A Theoretical Comparison of Two Maximal Frequent Itemset Mining Algorithms

Haifeng Li[1, a]

[1]School of Information, Central University of Finance and Economics, Beijing 100081, China.

[a]mydlhf@139.com

Keywords: maximal frequent itemset, data mining

Abstract. Frequent pattern mining is one of the most important methods in data mining. The maximal frequent patterns are the effective condensed representation of frequent patterns; thus, they can supply a deep understanding of data for users with less storage cost. This paper introduces the concept and characteristics of maximal frequent patterns and compares two maximal frequent itemset mining algorithms in detail.

Introduction

Frequent Pattern was proposed by Agrawal in 1993[1], the motivation is to explore the frequent sets over supermarket transactions; as a result, frequent patterns are called frequent itemsets. Frequent pattern mining was widely used in many applications; thus, it becomes one of the important techniques in data mining. The aim is to find the interesting patterns, such as association rules, data interrelations and sequential patterns. Recently, networks and commercial integrations generate the mass data, dense data and real-time data, which cannot be mined by the traditional mining methods: On the one hand, when data is high relative or the minimum support is set much lower, massive frequent patterns are generated, the count is even bigger than that of the original transactions; on the other hand, frequent patterns contain redundant information, which results in the users misunderstanding.

Researchers began to find the condense representations of frequent patterns for data concision. The maximal frequent patterns, closed frequent patterns, free frequent patterns and the non-derivable frequent patterns are all condense representations, in which the closed frequent patterns and the non-derivable frequent patterns are the lossless compressions, and the free frequent patterns and the maximal frequent patterns are the approximate compressions. Nevertheless, the count of maximal frequent patterns is much smaller when the minimum support is low, which can efficiently reduce the computing cost and storage cost; furthermore, they are easier to understand for users.

In this paper, we review the representation works of maximal frequent pattern mining research and development. In Section 2, we introduce the preliminaries of frequent pattern and maximal frequent pattern. Section 3 introduces two mining algorithms according to the mining manner in detail. Section 4 concludes this paper.

Preliminaries

Given a set of distinct items Γ = {i1,i2,…,in} where $|\Gamma|$ = n denotes the size of Γ, a subset X $\subseteq \Gamma$ is called an itemset; suppose |X| = k, we call X a k-itemset. A concise expression of itemset X = {x1,x2,…,xm} is x1x2…xm. A database D = {T1,T2,…,Tv} is a collection wherein each transaction is a subset of Γ, namely an itemset. Each transaction Ti(i =1… v) is related to an id, i.e., the id of Ti is i. The absolute support (AS) of an itemset X, also called the weight of X , is the number of transactions which cover X, denoted $\Lambda(X)$= {|T ||T$\in$D$\wedge$X$\subseteq$T }; the relative support (RS) of an itemset X is the ratio of AS with respect to |D|, denoted $\Lambda r(X)= \Lambda(X)/|D|$. Given a relative minimum support λ ($0 \leqslant \lambda \leqslant 1$), itemset X is frequent if $\Lambda r(X) \geqslant \lambda$. Table 1 is a simple database.

A maximal itemset is a largest itemset in a database D, that is, it is not covered by other itemsets. A maximal frequent itemset is both maximal and frequent in D, i.e., given an relative support λ, an itemset X is maximal frequent itemset if ∧r(X)⩾λ∧!Y|Y□X.

Table 1 Simple Database

ID	Itemsets
1	a b c d e
2	a b c d
3	b c d
4	b e
5	c d e

Example 1. Given a simple database D as shown in Table 1 and an absolute support 2, the frequent itemsets are {a, b, c, d, e, ab, ac, ad, bc, bd, be, cd, ce, de, abc, abd, acd, bcd, cde, abcd}. The maximal frequent itemsets are {abcd,be,cde}.

