Mining Frequent Patterns via Pattern Decomposition

Qinghua Zou
Department of Computer Science
University of California
Los Angeles, CA 90095
USA
voice: 310-206-4561
fax: 310-206-0068
email: zou@cs.ucla.edu

Wesley Chu
Department of Computer Science
University of California
Los Angeles, CA 90095-1596
USA
voice: 310-825-2047
fax: 310-825-2273
email: wwc@cs.ucla.edu

Mining Frequent Patterns via Pattern Decomposition

Qinghua Zou and Wesley W. Chu, UCLA, USA

INTRODUCTION

Pattern decomposition is a data mining technology that uses known frequent or infrequent patterns to decompose a long itemset into many short ones. It finds frequent patterns in a dataset in a bottom-up fashion and reduces the size of the dataset in each step. The algorithm avoids the process of candidate set generation and decreases the time for counting supports due to the reduced dataset.

BACKGROUND

A fundamental problem in data mining is the process of finding frequent itemsets (FI) in a large dataset which enables essential data mining tasks such as discovering association rules, mining data correlations, and mining sequential patterns. Three main classes of algorithms have been proposed:

- Candidates generation and test (Agrawal, Srikant, 1994; Heikki, Toivonen, Verkamo, 1994; Zaki et al., 1997;): starting at k=0, it first generates candidate *k+1* item sets from known frequent *k* item sets and then counts the supports of the candidates to determine frequent *k+1* item sets which meet a minimum support requirement.
- Sampling technique (Toivonen, 1996): uses a sampling method to select a random subset of a dataset for generating candidate itemsets and then test these candidates to identify frequent patterns. In general, the accuracy of this approach is highly dependant on the characteristics of the dataset and the sampling technique which has been used.

- Data transformation: transforms an original dataset to a new one which contains a smaller search space than the original dataset. FP-tree-based (Han, Pei, and Yin, 2000) mining first builds a compressed data representation from a dataset and then mining tasks are performed on the FP-tree rather than on the dataset. It has performance improvements over Apriori (Agrawal, Srikant, 1994) since infrequent items do not appear on the FP-tree and thus the FP-tree has a smaller search space than the original dataset. However, FP-tree can not further reduce the search space by using infrequent 2-item or longer itemsets.

What distinguishes pattern decomposition (Zou et al., 2002) from most previous works is that it reduces the search space of a dataset in each step of its mining process.

MAIN THRUST OF THE CHAPTER

Both the technology and application will be discussed to help clarify the meaning of pattern decomposition.

Search Space Definition

Let $N=X:Y$ be a transaction where X, called the head of N, is the set of required items and Y, called the tail of N, is the set of optional items. The set of possible subsets of Y is called the power set of Y, denoted by $P(Y)$.

Definition 1 For $N=X:Y$, the set of all the itemsets obtained by concatenating X with the itemsets in $P(Y)$ is called the **search space** of N, denoted as $\{X:Y\}$. That is,

$$\{X:Y\} = \{X \cup V \mid V \in P(Y)\}.$$

For example, the search space $\{b:cd\}$ includes four itemsets b, bc, bd, and bcd. The search space $\{:abcde\}$ includes all subsets of $abcde$.

By definition 1, we have $\{X{:}Y\}=\{X{:}Z\}$ where $Z=Y-X$, referring to the set of items contained in Y but not in X. Thus we will assume Y does not contain any item in X when $\{X{:}Y\}$ is mentioned in this paper.

Definition 2 Let S, S_1, and S_2 be search spaces. The set $\{S_1, S_2\}$ is a *partition* of S if and only if $S= S_1 \cup S_2$ and $S_1 \cap S_2=\phi$. The relationship is denoted by $S=S_1+S_2$ or $S_1= S-S_2$ or $S_2= S-S_1$. We say S is partitioned into S_1 and S_2. Similarly, a set $\{S_1, S_2, \ldots, S_k\}$ is a partition of S if and only if $S= S_1 \cup S_2 \cup \ldots \cup S_k$ and $S_i \cap S_j=\phi$ for $i,j\in [1..k]$ and $i\neq j$. We denote it as $S=S_1+S_2+\ldots+S_k$.

Let a be an item where aX is an itemset by concatenating a with X.

Theorem 1 For $a\notin X,Y$, the search space $\{X{:}aY\}$ can be partitioned into $\{Xa{:}Y\}$ and $\{X{:}Y\}$ by item a, i.e., $\{X{:}aY\}=\{Xa{:}Y\}+\{X{:}Y\}$.

