purchase history, there would be some fixed patterns in both classes. In this experiment, we used three data set sizes for each class: 1, 5, and 10 million records.

The system was implemented in Java, and the experiments were run on SUSE Linux Enterprise Server 10 with a quad-core AMD Opteron 3 GHz CPU, and 512 GB RAM. The execution times are shown in Table 1. The times shown are an average for ten trials. In the tables, "sequential search" in which all items in each record were ordered and stored in memory was done for comparison.

Table 1. Experimental Results for Performance Evaluation

(a) For Random Data Set (Time : msec)

	ZDD-based Search			Sequential Search		
	1M	5M	10M	1M	5M	10M
$N_{add} = 1, N_{delete} = 1, M = 1$	13	12	17	63	255	514
$N_{add} = 2, N_{delete} = 2, M = 1$	16	27	47	70	349	652
$N_{add} = 3, N_{delete} = 3, M = 1$	72	326	522	92	496	1152
$N_{add} = 3, N_{delete} = 3, M = 2$	74	308	521	93	513	1123
$N_{add} = 3, N_{delete} = 3, M = 3$	73	328	541	98	502	1163

(b) For Fixed Pattern Data Set (Time : msec)

	ZDD-based Search			Sequential Search		
	1M	5M	10M	1M	5M	10M
$N_{add} = 1, N_{delete} = 1, M = 1$	6	5	7	72	279	564
$N_{add} = 2, N_{delete} = 2, M = 1$	6	8	9	79	384	609
$N_{add} = 3, N_{delete} = 3, M = 1$	7	9	13	94	450	784
$N_{add} = 3, N_{delete} = 3, M = 2$	7	9	17	95	419	769
$N_{add} = 3, N_{delete} = 3, M = 3$	7	9	15	94	424	776

With the random data sets, there were no substantial differences between the ZDD-based search and the sequential search. This is because a ZDD data structure is not much more compact or efficient rather than a flat data structure. The number of ZDD nodes for the random data sets were respectively 2696527, 11789288, and 20738481. Their relationship is almost linear. In contrast, with the fixed pattern data sets, there were marked differences between the two searches. The ZDD-based search was more efficient due to the compact representation by ZDD. The numbers of ZDD nodes for the fixed pattern data sets were respectively 313594, 363112, and 377979. These data sets were relatively small and did not linearly increase in size. This was reflected in the total execution times.

In actual applications, there are usually fixed patterns in item occurrences for each class. Although actual application data are not as strongly biased as in our experiment, we can nevertheless conclude that the ZDD-based search approach is well suited for actual applications.

4.2 Example : Internet Shopping Advertising

As one Internet application, we used data from Rakuten Ichiba [13], the largest online shopping mall in Japan, with over 30,000 online stores (September 2009). The company has released some of their data for use in academic research[13].

We investigated the relationship between article descriptions at the time of article introduction for sale and the number of user reviews attached to each article. The descriptions had been written (in Japanese) by a person working for the shop where the article was to be sold. Article descriptions are important because they attract customers through on-line searching.

The data fields in the original data are shop code, purchase article id, article name, article description, price, category, and number of reviews. There are 31 top categories in total. We used data labeled with the category "Ladies Fashion" and "Japanese Sake" (liquor).

In order to define characteristics from each description, we first extracted the nouns, adjectives, verbs, and adverbs from the descriptions using the Japanese language morphological analysis program called Chasen[4]. After some modifications (n-gram word concatenation to generate collocation, word selection by $TF \times IDF$ measurement, etc.), the item sets (Σ) were defined. The explanatory variables for each article consists of the occurrences of the selected words in the article descriptions. The class of the article was determined by the number of reviews. That is, we classified an article "positive" if it had more than two reviews and "negative" if it had no reviews.

In the Japanese Sake category, there were 517 items in total ($|\Sigma| = 517$), 3085 positive records, and 3372 negative records. Each record generally had five to ten items. We set N_{delete} and N_{add} to respectively 2 and 4. In the Ladies Fashion category, there were 1475 items in total ($|\Sigma| = 1475$), 3166 positive records, and 7194 negative records. Both N_{delete} and N_{add} were set to 3.