Maximal Frequent Itemset Mining Algorithms

Maximal frequent pattern mining was proposed in 1998, Yang [2] denotes it with a bipartite graph, and prove that the count of maximal frequent patterns is #p-complete for the worst condition, further deduce that enumerating all the maximal frequent itemsets are NP-hard; thus, no algorithms can efficiently perform maximal frequent pattern mining when the minimum support is arbitrary. Nevertheless, if the proper data structure is used, as well effective traverse method and useful pruning strategy, the mining performance can be significantly improved[3]. In this paper, we review the proposed maximal frequent pattern mining algorithms, and discuss them into three categories: the static data mining, the dynamic data mining and the stream data mining.

Algorithm 1 The implementation of BFS

Require: Γ: collection of distinct items; D: collection of transactions; λ: absolute minimum support; MFI_D: collection of maximal frequent patterns;

```
1:  l=1;
2:  MFI_D={};
3:  C_l = Γ;
4:  for C_l is not empty do
5:      Gen={};
6:      for each pattern I in C_l do
7:          scan D and compute the support of I, Λ(I);
8:          if Λ(I) ≥ λ then
9:              ismax=true;
10:             for each pattern J in MFI_D do
11:                 if J ⊂ I then
12:                     MFI_D = MFI_D\J;
13:                 else
14:                     ismax=false;
15:             if ismax==true then
16:                 MFI_D = MFI_D ∪ I;
17:             Gen = Gen ∪ I;
18:     Generate the super pattern collection C_{l+1} from Gen;
19:     l+=1;
```

Algorithm 1 is a simple maximal frequent pattern mining algorithm based on the a priori. As can be seen, the basic mining algorithm is a breadth first traverse one, the support of candidate pattern is computed by scanning the dataset, and the new frequent pattern is compared to the existed maximal frequent itemsets. The drawbacks are as follows: 1) Breadth first traverse need to generate huge candidate patterns, when the transaction includes many items, the storage cost is high; nevertheless, maximal pattern mining is proposed to address mining long transactions(since long transaction may generate many redundant patterns), which is contradict to the breadth traverse method. 2) The support computing of candidate will scan dataset, which results in high I/O cost; 3) Normal breadth first

traverse is a bottom-up traverse, which requires massive super-pattern check and sub-pattern check, which results in the frequent insertion or deletion of maximal patterns, most of the computations, however, are redundant.

The development of maximal frequent pattern mining algorithms focuses on two aspects[4]: One is to improve the running time cost; another is to reduce the search space cost. Existed studies have achieved significant results.

Generally, maximal frequent pattern mining used breadth first traverse or depth first traverse. Algorithm 1 is the representation of breadth first traverse method, which may generate huge candidate itemsets when transaction is long. For an example, if a transaction has 100 items, then theoretically 2^{100} candidate itemsets need to be generated, which cannot be stored in current storage device[5].

According to the definition of maximal pattern, the mining is to find the longest patterns under a specified threshold. General frequent pattern mining algorithm is to deduce the support of super-pattern from the support of sub-patterns based on the a priori property. In such a case, if the supports of both sub-patterns and super-patterns are higher than the threshold, then the sub-patterns are redundant, since neither super-pattern check nor sub-pattern check is useful computation. The top-down traverse can reduce the redundant computation, that is, it compares the patterns from the longest pattern, which will guarantee that the generated maximal frequent patterns are the final results[6]. The advantages are as follows. On the one hand, no super-pattern check is required; on the other hand, once a maximal frequent pattern is generated, all its subsets are not the mining results, which can be ignored. The top-down traverse is also breadth first, the different is: a pattern is decided whether it is maximal first, then whether it is frequent. Given the input and output of Algorithm 1, the method topdown(T, D, λ) can be describe as an incursion process.