Proof: It follows from the fact that each itemset of $\{X{:}aY\}$ either contains a , i.e. $\{Xa{:}Y\}$, or does not contain a, i.e. $\{X{:}Y\}$. ■

For example, we have $\{b{:}cd\}=\{bc{:}d\}+\{b{:}d\}$.

Theorem 2 *Partition search space*: Let $a_1, a_2, \ldots, a_k$ be distinct items and $a_1a_2\ldots a_k$Y be an itemset, the search space of $\{X{:}\ a_1a_2\ldots a_k\text{Y}\}$ can be partitioned into

$$\sum_{i=1}^{k}\{Xa_i : a_{i+1}\ldots a_kY\}+\{X:Y\}, \text{where } a_i \notin X,Y.$$

Proof: It follows by partitioning the search space via items $a_1,a_2,\ldots,a_k$ sequentially as in theorem 1. ■

For example, we have $\{b{:}cd\}=\{bc{:}d\}+\{bd{:}\}+\{b{:}\}$ and $\{a{:}bcde\}= \{ab{:}cde\}+\{ac{:}de\}+\{a{:}de\}$.

Let {*X:Y*} be a search space and *Z* be a known frequent itemset. Since *Z* is frequent, all subset of *Z* will be frequent, i.e. every itemset of {*:Z*} is frequent. Theorem 3 shows how to prune the space {*X:Y*} by *Z*.

Theorem 3 *Pruning search space*: if *Z* does not contain the head *X*, the space {*X:Y*} can not be pruned by *Z*, i.e., {*X:Y*}-{*:Z*}={*X:Y*}. Otherwise, the space can be pruned as

{*X:Y*}-{*:Z*} $=\sum_{i=1}^{k}\{Xa_i : a_{i+1}...a_k(Y \cap Z)\}$, $a_1a_2...a_k$=*Y-Z*.

Proof: If *Z* does not contain *X*, no itemset in {*X:Y*} is subsumed by *Z*. Therefore, knowing that *Z* is frequent, we can not pruned any part of the search space {*X:Y*}.

Otherwise, when *X* is a subset of Z, we have {*X:Y*}$=\sum_{i=1}^{k}\{Xa_i : a_{i+1}...a_kV\}+X:V$,

where *V*=Y$\cap$Z. The head in the first part is Xa_i where a_i is a member of Y-Z. Since Z does not contain a_i, the first part can not be pruned by Z. For the second part, we have {*X:V*}-{*:Z*}={*X:V*}-{*X:(Z-X)*}. Since X$\cap$Y=ϕ, we have *V*$\subseteq$*Z-X*. Therefore {*X:V*} can be pruned away entirely. ■

For example, we have {*:bcde*}-{*:abcd*} = {*:bcde*}-{*:bcd*} = {*e:bcd*}. Here *a* is irrelevant and is removed in the first step. Another example, {*e:bcd*}-{*:abe*} = {*e:bcd*}-{*:be*} = {*e:bcd*}-{*e:b*} = {*ec:bd*}+{*ed:b*}.

Pattern Decomposition

Given a known frequent itemset Z, we are able to decompose the search space of a transaction *N=X:Y* to *N'=Z:Y'* if *X* is a subset of *Z*, where *Y'* is the set of items that appear in *Y* but not in *Z*, denoted by PD(*N=X:Y*|Z)= Z:Y'.

For example, if we know that an itemset *abc* is frequent, we can decompose a transaction *N=a:bcd* into *N'=abc:d*. That is PD(*a:bcd*|*abc*)=*abc:d*.

Given a known infrequent itemset Z, we can also decompose the search space of a transaction *N=X:Y*. For simplicity, we use three examples to show the decomposition by known infrequent itemsets and leave out its formal mathematic formula in general cases. Interested readers can refer to (Zou, Chu, Lu, 2002) for details. For example, if *N*=*d:abcef*, and a known infrequent itemsets, then we have:

- For infrequent 1-itemset ~*a*, PD(*d*:*abcef*|~a) = *d*:*bcef* by dropping *a* from its tail.
- For infrequent 2-itemset ~*ab*, PD(*d*:*abcef*|~*ab*) = d:*bcef*+*da*:*cef* by excluding *ab*.
- For infrequent 3-itemset ~*abc*, PD(*d*:*abcef*|~*abc*) = *d*:*bcef*+*da*:*cef*+*dab*:*ef* by excluding *abc*.