Table 2 shows some of the results translated from the original Japanese. We assumed that all items in D (original item set) were removable (i.e., it did not contain any universal items) and set weight condition M to 1. For the Japanese Sake category, we found that keywords that created attractive images were preferred rather than technical keywords such as "rice malt" and "carefully screened." These results reflect the Internet shopping situation for Japanese Sake; that is, people who like to buy Japanese sake on the Internet generally put more emphasis on the image or feeling of drinking sake rather than the technical details, unlike those who buy it in actual shops. For the Ladies Fashion category, keywords giving specific descriptions of each article, such as "tiered skirt," "hemline," and "shoulder strap adjustment" were preferred to ones that created a visual image of their usage or that described the technical details. That is, people shopping on the Internet for fashion prefer specific images and specifications, unlike people shopping in actual shops.

4.3 Example : AOL Search Logs

As the other Internet application, we used data from AOL's Search Log Collection [14]. This collection consists of about 21 million web search queries input by about 650,000 AOL users from March to May 2006. The records include 'Query,' 'ItemRank,' and 'ClickURL' (the last two items were included only if the user had clicked on a search result).

We used records in which there was a 'ClickURL' entry and the 'ItemRank' was less than 5 as the positive data and those without a 'ClickURL' entry as negative data. The objective of this analysis was to present sets of items to be

[4] http://chasen.naist.jp

Table 2. Example : Internet Shopping Advertising

Category	Original	Added Items	Deleted Items
Sake	rice malt, production area, box, gift	plum brandy, woman	rice malt, production area
	low temperature, slow, distillation, actual producer, distilled spirit, flavor , production area	rich, black malt, tasty	low temperature, slow, distillation
	river-bed water, carefully-screened, actual producer, distilled spirit, characteristics, production area, flavor, tasty	deepness, barley distilled spirit	river-bed water, carefully-screened, characteristics
	cold storage, representative, refined sake, barm, acid degree	bright, limited, flavor	representative, refined sake
	recommend, cold storage, barm, father, acid degree	wonderful, brand sake	recommend, father
Ladies	skirt, casual, polyurethane, hip, real scale	tiered skirt, appeal	casual, polyurethane, hip
	shopping, travel, event, import	hemline, casual	shopping
	love, casual, travel, event, import	shoulder strap adjustment, beautiful leg	love, casual, travel
	boot-cut, straight, pants, polyurethane, silhouette, body	camel-hair, autumn and winter	pants

deleted from or added to the original queries so as to increase the likelihood that the user would click on a search result.

The results are shown in Table 3 (words preceded by "*" are universal items). We found that, if the user wants to find specific information about travel, it would be better to add a specific place-name such as europe, south africa, or italy. From the last example in the table, the keyword 'cheap' would not be adequate if the user wanted a reasonable price. Better keywords would be 'discounts,' 'best,' and 'packages.'

These results shows that our recommendation flamework can suggest possible candidates for query modifications, in order to get more appropriate search results for users.

5 Summary and Future Works

In this paper, we have described a new approach to the set recommendation problem: changes (item addition and deletion) to a set of items are recommended on the basis of class differences. Since recommendation services are becoming more and more popular, our framework should be effective for actual applications

Table 3. Example : AOL Search Logs

Original	Added Items	Deleted Items
adventure,*travel,blogs	tours,student	blogs
	africa	blogs
	family	blogs
	italy	adventure
	south,africa	adventure
*family,travel,vacations	europe	vacations
	packages,rome	vacations
	best	travel
	cheap	travel
	packages	travel
cheap,*europe,*vacation	discounts	cheap
	best	cheap
	packages	cheap

rather than simply being used for collaborative filtering. The use of our algorithms, which use the ZDD data structure, results in efficient calculation for huge data sets, especially when the data is biased, as it generally is in actual applications. Although we only considered the case of two classes for simplicity, we can easily consider a case in which there are three or more classes or there are multiple classification criteria for the input data.

