Mining Frequent Closed Itemsets from Highly Distributed Repositories

Claudio Lucchese  
Ca’ Foscari University of Venice  
clucches@dsi.unive.it

Raffaele Perego  
ISTI-CNR of Pisa  
perego@isti.cnr.it

Salvatore Orlando  
Ca’ Foscari University of Venice  
orlando@dsi.unive.it

Claudio Silvestri  
Ca’ Foscari University of Venice  
silvestri@dsi.unive.it

Abstract

In this paper we address the problem of mining frequent closed itemsets in a highly distributed setting like a Grid. The extraction of frequent (closed) itemsets is an important problem in Data Mining, and is a very expensive phase needed to extract from a transactional database a reduced set of meaningful association rules, typically used for Market Basket Analysis. We figure out an environment where a transactional dataset is horizontally partitioned and stored in different sites. We assume that, due to the huge size of datasets and privacy concerns, dataset partitions cannot be moved to a centralized site where to materialize the whole dataset and perform the mining task. Thus it becomes mandatory to perform separate mining at each site, and then merge local results for deriving global knowledge. This paper shows how frequent closed itemsets, mined independently in each site, can be merged in order to derive globally frequent closed itemsets. Unfortunately, such merging might produce a superset of all the frequent closed itemsets, while the associated supports could be smaller than the exact ones because some globally frequent closed itemsets might be not locally frequent in some partition. In order to avoid an expensive post-processing phase, needed to compute exact global results, we employ a method to approximate the supports of closed itemsets. This approximation is only needed for those globally (closed) frequent itemsets which are locally infrequent on some dataset partitions, and thus are not returned at all from the corresponding sites.

1 Introduction.

Association Rule Mining (ARM) is one of the most popular Data Mining topic. In this paper we are interested in the most computationally expensive phase of ARM, i.e the Frequent Itemset Mining (FIM) one, during which the set of all the frequent itemsets are extracted from a transactional database. Unfortunately, FIM algorithms may fail to extract all the frequent itemsets from dense databases, which contain strongly correlated items and long frequent patterns. The complexity of the mining task becomes rapidly intractable. Moreover, the huge size of the output makes the task of an analyst very hard, since she/he has to extract useful knowledge from a very large amount of frequent patterns and association rules. Closed itemsets [10, 11, 1, 9, 16, 15, 4] are a solution to the problems described above, since they are a lossless and condensed representation of all the frequent itemsets. Therefore, mining the closed frequent itemsets guarantees at the same time better performance, non redundant results, and concise representation of results. Closed frequent itemsets are the unique maximal elements of the equivalence classes that can be defined over the lattice of all the frequent itemsets. Each class includes a distinct group of frequent itemsets, all supported by the same set of transactions. When a dataset is dense, frequent closed itemsets extracted can be orders of magnitude fewer than corresponding frequent itemsets, since they implicitly benefit from data correlations. Nevertheless, they concisely represent exactly the same knowledge. From closed itemsets it is in fact trivial to generate all the frequent itemsets along with their supports. More importantly, association rules extracted from closed itemsets have been proved to be more meaningful for analysts, because all redundancies are discarded.

In this paper we address the problem of mining frequent closed itemsets in a highly distributed setting [7, 3], such as a Data Grid. While many papers address the problem of parallel/distributed FIM, at our best knowledge, no proposal for distributed closed itemset mining exists. We figure out a distributed framework in which one (virtually single) transactional dataset is horizontally partitioned among different sites. We assume that due to the huge size of datasets and privacy
concerns dataset partitions cannot be moved to a centralized site where to materialize the whole dataset and perform the mining task. Thus it becomes mandatory to apply a loosely coupled meta-miner approach, according to which we perform separate mining on each site, and then merge the local results do derive a global knowledge.

