http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 9 (6): 1246-1250, 2010
ISSN 1812-5638

Frequent Browsing Patterns Mining Based on Dependency for Online Shopping

Qing Yang, Ping Zhou and Jingwei Zhang
Computer and Control Institute,
Guilin University of Electronic Technology, Guilin, 541004, China

Abstract: Internet sellers hope to let the consumers find their commodities easily, but there is a divergence between commodities presentation requirements and the limitation of a web page. In order to pursuit the profit-maximizing, a web page must contain the most welcome commodities. In this study, we proposed a multiple candidate set method to discover those most welcome commodities from users' browsing data. Our approach utilizes frequent dependent relationships among the commodities to partition the unique candidate set, composed of all single frequent item, into multiple sub-candidate sets for computing the frequent browsing patterns, which avoids the problem of searching through a large space of candidate set. Our experiments illustrate the benefits of the proposed method against the single frequent item candidate set.

Key words: Frequent dependency, containing class, multiple candidate sets

INTRODUCTION

Online shopping has become the primary activity in web. Over 85% of the world's online population have used the internet to make a purchase and more than half of internet users are regular online shoppers who make online purchases at least once a month (Nielsen Company, 2008). Not like shopping in supermarkets, consumers can easily find what they want in the capacious room for display of goods. In Internet mall, goods can be only presented in web pages, which can not present more goods for consumers at the same time. Though searching functions are provided in most of web sites, it is not still convenient to find what consumers really need. Efficient commodity organization and recommendation can increase sales (Wolf, 2007; Lee and Kwon, 2008). It is necessary for webmasters to let consumers see the commodities that they really want easily, which means that the webmasters must present the salable goods for most of consumers on the prominent position in their websites. Obviously, what are the salable goods is the primary problem. Some webmasters often place the promotional or discount products in the front page, but maybe they are not what the consumers really want. In this study, a strategy based on frequent pattern mining is proposed to find the most welcome product combination for most of consumers, which is applied on the consumers' browsing records.

Frequent pattern mining is a classical problem in data mining, many researchers give out their solutions for different data environments. Agrawal and Srikant (1995) firstly introduced frequent sequential patterns in large databases of customer transactions and in Srikant and Agrawal (1996), they generalized the problem with time constraints and relaxed the restriction on the consistency of the elements' sources. The advised methods use the strategy of candidate set generation-and-test. Han *et al.* (2004) proposed a novel FP-growth mining method, which adapts a FP-tree structure and can compress a large database into a smaller FP-tree structure to reduce the database scans and the number of candidate sets. Pei *et al.* (2004) and Han *et al.* (2000), are developed FreeSpan and PrefixSpan, both of them are projection-based, sequential pattern-growth approach, which can reduce the generation of candidate subsequence and are fit for more larger sequence databases.

The online shopping environment has its own characteristics, which is a little different from classical sequential database in the strict sense. We can consider its data as two-valued. Every sequence is possible to be long because of varieties of commodities and frequent browsing patterns are often short because of the fact that a consumer is impossible to be interested in large numbers of goods at one time. It is important to examine the frequent pattern problem in this situation. In this study, we propose a method based on frequent dependent relationship among commodities, the dependent relationship is computed according to a frequent similar function and a given support threshold. Through the dependent relationship, we can find all containing classes which can help us to narrow the search space.

Corresponding Author: Qing Yang, Computer and Control Institute, Guilin University of Electronic Technology, No. 1, Jinji Road, Qixing District, Guilin, 541004, China Tel/Fax: 86-535-5601330

PROBLEM DEFINITION

The problem of frequent browsing pattern mining in online shopping environment is detailed and some formal definitions are made for the problem. A concrete example is accompanied to understand these definitions.

A two-dimension matrix is used to denote users' browsing data, whose rows represent users and columns represent commodities, which will be also called as items. Every row can be called user's browsing vector, every column can be called commodity's browsed vector. The whole matrix is called as basic fact table (abbr. FT). Every field in the table is two-valued, 0 or 1. If the ith user browsed the jth commodity, FT[i][j]=1, otherwise 0.

A commodity pattern is a combination of some commodities, which is a subset of the whole commodity set. A minimum support value is given, remembered as min_support, which is a threshold for deciding whether a commodity combination pattern is frequent. A pattern is frequent iff all commodities in the pattern appear in at least min_support users' browsing vectors synchronously.