Let I be a pattern, which covers all the distinct items in T, then

$MFI_D = MFI_D \cup I$ if $\Lambda(I) >= \lambda$

$MFI_D = MFI_D \cup topdown(T \backslash e, D, \lambda)$ for each e in I if $\Lambda(I) < \lambda$

The weakness of top-down traverse is, although plenty of comparisons are saved, huge sub-patterns will be generated when |T| is big or the minimum support is high, which may result in massive support computation[7]; further, the sup-patterns as presented in the example, will be generated repeatedly, which results in the low efficiency. When we perform breadth first traverse in a bottom-up manner, most of the current maximal patterns will become un-maximal when longer patterns are obtained, i.e., generating these temp maximal patterns is unnecessary. The top-down traverse can address this problem, but generates huge infrequent patterns. Consequently, how to prune earlier in the bottom-up manner becomes the new consideration to reduce the search space and computing cost. Depth first traverse is the representation[8], which generates certain long patterns based on the pattern growth strategy, further reduces the candidate patterns by local pruning. Generally, since the depth first traverse is more adaptive for the characteristics of maximal patterns in structure and method, the performance is much better. We describe the implementation of this method combining with candidate group strategy in Algorithm 2.

Algorithm 2 The implementation of DFS

Require: Γ: collection of distinct items; D: collection of transactions; λ: absolute minimum support; MFI_D: collection of maximal frequent patterns;

1: **for** each item i in $t(g)$ **do**
2: generate new candidate group g', in which $h(g') = h(g) \cup i$, $t(g') = t(g) \backslash i$;
3: scan D to compute the support of $h(g')$, the $\Lambda(h(g'))$;
4: **if** $\Lambda(h(g')) \geq \lambda$ **then**
5: call DFS(Γ, D, g', λ);
6: **if** g is a leaf node and $h(g)$ not in MFI_D **then**
7: $MFI_D = MFI_D \cup h(g)$;

Summary

Frequent pattern mining is one of the most important methods in data mining. The maximal frequent patterns are the effective condensed representation of frequent patterns, which can supply a deep understanding of data for users with less storage cost. This paper introduced the concept and characteristics of maximal frequent patterns and compared two maximal frequent itemset mining algorithms in detail.

Acknowledgements

This research is supported by the National Natural Science Foundation of China (61100112, 61309030), Beijing Higher Education Young Elite Teacher Project (YETP0987). Key project of National Social Science Foundation of China(13AXW010), 121 of CUFE Talent project Young doctor Development Fund in 2014 (QBJ1427).

References

[1]. R. Agrawal, T. Imielinski and A. N. Swami. Mining Association Rules between Sets of Items in Large Databases. In Proceeding of ACM SIGMOD the International Conference on Management of Data, 1993.

[2]. G. Yang. The Complexity of Mining Maximal Frequent Itemsets and Maximal Frequent Patterns. In Proceeding of SIGKDD the International Conference on Knowledge Discovery and Data Mining, 2004.

[3]. D. Lin and Z. M. Kedem. Pincer-Search: A New Algorithm for Discovering the Maximum Frequent Set. In Proceeding of EDBT the Conference on Extending Database Technology, 1998.

[4]. R. J. Bayardo. Efficiently Mining Long Patterns from Databases. In Proceeding of ACM SIGMOD the International Conference on Management of Data, 1998.

[5]. R. C. Agarwal, C. C. Aggarwal and V. V. V. Prasad. Depth First Generation of Long Patterns. In Proceeding of SIGKDD the International Conference on Knowledge Discovery and Data Mining, 2000.

[6]. D. Burdick, M. Calimlim and J. Gehrke. MAFIA: A Maximal Frequent Itemsets Algorithm for Transactional Databases. In Proceeding of ICDE the International Conference on Data Engineering, 2001.

[7]. G. Grahne, J. Zhu. High Performance Mining of Maximal Frequent Itemsets. In Proceeding of HPDM the Workshop on High Performance Data Mining, 2003.

[8]. K. Gouda and M. J. Zaki. Efficiently Mining Maximal Frequent Itemsets. In Proceeding of ICDM the IEEE International Conference on Data Mining, 2001.