All the issues that may occur when we separately extract all the frequent itemsets from non-overlapping horizontal partitions of a dataset have been raised by a famous algorithm, Partition [12]. This algorithm can be straightforwardly implemented in a loosely-coupled distributed setting, i.e. with a limited number of communications/synchronizations (see for example [6]). We already proposed an approximate algorithm called AP [13] inspired by the Partition for solving the FIM problem. To avoid the second counting phase of Partition, which may be particularly expensive in a distributed setting, AP approximates the unknown supports of frequent itemsets by interpolation. In this paper we discuss AP_{Closed} (Approximate Partition for Closed Itemsets) which is based on AP. Unfortunately, focusing on frequent closed itemsets rather than frequent itemsets, makes the final phase, during which the locally computed results must be merged, particularly challenging. In order to address the additional issues, AP_{Closed} exploits the theoretical results described in [5].

The rest of the paper is organized as follows. Section 2 discusses the FIM problem and the concept of closed itemsets. Section 3 discusses the issues for realizing a loosely-coupled distributed algorithm, inspired by Partition, for extracting all the frequent closed itemset from an horizontally partitioned transactional dataset. Finally, Section 4 introduces our AP_{Closed} algorithm, in particular the specific solution related to the interpolation/approximation of unknown itemset supports, and the particular merging phase of the local results collected from the various distributed sites.

2 Frequent itemsets and closed ones

We introduce the concept of closed itemsets by first discussing the FIM problem, i.e. the problem of mining all the frequent itemsets from a transactional dataset \( \mathcal{D} \). Let \( \mathcal{I} = \{i_1, ..., i_m\} \) be a finite set of items, and \( \mathcal{D} \) a dataset containing \( N = |\mathcal{D}| \) transactions, where each transaction \( t \in \mathcal{D} \) is simply a list of distinct items \( t = \{i_1, ..., i_k\} \), \( i_j \in \mathcal{I} \). Let \( I \) be a \( k \)-itemset, where \( I = \{i_1, ..., i_k\} \) is a set of \( k \) distinct items \( i_j \in \mathcal{I} \). Given a \( k \)-itemset \( I \), let \( \sigma(I) \) be its support, defined as the number of transactions in \( \mathcal{D} \) that include \( I \). Mining all the frequent itemsets from \( \mathcal{D} \) with respect to a threshold \( \text{minsup} \), \( 0 \leq \text{minsup} \leq 1 \), requires to discover all the itemsets having a support higher than (or equal to) a given absolute value \( \sigma \), where \( \sigma = \text{minsup} \cdot |\mathcal{D}| \).

To introduce the concept of closed itemset, let \( T \) and \( I, T \subseteq \mathcal{D} \) and \( I \subseteq \mathcal{I} \), be subsets of all the transactions and items appearing in \( \mathcal{D} \), respectively. SE can finally define the two following functions \( f \) and \( g \):

\[
\begin{align*}
  f(T) & = \{i \in \mathcal{I} \mid \forall t \in T, i \in t\} \\
  g(I) & = \{t \in \mathcal{D} \mid \forall i \in I, i \in t\}.
\end{align*}
\]

Function \( f \) returns the set of items included in all the transactions belonging to \( T \), while function \( g \) returns the set of transactions supporting a given itemset \( I \).

**Definition 1.** An itemset \( I \) is said to be closed if and only if

\[
c(I) = f(g(I)) = f \circ g(I) = I
\]

where the composite function \( c = f \circ g \) is called Galois operator or closure operator.

The closure operator defines a set of equivalence classes over the lattice of frequent itemsets: two itemsets belong to the same equivalence class iff they have the same closure, i.e. they are supported by the same set of transactions. We can also show that an itemset \( I \) is closed iff no superset of \( I \) with the same support exist. Therefore mining the maximal elements of all the equivalence classes corresponds to mine all the closed itemsets.

Fig. 1(b) shows the lattice of frequent itemsets derived from the simple dataset reported in Fig. 1(a), mined with \( \sigma = 1 \). We can see that all the itemsets with the same closure are grouped in the same equivalence class. Each equivalence class contains elements sharing the same supporting transactions, and closed itemsets are their maximal elements. Note that closed itemsets (six) are remarkably less than frequent itemsets (sixteen). For example, the closed itemset \( A, B, C, D \), \( \sigma(A, B, C, D) = 1 \), is returned, while there are further 5 frequent itemsets that are supported by the same transaction set (only transaction 1, in this simple case).