In order to compare the frequent browsed similarity among commodities. A frequent similar function is introduced, remembered as simFrequent. This function returns an integer.

$$\text{simFrequent}(CV_{ci},CV_{cj}) = \sum_{k=1}^{n} CV_{ci}[k] \times CV_{cj}[k]$$

Given two different commodities c_i an c_j, whose corresponding browsed column vectors are Cv_{ci} and Cv_{cj} and both of them are n-dimensional. c_i and c_j will appear in some frequent subpatterns iff simFrequent (Cv_{ci}, Cv_{cj})≥min_support. simFrequent can also take more than two parameters or only one parameter. For example:

$$\text{simFrequent}(CV_{c_1},CV_{c_2},\cdots,CV_{c_m}) = \sum_{k=1}^{n}\prod_{i=1}^{m} CV_{c_i}[k]$$

Obviously, simFrequent is monotone. Given two sets of items, S_1 and S_2, $S_1 \subseteq S_2$. If simFrequent (S_2)≥min_support, simFrequent (S_1)≥min_support, is also right. If simFrequent (S_1)≥min_support, simFrequent (S_1)≥min_support, too. We will utilize the characteristic to optimize our computing later. We make several related definitions as follows,

Definition 1: Frequent dependent relationship given a commodity X, for every element Y in the commodity set, if simFrequent(X,Y)≥min_support, we say frequent dependent relationship exists between X and Y, remembered as X→Y.

Table 1: Basic fact table of browsing log

	Commodities									
	Category 1			Category 2				Category 3		
UID	a1	a2	a3	b1	b2	b3	b4	c1	c2	c3
U1	0	1	1	0	1	0	0	0	1	0
U2	1	1	0	1	1	0	0	1	1	0
U3	1	1	1	0	0	0	0	1	0	0
U4	1	0	0	0	0	0	0	1	1	1
U5	1	0	1	0	0	1	1	0	1	0

Definition 2: If X→Y and ∀i, X[i]≥Y[i], we say that X decides Y, remembered as x⇒Y.

Obviously, X→Y, iff Y→X. → is symmetrical, but not transitive and ⇒ is transitive, not symmetrical. If there are X→Y and Y→Z, we can not get X→Z. If there are X→Y, Y→Z and X→Z, we still cannot conclude X→Y∪Z. But to operator ⇒, the results are reverse. Here, $Y \cup Z = (Y[1]*Z[1],\cdots,Y[n]*Z[n])$.

Definition 3: Containing class given a commodity X, for every element Y in the commodity set, if X→Y, then Y is an element of X's containing class, remembered as Contain(X).

If X⇒Y, Y is an element of X's containing class, but X does not belong to Contain(Y) and it is trivial to compute Contain(Y) at this situation, because any element that has dependent relationship with Y must have dependent relationship with X. All commodities that possibly appear in the same frequent pattern with X are in X's containing class.

Example 1: Let, Table 1 be the basic fact table. Suppose min_support = 2, a1→a2, a1→a3, a1→c1, a1→c2 are right in the table, then we have Contain(a1) = {a2,a3,c1,c2}. It means that the biggest frequent pattern containing a1 is possible to be {a1,a2,a3,c1,c2}. Equally, Contain(a2) {a1,a3,b2,c1,c2}, a2 = Contain(b2) because of a2⇒b2.

Definition 4: Frequent browsing pattern given a containing classes C of X, if simFrequent (X,Y)≥min_support, ∀Y, Y⊆C, we say X∪Y is a frequent browsing pattern. Considering Contain(a1), the frequent patterns are {a1,a2,c1}, {a1,a3}, {a1,c1,c2}.

BROWSING PATTERN MINING WITH FREQUENT DEPENDENCY

Here, we propose an algorithm to find all maximum frequent browsing patterns from the basic fact table considering the given support threshold.