In related work, Dong et al.[3] proposed using an *emerging patterns* approach to detecting differences in classes and using a classification framework based on the emerging patterns. While frequent pattern mining generally cannot detect the characteristic item pattern for each class, their approaches focus on detecting item sets that are meaningful for classification. Although their motivation is very similar to ours, they have not yet reported a recommendation procedure based on emerging patterns.

Other researchers have developed set recommendation procedures based on certain constraints such as recommendation costs, orders and other conditions[12, 10]. These procedures are practically applicable to trip advice and university course selection, for example. Although we do not assume any constraint between items as pre-defined knowledge, incorporating such constraints into our recommendation framework should improve its effectiveness.

Searching under the constraints described in this paper is closely related to searching based on the Levenshtein distance (edit distance). Efficient algorithms, such as dynamic programming approach, have been developed to calculate the distance, and many implementations including approximation approaches have been introduced [9]. The problem we focused on in this paper is slightly different from those for the edit distance. Our problem is to find similar items from given polynomials under the constraints of a limited number of item additions and

deletions with a weight constraint. A comparison of the problems remains for future work.

Future work also includes extending our results in several directions :

- The search procedure based on the ZDD structure described in this paper still contains redundant processes. Efficient search strategies such as using a cache of pre-searched results need to be investigated.
- We assume in this framework that items occur only positively in patterns. In actual applications, however, "don't care" plays an important role in recommendation. We need to investigate ways to incorporate such items.

Acknowledgment

We are grateful to Rakuten, Inc., for providing the Internet shopping data used in this research.

References

1. G. Adomavicius and A. Tuzhilin : Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions, IEEE Transactions on Knowledge and Data Engineering, 17(6), 734–749, 2005
2. R. E. Bryant : Graph-based algorithms for Boolean function manipulation, IEEE Transactions on Computers, Volume 35 Issue 8, August 1986
3. G. Dong , X. Zhang , L. Wong and J. Li : CAEP: Classification by Aggregating Emerging Patterns, Proc. of Second International Conference on Discovery Science, LNCS Vol. 1721, 1999
4. D. E. Knuth : The Art of Computer Programming, Volumue 4, No.1, Bitwise Tricks & Techniques, pp.117-126, Addison-Wesley, 2009
5. P. Melville and V. Sindhwani : Recommender Systems, Encyclopedia of Machine Learning, pp. 829–838, Springer, 2010.
6. S. Minato : Zero-Suppressed BDDs for Set Manipulation in Combinatorial Problems, In Proc. of 30th ACM/IEEE Design Automation Conference (DAC'93), 1993.
7. S. Minato : VSOP (Valued-Sum-of-Products) Calculator for Knowledge Processing Based on Zero-Suppressed BDDs, In K. P. Jantke, et al. editors, Federation over the Web, LNAI 3847, 2006.
8. S. Minato : Implicit Manipulation of Polynomials Using Zero-Suppressed BDDs, In Proc. of IEEE The European Design and Test Conference, 1995.
9. G. Navarro : A Guided Tour to Approximate String Matching, ACM Computing Surveys, Vol. 33, No.1, 2001
10. A. Parameswaran, P. Venetis and H. Garcia-Molina: Recommendation Systems with Complex Constraints: A Course Recommendation Perspective, Transactions on Information Systems, 2011 (To Appear)
11. X. Su and T. M. Khoshgoftaar: A survey of collaborative filtering techniques, Advances in Artificial Intelligence, 2009
12. M. Xie, L.V.S. Lakshmanan and P.T. Wood: Breaking out of the box of recommendations: From Items to Packages, In Proc. of the 4th ACM Conf. on Recommender systems, 2010
13. `http://rit.rakuten.co.jp/rdr/index_en.html` (Rakuten data disclosure)
14. `http://www.gregsadetsky.com/aol-data` (AOL search logs)