By decomposing a transaction t, we reduce the number of items in its tails and thus reduce its search space. For example, the search space of *a:bcd* contains the following eight itemsets {*a*, *ab*, *ac*, *ad*, *abc*, *abd*, *acd*, *abcd*}. Its decomposition result *abc:d* contains only two itemsets {*abc*, *abcd*}, which is only 25% of its original search space.

When using pattern decomposition, we find frequent patterns in a stepwise fashion starting at step 1 for 1-item itemsets. At a step *k*, it first counts the support for every possible *k*-item itemsets contained in the dataset D_k to find frequent *k*-item itemsets L_k and infrequent *k*-item itemsets $\sim L_k$. Then, using the L_k and $\sim L_k$, D_k can be decomposed into D_{k+1}, which has a smaller search space than D_k. The above steps continue until the search space D_k becomes empty.

An Application

The motivation of our work originates from the problem of finding multi-word combinations in a group of medical report documents, where sentences can be viewed as transactions and words can be viewed as items. The problem is to find all multi-word combinations that occur at least in two sentences of a document.

As a simple example, for the following text

Aspirin greatly underused in people with heart disease

DALLAS (AP) -- Too few heart patients are taking aspirin despite its widely known ability to prevent heart attacks, according to a study released Monday.

The study, published in the American Heart Association's journal Circulation, found that only 26 percent of patients who had heart disease and could have benefited from aspirin took the pain reliever.

"This suggests that there's a substantial number of patients who are at higher risk of more problems because they're not taking aspirin," said Dr. Randall Stafford, an internist at Harvard's Massachusetts General Hospital who led the study. "As we all know, this is a very inexpensive medication -- very affordable."

The regular use of aspirin has been shown to reduce the risk of blood clots that can block an artery and trigger a heart attack. Experts say aspirin can also reduce the risk of a stroke and angina, or severe chest pain.

Because regular aspirin use can cause some side effects -- such as stomach ulcers, internal bleeding and allergic reactions – doctors are too often reluctant to prescribe it for heart patients, Stafford said.

"There's a bias in medicine toward treatment and within that bias we tend to underutilize preventative services -- even if they've been clearly proven," said Marty Sullivan, a professor of cardiology at Duke University in Durham, N.C.

Stafford's findings were based on 1996 data from 10,942 doctor visits by people with heart disease. The study may underestimate aspirin use; some doctors may not have reported instances in which they recommended patients take over-the-counter medications, he said.

He called the data "a wake-up call" to doctors who focus too much on acute medical problems and ignore general prevention.

We can find frequent 1-word 2-word, 3-word, 4-word, 5-word combinations. For instance, we found 14 4-word combinations

heart aspirin use regul, aspirin they take not, aspirin patient take not, patient doct use some, aspirin patient study take, patient they take not, aspirin patient use some, aspirin doct use some, aspirin patient they not, aspirin patient they take, aspirin patient doct some, heart aspirin patient too, aspirin patient doct use, heart aspirin patient study.

Multi-word combinations are effective for document indexing and summarization. The work in (Johnson et al., 2002) shows that multi-word combinations can index documents more accurately than using single-word indexing terms. Multi-word combinations can delineate the concepts or content of a domain specific document collection more precisely than single word. For example, from the frequent 1-word table, we may infer that "*heart*," "*aspirin*," and "*patient*" are the most important concepts in the text since they occur more often than others. For the frequent 2-word table, we see a large number of 2-word combinations with "aspirin," i.e. "*aspirin patient*," "*heart aspirin*," "*aspirin use*," "*aspirin take*," etc. This infers that the document emphasizes "*aspirin*" and "*aspirin related*" topics more than any other words.

FUTURE TRENDS

There is a growing need for mining frequent sequence patterns from human genome datasets. There are 23 pairs of human chromosomes, approximately 30,000 genes, and more than one million proteins. The above discussed pattern decomposition method can be used to capture sequential patterns with some small modifications.

When the frequent patterns are long, mining frequent itemsets (FI) is infeasible because of the exponential number of frequent itemsets. Thus, algorithms mining frequent closed itemset (FCI) (Pasquier, Bastide, Taouil, and Lakhal 1999; Zaki and Hsiao, 1999; Pei, Han, and Mao, 2000) are proposed since FCI is enough to generate association rules. However, FCI could also be as exponentially large as the FI. As a result, many algorithms for mining maximal frequent itemset (MFI) are proposed such as Mafia (Burdick, Calimlim, and Gehrke, 2001) and GenMax (Gouda, and Zaki, 2001) and SmartMiner (Zou, Chu, Lu, 2002).