Data cleaning and preparation: Users' browsing data often exists much noise, which are not users' real intention about shopping and should be cleaned. For example, if a consumer wants to buy something from some website, his (her) interests are often concentrated on some goods in one or several categories. If one consumer browses various goods or categories in a short time, obviously his (her) browsing is not for buying, or does not contain a clearing buying objectives, such records should be cleaned. Here we should define a threshold which can help to delete the records efficiently and it will be set empirically. Certainly, if any dependency does not hold among the items in the table, the deep mining is no meaning. When we remove some illegal data, we should find all single frequent items according to the simFrequent function. The frequent item, c_i, should satisfy:

$$\text{simFrequent}(CV_{c_i}) = \sum_{k=1}^{n} CV_{c_i}[k] \geq \text{min\_support}$$

The set including all single frequent items is marked as:

$$\text{FreItemSet} = \{c_i \mid \forall i,\ \text{simFrequent}(CV_{c_i}) \geq \text{min\_support}\}$$

Multiple candidate sets method: All frequent browsing patterns are composed of the items in FreItemSet, we can use an exhaustive method to find them, but it has a very low efficiency because of massive computing. The computing is necessary only for items that have dependent relationship. The basic strategy by our algorithm is to find the containing classes based on dependent relationship of every element in the commodity set, which can efficiently reduce the searching space and degrade the computing complexity and then use an iterative process to compute the frequent patterns in every containing class. As we have seen in the above section, transitivity does not hold for operation →. For example, if we want to know whether {a1, a2 ,a3} is a frequent pattern, we should have simFrequent(a1,a2,a3)≥min_support, not just simFrequent(a1,a2)≥min_support and simFrequent (a2,a3)≥min_support.

Firstly, we should compute the dependent relationships among the items in FreItemSet with the simFrequent function, the result is a set composed of all dependent pairs, remembered as dependSet. It is possible for some single frequent items to have not any dependent relationship with other frequent items. We should remove all such single frequent items. The rest dependent pairs are grouped by their left part, for every group, we obtain a corresponding containing class. For example, {X→Y, X⇒Z, X→R, X→S} is one group of the result, then Contain{X} = {Y,Z,R,S} and it is impossible for other items to be included by Contain(X), which means that no other items except the ones in Contain(X) can co-occurrence in the final frequent browsing patterns with X. We will compute the frequent browsing patterns according to the above containing classes. Here, we utilize the monotone of simFrequent and the similar thinking of Hasse Diagram (Sharma *et al.*, 2007) to optimize computing.

We give a Hasse diagram in Fig. 1, suppose {a, b, c, d} is the current containing class for x. When we find that {x,a,b,c} is a frequent pattern, then {x,a,b}, {x,a,c}, {x,b,c} must be under monotone and extra

Algorithm 1: Multiple candidate sets browsing pattern mining based on frequent dependency

Input: the basic fact table, FT
the single frequent item set, FreItemSet
minimum support value, min_support
Output: frequent browsing patterns set

```
dependSet = ∅
FOR any element X, Y in FreItemSet
    IF (simFrequent(X,Y)≥min_support)
        IF ∑_{i=1}^{n} Y[i] == simFrequent(X,Y)
            dependSet = dependSet∪{X⇒Y}
        ELSE
        dependSet = dependSet∪{X→Y, Y→X}
        ENDIF
    ENDIF
ENDFOR
remove all single frequent items that don't appear in any dependent relationship
group all the dependency by their left part
Contain(X) = {Y|X→Y∈dependSet∨X⇒Y∈dependSet}
FOR each containing class
    construct Hasse Diagram on the containing class
    Find the biggest element E_j and simFrequent(E_j)
    WHILE(C_i ≠ ϕ)
        IF simFrequent(E_j)≥min_support
            output E_j as a frequent pattern
            remove E_j and all its subsets
        ELSE
            remove E_j
            choose the next computing element
            according to path optimization rule
        ENDIF
    ENDWHILE
ENDFOR
```

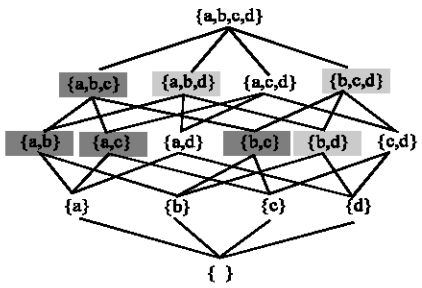

Fig. 1: Lattice for dependent relationship

computing can be avoided. If {x,b,d} is not a frequent pattern, the computation for {x,a,b,d} and {x,b,c,d} is also redundant. Here, we can do further optimization based on computation path. For example, {x,a,b,c} is not a frequent pattern, which means that at least one of its subsets is not frequent, suppose it is {x,a,b}, according to the monotone of simFrequent, the other supersets of {x,a,b}, here is {x,a,b,d}, must not be. If we find {x,a,c} is frequent, the next better choice should be {x,b,c,d}, not {x,a,c,d}. The computing complexity for {x,b,c,d} and {x,a,c,d} is same, but $\{x,a,c\}\cap\{x,b,c,d\}\subseteq \{x,a,c\}\cap\{x,a,c,d\}$, when {x,b,c,d} is frequent, we can remove more elements. The details are elaborated in Algorithm 1.

