Frequent Pattern Discovery with Memory Constraint

Kun-Ta Chuang and Ming-Syan Chen
Graduate Institute of Communication Engineering
National Taiwan University
Taipei, Taiwan, ROC
E-mail: doug@arbor.ee.ntu.edu.tw, mschen@cc.ee.ntu.edu.tw

ABSTRACT

We explore in this paper a practicably interesting mining task to retrieve frequent itemsets with memory constraint. As opposed to most previous works that concentrate on improving the mining efficiency or on reducing the memory size by best effort, we first attempt to constrain the upper memory size that can be utilized by mining frequent itemsets in this paper.

Categories and Subject Descriptors: H.2.8 [Database Applications]: Data mining

General Terms: Algorithms, Management

Keywords: Frequent patterns, association rules, memory constraint

1. INTRODUCTION

The discovery of frequent relationship among a huge database has been known to be useful in selective marketing, decision analysis, and business management. A popular area of its applications is the market basket analysis, which studies the buying behaviors of customers by searching for sets of items that are frequently purchased together.

Unfortunately, discovering frequent itemsets suffers from an inherent obstacles, namely, the unbounded memory consumption. A large memory, which may not be prevalent in most computers nowadays, is in general required when the database is large or the minimum support is small. That will result in the serious "out of memory" system crash, making users shy away from executing the frequent itemset mining.

As a result, we in this paper attempt to discover frequent patterns in the presence of the memory constraint. Specifically, the memory constraint does not mean that the memory consumption is small as claimed in previous works. We attempt to broaden the applicability of mining frequent itemsets towards the case that the available memory is limited.

The contribution of this paper can be summarized: while previous works on mining frequent patterns mostly concentrate on improving the mining efficiency or on reducing the memory size by best effort, we further investigate in this paper the important issue of mining frequent itemsets with the explicit memory constraint.

CIKM'05, October 31–November 5, 2005, Bremen, Germany.
ACM 1-59593-140-6/05/0010.

2. MINING FREQUENT ITEMSETS WITH MEMORY CONSTRAINT

Let $sup(X)$ denote the support of itemset X in the database D. Our goal in this paper is to discover frequent itemsets in the presence of the memory constraint. We resort to level-wise search algorithms to achieve such a goal. We then formally present the support distribution plot, which will provide a good perspective to analyze the problem of mining top-k frequent itemsets.

The support distribution plot: Given the support of each l-itemset, where $l \geq 1$, the support distribution plot will consist of lines, where the i^{th} line presents the range of supports of all i-itemsets, and each i-itemset can be plotted in the i^{th} line with respect to its support. Note that according to the downward closure property, the line with respect to i-itemsets will be shorter than the line with respect to j-itemsets, where $i > j$. □

Note that, without loss of generality, the memory consumption of level-wise search algorithms is solely proportional to the number of itemsets residing in the memory, including the candidate itemsets and the stored itemsets[1]. As such, Remark 1 below gives inspiration that we can limit the memory consumption in level-wise search algorithms by constraining the size of candidates tested in each database scan.

Remark 1: *Suppose that the upper bound of the memory size is specified as M. M can be equivalently transformed to the upper number of itemsets concurrently residing in the memory. Let the corresponding upper number of itemsets in memory be denoted by M_c. As such, the memory consumption will be limited below M if at most M_c candidates will be concurrently generated-and-tested in each database scan*[2].

Clearly, Remark 1 states that a level-wise search algorithm is able to limit its memory consumption by only testing a fixed number of candidate itemsets in one database scan. For example, suppose that $M_c = 300,000$, meaning that at most 300,000 candidate itemsets can be generated-and-tested in one database scan. Assuming we have 1,000 frequent 1-items, we can only select 775 1-items to generate candidate 2-itemsets since $\binom{775}{2} = 299,925$, which is

[1]We assume that discovered frequent itemsets will be stored in the menory for further use.

[2]It is reasonable to assume that M_c is much larger than the desired number of frequent itemsets, i.e., k. Therefore, without loss of generality, we simply assume M_c only indicates the upper number of candidate itemsets which will be concurrently generated-and-tested in each database scan.

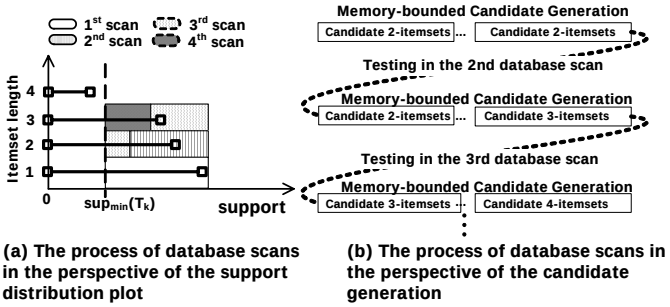

Figure 1: The illustration of mining top-k frequent itemsets under the memory constraint.

bounded below M_c. These 299, 925 candidate 2-itemsets will be tested in one database scan, and the remaining $\binom{1000}{2} - \binom{775}{2} = 199,575$ candidate 2 itemsets will be generated-and-tested in the next database scan. Note that using such an approach may incur extra database scans, which may conflict the spirit of previous works to reduce the number of database scans. But as we claimed above, the guarantee of the memory bound is in practice equally or more important than the guarantee of the small number of database scans. Fortunately, in case that the available memory is not much small, the number of database scans required by our proposed algorithm is even smaller than that required by traditional level-wise algorithms such as algorithm DHP. We will show and discuss such an advantage in the next section. It is worth mentioning that, in the implementation, the memory required to store a candidate i-itemsets and a candidate j-itemsets are indeed different, for $i \neq j$. We will also postpone the discussion of this issue to the next section. For simplicity, we assume the memory required by a candidate itemset of any size is identical in this section.