The main idea of pattern decomposition is also used in SmartMiner except that SmartMiner uses tail information (frequent itemsets) to decompose the search space of a dataset rather than the dataset itself. While pattern decomposition avoids candidate set generation, SmartMiner avoids superset checking, which is a time-consuming process.

CONCLUSION

We propose to use pattern decomposition to find frequent patterns in large datasets. The PD algorithm shrinks the dataset in each pass so that the search space of the dataset is reduced. Pattern decomposition avoids the costly candidate set generation procedure and using reduced datasets greatly decreases the time for support counting.

ACKNOWLEDGEMENT

This research is supported by NSF IIS ITR 0313283 and NIH PPG Grant #4442511-33780.

REFERENCES

Agrawal, R. & Srikant, R. (1994). Fast algorithms for mining association rules. In Proceedings of *the 1994 International Conference on Very Large Data Bases*, 487-499.

Burdick, D., Calimlim, M., & Gehrke, J. (2001). MAFIA: a maximal frequent itemset algorithm for transactional databases. In Proceedings of *International Conference on Data Engineering*.

Gouda, K. & Zaki, M. J. (2001). Efficiently Mining Maximal Frequent Itemsets. In Proceedings of *the IEEE International Conference on Data Mining*, San Jose, 2001.

Han, J., Pei, J., & Yin, Y. (2000). Mining Frequent Patterns without Candidate Generation, In Proceedings of *the 2000 ACM International Conference on Management of Data*, Dallas, TX.

Heikki, M., Toivonen, H., & Verkamo, A. I. (1994). Efficient algorithms for discovering association rules. *In Usama M. Fayyad and Ramasamy Uthurusamy, editors,* In Proceedings of *the AAAI Workshop on Knowledge Discovery in Databases*, 181-192, Seattle, Washington.

Johnson, D., Zou, Q., Dionisio, J.D., Liu, Z., Chu, W. W. (2002). "Modeling Medical Content for Automated Summarization" *Annals of the New York Academy of Sciences*.

Pasquier, N., Bastide, Y., Taouil, R., & Lakhal, L. (1999). Discovering frequent closed itemsets for association rules. In Proceedings of *the 7th International Conference on Database Theory*, *January*.

Pei, J., Han, J., & Mao, R. (2000). Closet: An efficient algorithm for mining frequent closed itemsets. In Proceedings of *SIGMOD International Workshop on Data Mining and Knowledge Discovery*.

Toivonen, H. (1996). Sampling Large Databases for Association Rules. In Proceedings of *the 22nd International Conference on Very Large Data Bases*, Bombay, India, September.

Zaki, M. J. & Hsiao, C. (1999). Charm: An efficient algorithm for closed association rule mining. In *Technical Report 99-10*, Computer Science, Rensselaer Polytechnic Institute.

Zaki, M. J., Parthasarathy, S., Ogihara, M., & Li, W. (1997). New Algorithms for Fast Discovery of Association Rules. In Proceedings of *the Third International Conference on Knowledge Discovery in Databases and Data Mining*, 283-286.

Zou, Q., Chu, W., Johnson, & D., Chiu, H. (2002). Using Pattern Decomposition (PD) Methods for Finding All Frequent Patterns in Large Datasets. *Journal Knowledge and Information Systems (KAIS).*

Zou, Q., Chu, W., Lu, B. (2002). SmartMiner: A Depth First Algorithm Guided by Tail Information for Mining Maximal Frequent Itemsets. In Proceedings of *the IEEE International Conference on Data Mining*, Japan, December

TERMS AND THEIR DEFINITION

Frequent Itemset (FI): An itemset whose support is greater than or equal to the minimal support.

Infrequent Pattern: An itemset that is not a frequent pattern.

Minimal Support (minSup): A user given number which specifies the minimal number of transactions in which an interested pattern should be contained.

Pattern Decomposition: A technique that uses known frequent or infrequent patterns to reduce the search space of a dataset.

Transaction: An instance which usually contains a set of items. In this paper, we extend a transaction to a composition of a head and a tail, i.e., N=X:Y, where the head represents a known frequent itemset and the tail is the set of items for extending the head for new frequent patterns.

Support of an itemset x: The number of transactions that contains x.

Search Space: The union of the search space of every transaction in a dataset.

Search Space of a transaction *N=X:Y*: The set of unknown frequent itemsets contained by *N*. Its size is decided by the number of items in the tail of *N*, i.e. *Y*.