In the proposed method, we divide the only candidate set composed by all single frequent items into multiple candidate sets through the computing of dependent relationship, which can reduce the search space efficiently. In every sub-search space, we utilize the monotone of simFrequent and the Hasse diagram structure to prune the search space and find an efficient computation path.

RESULTS

Here, we evaluate multi-candidate set browsing pattern mining based on frequent dependent relationship (MCSFD) on our datasets and show how MCSFD outperforms single frequent item candidate set (SFICS) in runtime and the later regards all single frequent items as a candidate set without considering the dependent relationships. Both SFICS and MCSFD are implemented in Java. All experiments have been performed on Windows environment with a Core Duo 2.2 GHz CPU and 2 GB main memory. The datasets are simulated according to the real online shopping scenarios, which are represented by matrixes and summarized in Table 2.

The name of every dataset is formatted as AAA_BBB_CCC, in which AAA represents the number of consumers, BBB the number of commodities and CCC the minimum support threshold. For our evaluation, we compare MCSFD against SFICS according to the execution time on different datasets. For every dataset, we execute both SFICS and MCSFD for three times and get the average time for the experiment results (Table 3).

In Fig. 2, we fixed the number of commodities to 1000, let the number of consumers varies from 10000 to 100000 and then compare the runtime between SFICS and MCSFD. With the different number of consumers, the value of frequent support degree is decided empirically. The number of both frequent items and frequent

Table 2: Datasets summarization

Dataset name	No. of single frequent-item	No. of frequent dependency
1000_500_500	144	972
10000_1000_500	234	970
30000_1000_1451	290	1399
50000_500_2500	122	1691
50000_1000_2500	349	2589
50000_1500_2500	403	3186
50000_2000_2500	444	4081
100000_1000_4550	312	1608
10000_1000_3000	300	1945
30000_1000_3000	300	2867
50000_1000_3000	300	3123
100000_1000_3000	300	4428

Table 3: Experiment execution time (sec)

	SFICS				MCSFD			
Dataset	1st	2nd	3rd	Avg.	1st	2nd	3rd	Avg.
10000_500_500	16	17	16	16.3	5	5	5	5
50000_1000_2500	642	646	647	645.0	177	178	185	180

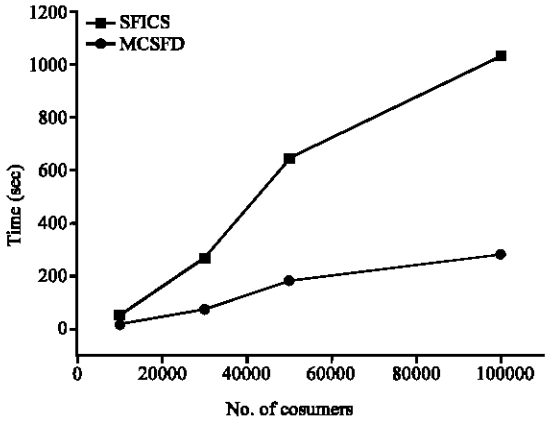

Fig. 2: Commodity no. = 1000

dependency increase when the number of consumers changes from 10000 to 50000. But both of them in 100000_1000_4550 decrease, which is because of a bigger threshold for the frequent support value. Every commodity is welcome by some certain number of people, when the number of consumers reaches a certain value, its increase contributes little to the support degree of commodities. To every dataset, MCSFD need less time than SFICS. In the second group of experiments (Fig. 3), we fixed the number of consumers on 50000, the corresponding support degree is pegged at 2500 and let the number of commodities rises from 500 to 2000. In every experiment, MCSFD has better performance than SFICS. When the number of commodities is 2000, MCSFD needs only 313 sec, while SFICS, 1139.3 sec, is 2.64 times larger than MCSFD. The experiments show that MCSFD is more effective especially for large number of commodities.