3 The Distributed Partition algorithm

The basic idea exploited by Partition is the following: given \( m \) disjoint horizontal partitions \( \mathcal{D}_1, ..., \mathcal{D}_m \) of \( \mathcal{D} \), each one mined using the same relative support threshold \( \text{minsup} \), each globally frequent pattern \( x \) such that \( \sigma(x) \geq \text{minsup} \cdot |\mathcal{D}| \) must be locally frequent in at least one partition, e.g., \( \sigma_i(x) \geq \text{minsup} \cdot |\mathcal{D}_i| \). This guarantees that the union of all local solutions is a superset of the global solution. However, one further pass over the database is necessary to remove false positives, i.e. patterns that result locally frequent but globally infrequent.
LUCCHESE et al. : FAST AND MEMORY EFFICIENT MINING OF FREQUENT CLOSED ITEMSETS 3

Therefore, given a generator \( X \), we can extend to the case of multiple disjoint horizontal partitions \( D_1, D_2, \ldots, D_m \). The algorithm only requires two communications from the master to the slaves when the first phase of the algorithm produces false positives. Consider that, when low absolute minimum supports are used, it is likely to produce a lot of the global result \( \mathcal{C} \) is defined as follows:

\[
\mathcal{C} = \mathcal{C}_1 \oplus \mathcal{C}_2 = \mathcal{C}_1 \cup \mathcal{C}_2 \cup \{ X_1 \cap X_2 \mid (X_1, X_2) \in (\mathcal{C}_1 \times \mathcal{C}_2) \} \equiv \mathcal{C}.
\]

Unfortunately, this merging function works properly only if all locally frequent closed itemsets extracted, i.e. \( \mathcal{C}_1, \ldots, \mathcal{C}_m \), are also globally frequent. In order to ensure this, a possible solution is to adopt an expensive method like the one suggested by the Distributed Partition algorithm, which requires a second global scan of the dataset partitions to check whether locally frequent itemsets (or closed ones) are also globally frequent.

While the Distributed Partition algorithm is able to gives the exact values for supports, it has pros and cons with respect to other distributed algorithms. The pros are related to the number of communications/synchronizations: other methods as count-distribution [2, 14] require several communications/synchronizations, while the Distributed Partition algorithm only requires two communications from the slaves to the master, and a single one from the master to the slaves. The cons are the size of messages exchanged, and the possible additional computation performed by the slaves when the first phase of the algorithm produces false positives. Consider that, when low absolute minimum supports are used, it is likely to produce a lot

\[\text{TID} \quad \text{items} \]
\[
\begin{array}{cccc}
1 & B & D & \\
2 & A & B & C & D \\
3 & A & C & D & \\
4 & C & D & \\
\end{array}
\]

\[\begin{array}{cccc}
\text{Frequent Closed Itemset ABD} \\
\text{Frequent Itemset} \\
\text{Support} \\
\text{Class}
\end{array}\]

\[\begin{array}{cccc}
1 & 2 & 3 & 4 \\
ACD & ABC & ABD & BCD \\
AC & AD & AB & BD \\
AB & BC & CD & A \\
\end{array}\]

\[\begin{array}{cccc}
\emptyset & D & \mathcal{C}_2 & \mathcal{C}_1 \oplus \mathcal{C}_2 \\
\mathcal{C}_1 & ABD & ACD & BCD \\
\mathcal{C} & \mathcal{C}_1 \oplus \mathcal{C}_2 & \mathcal{C}_1 \oplus \mathcal{C}_2 & \mathcal{C}_1 \oplus \mathcal{C}_2 \\
\end{array}\]

\[\begin{array}{cccc}
\text{Equivalence} \\
\text{ABCD} \\
\text{A} \\
\text{B} \\
\text{C} \\
\text{D} \\
\end{array}\]