Accordingly, we then need a solution to constrain the number of generated candidates. One intuitive solution is to utilize the Apriori candidate generation technique [1] to directly generate candidate (i+1)-itemsets by combining all frequent i-itemsets until the number of candidate reaches the constrained number of candidates M_c. In fact, such a brute force approach will be costly to compute top-k frequent itemsets. We therefore resort to the recent advanced technique presented in [3] to select the appropriate set of frequent i-itemsets to generate their candidate (i+1)-itemsets. While deferring the description of the algorithm proposed, we highlight the technique in [3] to estimate the tight upper bound of candidate itemsets. Specifically, given a set of j-itemsets F_j, after the mathematical manipulation, the upper bound of candidate $(j+i)$-itemsets, where $i \geq 1$, generated from F_j can be estimated as $\widehat{C}_{j,i}(N) = \binom{m_j}{j+i} + \binom{m_{j-1}}{j-1+i} + ... + \binom{m_{s+1}}{s+i+1}$, where s is the smallest integer such that $m_s < s+i$. If no such an integer exists, $s = r-1$ [3].

Accordingly, we can devise the extension of level-wise search algorithms to discover frequent itemsets under the memory constraint as the following way:

Approach: Without loss of generality, we illustrate the idea in Figure 1, where Figure 1(a) shows the process of database scans in the perspective of the support distribution plot and Figure 1(b) shows the perspective of candidates generated in the bounded memory. Assuming the given minimum support is $\sup_{\min}$, we can initially obtain the set of 1-items whose supports exceed $\sup_{\min}$ after the first database scan. Suppose that L_i denotes the set of i-itemsets whose supports exceed $\sup_{\min}$, and $|L_i|$ denotes the number of itemsets in L_i. We then select the most n_1 frequent items of L_1, i.e., $\{X_{1,1}, X_{1,2}, ..., X_{1,n_1}\}$ to generate candidate 2-itemsets in the second database scan, where

$$n_1 = \max\left\{n \,\middle|\, \widehat{C}_{1,1}(n) \leq M_c\right\}.$$

For example, $n_1 = 775$ if M_c is specified as 300,000 ($\because \widehat{C}_{1,1}(775) = 299,925$). As such, only candidate 2-itemsets from the most frequent 775 1-items will be generated in memory and tested in the second database scan.

Afterward, if $\widehat{C}_{1,1}(|L_1|) - \widehat{C}_{1,1}(n_1) < M_c$, the remaining candidate 2-itemsets and partial candidate 3-itemsets from the most frequent n_2 2-itemsets will be generated and tested in the third database scan, where

$$n_2 = \max\left\{n \,\middle|\, \widehat{C}_{2,1}(n) < 2M_c - \widehat{C}_{1,1}(|L_1|)\right\}.$$

For example, suppose $|L_1| = 1000$. We will generate and test $\widehat{C}_{1,1}(1000) - \widehat{C}_{1,1}(775) = 199,575$ candidate 2-itemsets, and at most $2 \times 300,000 - 499,500 = 100,500$ candidate 3-itemsets in the 3^{rd} database scan, where candidate 3-itemsets are generated from most frequent 3,620 2-itemsets since $n_2 = 3620$ ($\because \widehat{C}_{2,1}(3620) = 99,995$).

Explicitly, at most M_c candidate itemsets, possibly including candidate itemsets with various lengths, will be generated and tested in one database scan until no further candidates will be generated-and-tested. Following the procedure of traditional level-wise search algorithms except the strategy of the candidate generation, we will retrieve frequent itemsets finally. In case M_c is large enough, we may directly generate candidate i-itemsets from L_{i-2} or L_{i-3} as long as the candidate count is below M_c. It can be achieved by the technique similar to the scan-reduction technique discussed in [2].

3. SUMMARY

We in this paper studied an interesting mining problem, namely mining frequent itemsets in the presence of the memory constraint. An approach was proposed to efficiently solve this problem, which will improve the feasibility of mining frequent itemsets.

Acknowledgements

The work was supported in part by the National Science Council of Taiwan, R.O.C., under Contracts NSC93-2752-E-002-006-PAE.

4. REFERENCES

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of VLDB*, 1994.

[2] M.-S. Chen, J.-S. Park, and P. S.Yu. Efficient Data Mining for Path Traversal Patterns. *IEEE Transactions on Knowledge and Data Engineering*, 1998.

[3] F. Geerts, B. Goethals, and J. V. D. Bussche. Tight upper bounds on the number of candidate patterns. *ACM Transactions on Database Systems*, 2005.