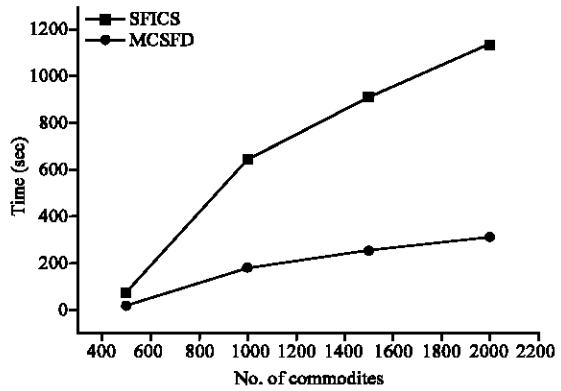

Fig. 3: Consumer No. = 50000

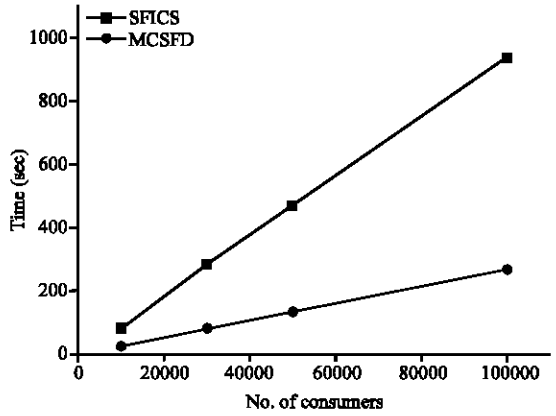

Fig. 4: CommodityNo. 1000 and support = 3000

Another group of experiments are carried on a group of datasets in which the number of commodities and single frequent items are fixed, 1000 and 3000. The runtime by both SFICS and MCSFD is linear with the number of consumers (Fig. 4). The experiments shows that our method is more fit in the environment where more single frequent items and more frequent subpatterns exist.

CONCLUSIONS

In this study, we examined the problem of effective commodity presentation on online shopping websites and have proposed an approach based on frequent dependent relationship to mine frequent browsing patterns from the users' browsing log. Through introducing the frequent dependent relationship among commodities, we can construct containing classes that are just the candidate sets for frequent pattern mining, the process narrows the searching space and reduces the computing complexity. The experimental results on our datasets show our method's efficiency for finding all frequent browsing patterns and it can be applied on more large datasets. Based on the present study, we will step further to find the optimal browsing patterns for every kind of users and provide more effective recommendation for specific types of users.

ACKNOWLEDGMENTS

This study is supported by National Natural Science Foundation of China (NSFC No. 60961002). Authors would like to thank Ping Zhou for helpful discussions and reviewing the draft of the study and her advice improved the quality of the study. Jingwei Zhang devoted himself to designing and carrying out some experiments.

REFERENCES

Agrawal, R. and R. Srikant, 1995. Mining sequential patterns. Proceedings of the 11th International Conference on Data Engineering, March 6-10, Taipei, Taiwan, pp: 3-14.

Han, J., J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal and M. Hsu, 2000. FreeSpan: Frequent pattern-projected sequential pattern mining. Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Aug. 20-23, Boston, Massachusetts, USA., pp: 355-359.

Han, J., J. Pei, Y. Yin and R. Mao, 2004. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. Data Mining Knowledge Discovery, 8: 53-87.

Lee, K.C. and S. Kwon, 2008. Online shopping recommendation mechanism and its influence on consumer decisions and behaviors: A causal map approach. Expert Syst. Appl. Int. J., 35: 1567-1574.

Nielsen Company, 2008. Trends in online shopping: A global Nielsen consumer report. pp: 1-6. http://th.nielsen.com/site/documents/GlobalOnlineShoppingReportFeb08.pdf.

Pei, J., J. Han, B. Mortazavi-Asl, J. Wang and H. Pinto *et al*., 2004. Mining sequential patterns by pattern-growth: The prefixspan approch. IEEE Trans. Knowledge Data Eng., 16: 1424-1440.

Sharma, S., A. Tiwari, S. Sharma and K.R. Pardasani, 2007. Design of algorithm for frequent pattern discovery using lattice approach. Asian J. Inform. Manage., 1: 11-18.

Srikant, R. and R. Agrawal, 1996. Mining sequential patterns: Generalizations and performance improvements. Proc. Int. Conf. Extend. Database Technol., 1057: 3-17.

Wolf, K.L., 2007. The environmental psychology of shopping: Assessing the value of trees. Res. Rev., 14: 39-43.