\[\begin{array}{cccc}
\text{Support} \\
\text{Class}
\end{array}\]

\[\begin{array}{cccc}
\mathcal{T}ID \quad \text{items} \\
1 & B & D \\
2 & A & B & C & D \\
3 & A & C & D & \\
4 & C & D & \\
\end{array}\]

\[\begin{array}{cccc}
\mathcal{C} \cup \mathcal{C}_2 & \{ X_1 \cap X_2 \mid (X_1, X_2) \in (\mathcal{C}_1 \times \mathcal{C}_2) \} \equiv \mathcal{C}.
\end{array}\]

\[\begin{array}{cccc}
\text{ABCD} \\
\text{A} \\
\text{B} \\
\text{C} \\
\text{D} \\
\end{array}\]
of false positives due to data skew present in the various dataset partitions [8]. This has a large impact on the cost of the second phase of the algorithm too: most of the slaves will participate in counting the local supports of these false positives, thus wasting a lot of time. Furthermore, in the frequent closed itemsets case, there will be also a number of unnecessary intersections.

One naïve work-around, that we will name Distributed One-pass Partition, consists in stopping Distributed Partition after the first-pass. So in Distributed One-pass Partition each slave independently computes locally frequent patterns and sends them to the master which sums-reduces the support for each pattern and writes in the result set only patterns having the sum of the known supports greater than (or equal to) minsup. In the frequent closed case, the sum reduction is substituted by the merge operator $\oplus$, which also generates the closed patterns subsumed by other patterns in local results and no longer subsumed in global results.

Distributed One-pass Partition has obvious performance advantages vs Distributed Partition. On the other hand it yields a result which is approximate with respect to both the content of the result set and the supports of discovered patterns. A globally frequent pattern $x$ may result infrequent on a given partition $D_i$ only. In other words, since $\sigma_i(x) < \minsup \cdot |D_i|$, $x$ will not be returned as a frequent pattern by the $i^{th}$ slave. As a consequence, the master of Distributed One-pass Partition can not count on the knowledge of $\sigma_i(x)$, and thus can not exactly compute the global support of $x$. Unfortunately, in Distributed One-pass Partition the master might also deduce that $x$ is not globally frequent, because $\sum_{j \neq i} \sigma_j(x) < \minsup \cdot |D|$. In case we are dealing with closed itemsets the global support is computed in a different way [5], but the issue about inexact count of globally frequent patterns remains the same.

4 The AP$_{Closed}$ algorithm

AP$_{Closed}$, the distributed algorithm we propose in this paper, tries to overcome some of the problems encountered by Distributed One-pass Partition for frequent closed itemset discovery using the same interpolation method proposed by AP [13] in the case of frequent itemsets.

When the algorithm need the support $\sigma_i(x)$ of the pattern $x$ in the $i^{th}$ partition where it is currently unknown, AP$_{Closed}$ infer an approximate value $\sigma_i(x)_{interp}$ by exploiting an interpolation method. The master bases its interpolation reasoning on the knowledge of:

- the exact support of each single item on all the partitions, and
- the support of its subpattern on all the partitions where they resulted actually frequent.

Besides the issues coming from the need for approximating the unknown local supports of some closed itemsets, our algorithm faces the problem of efficiently implementing operator $\oplus$, used to merge local results (see Theorem 3.1). In particular, we need a fast method to understand whether a given closed itemset $x$ is a subset of other ones that are already present in the global collection, or whether there exists any closed itemset whose set-intersection with $x$ is not empty. To this end, we exploit a sort of inverted file, a very popular Information Retrieval index. In particular, we assign a unique identifier (CID) to every stored closed itemset, and then store for each frequent single item the list of CIDs associated with closed itemsets that include it. Using this index it is possible to perform not only fast intersections of closed itemsets, but also to derive the various local supports of new itemsets generated by the merging function.

In the final paper we shall give all the details regarding the approximate function actually used to interpolate unknown local supports of closed itemsets, and discuss in depth our implementation of the merging function based on operator $\oplus$, also presenting preliminary results.

References


