Efficient Incremental Mining of Top-K Frequent Closed Itemsets*

Andrea Pietracaprina and Fabio Vandin

Dipartimento di Ingegneria dell'Informazione, Università di Padova, Via Gradenigo 6/B, 35131, Padova, Italy. E-mails: {capri,vandinfa}@dei.unipd.it

Abstract. In this work we study the mining of top-K frequent closed itemsets, a recently proposed variant of the classical problem of mining frequent closed itemsets where the support threshold is chosen as the maximum value sufficient to guarantee that the itemsets returned in output be at least K. We discuss the effectiveness of parameter K in controlling the output size and develop an efficient algorithm for mining top-K frequent closed itemsets in order of decreasing support, which exhibits consistently better performance than the best previously known one, attaining substantial improvements in some cases. A distinctive feature of our algorithm is that it allows the user to dynamically raise the value K with no need to restart the computation from scratch.

1 Introduction

The discovery of frequent (closed) itemsets is a fundamental primitive in data mining. Let $\mathcal{I}$ be a set of *items*, and $\mathcal{D}$ a (multi)set of *transactions*, where each transaction $t \in \mathcal{D}$ is a subset of $\mathcal{I}$. For an *itemset* $X \subseteq \mathcal{I}$ we define its *conditional dataset* $\mathcal{D}_X \subseteq \mathcal{D}$ as the (multi)set of transactions $t \in \mathcal{D}$ that contain X, and define the *support of* X *w.r.t.* $\mathcal{D}$, $\text{supp}_{\mathcal{D}}(X)$ for short, as the number of transactions in $\mathcal{D}_X$. An itemset X is *closed w.r.t.* $\mathcal{D}$ if there exists no itemset Y, with $X \subset Y \subseteq \mathcal{I}$, such that $\text{supp}_{\mathcal{D}}(Y) = \text{supp}_{\mathcal{D}}(X)$. The standard formulation of the problem requires to discover, for a given support threshold σ, the set $\mathcal{F}(\mathcal{D}, \sigma)$ of all itemsets with support at least σ, which are called *frequent itemsets* [1]. In order to avoid the redundancy inherent in $\mathcal{F}(\mathcal{D}, \sigma)$, it was proposed in [5] to restrict the discovery to the subset $\mathcal{FC}(\mathcal{D}, \sigma) \subseteq \mathcal{F}(\mathcal{D}, \sigma)$ of all closed itemsets with support at least σ, called *frequent closed itemsets*.

A challenging aspect regarding the above formulation of the problem is related to the difficulty of predicting the actual number of frequent (closed) itemsets for a given dataset $\mathcal{D}$ and support threshold σ. Indeed, in some cases setting σ too small could yield a number of frequent itemsets impractically large, possibly exponential in the dataset size [11], while setting σ too big could yield very few or no frequent itemsets.

* This work was supported in part by MIUR of Italy under project MAINSTREAM, and by the EU under the EU/IST Project 15964 *AEOLUS*.

In [10], an elegant variant of the problem has been proposed which, for a given "desired" output size $K \geq 1$, requires to discover the set $\mathcal{FC}_K(\mathcal{D})$ of *top-K frequent closed itemsets* (*top-K f.c.i.*, for short), defined as the set $\mathcal{FC}(\mathcal{D}, \sigma_K)$ where σ_K is the maximum support threshold that such that $|\mathcal{FC}(\mathcal{D}, \sigma_K)| \geq K$. Although one is not guaranteed that $|\mathcal{FC}_K(\mathcal{D})| = K$, it is conceivable that parameter K be more effective than an independently fixed support threshold σ in controlling the output size. In the same paper, the authors present an efficient algorithm, called TFP, for mining the top-K f.c.i. TFP discovers frequent itemsets starting with a low support threshold which is progressively increased, as the execution proceeds, by means of several heuristics, until the final value σ_K is reached. TFP allows also the user to specify a minimum length $\min_\ell$ for the closed itemsets to be returned. The main drawbacks of TFP are that no bound is given on the number of non-closed or infrequent itemsets that the algorithm must process, and that an involved itemset closure checking scheme is required. Moreover, TFP does not appear to be able to handle efficiently a dynamic scenario where the user is allowed to raise the value K. A number of results concerning somewhat related problems can be found in [3, 4, 8].

The mining of top-K f.c.i. is the focus of this paper. In Section 2, we study the effectiveness of parameter K in controlling the output size by proving tight bounds on $|\mathcal{FC}_K(\mathcal{D})|$. In Section 3 we present a new algorithm, TopKMiner, for mining top-K f.c.i. in order of decreasing support. Unlike algorithm TFP, TopKMiner features a provable bound on the number of itemsets touched during the mining process, and, moreover, it allows the user to dynamically raise the value K efficiently without restarting the computation from scratch. Section 4 reports some results from extensive experiments which show that TopKMiner consistently exhibits better performance than TFP, with substantial improvements in some cases (more than two orders of magnitude). The efficiency of TopKMiner becomes considerably higher when used in the dynamic scenario[1].

2 Effectiveness of K in controlling the output size

Let $\Delta(n)$ be the family of all datasets $\mathcal{D}$ whose transactions comprise n distinct items. Define $\rho(n, K) = \max_{\mathcal{D} \in \Delta(n)}(|\mathcal{FC}_K(\mathcal{D})|/K)$, which provides a worst-case estimation of the deviation of the output size from K when mining top-K closed frequent itemsets. Building on a result by [2] we can prove the following theorem.

Theorem 1. *For every $n \geq 1$ and $K \geq 1$, we have $\rho(n, K) \leq n$. Moreover, for every $n \geq 1$ and every constant c, there are $\Omega(n^c)$ distinct values of K such that $\rho(n, K) \in \Omega(n)$.*

We note that in several tests we performed on real and synthetic datasets with values of K between 100 and 10000, the ratio $|\mathcal{FC}_K(\mathcal{D})|/K$ turned out to be much smaller than n and actually very close to 1. In fact, it can be shown that $\rho(n, K) = n$ is attained only for $K = 1$, and we conjecture that $\rho(n, K)$ is a decreasing function of K.

[1] For lack of space many details have been omitted from the paper and can be found in the companion technical report [6].

3 TopKMiner

In this section, we briefly describe our algorithm TopKMiner for mining the top-K f.c.i. for a dataset $\mathcal{D}$ defined over the set of items $\mathcal{I} = \{a_1, a_2, \ldots, a_n\}$ (the indexing of the items is fixed but arbitrary). The algorithm crucially relies on the notion of ppc-extension introduced in [9] and recalled below. For an itemset $X \subseteq \mathcal{I}$, we define its *j-th prefix* as $X(j) = X \cap \{a_i \ : \ 1 \leq i \leq j\}$, with $1 \leq j \leq n$, and define $\mathrm{Clo}_{\mathcal{D}}(X) = \bigcap_{t \in \mathcal{D}_X} t$, which is the smallest closed itemset that contains X. The *core index* of a closed itemset X, denoted by $\mathrm{core}(X)$, is defined as the minimum j such that $\mathcal{D}_X = \mathcal{D}_{X(j)}$. A closed itemset X is a called a *prefix-preserving closure extension (ppc-extension)* of a closed itemset Y if: (1) $X = \mathrm{Clo}_{\mathcal{D}}(Y \cup \{a_j\})$, for some $a_j \notin Y$ with $j > \mathrm{core}(Y)$; and (2) $X(j-1) = Y(j-1)$. Clearly, if X is a ppc-extension of Y, then $\mathrm{supp}(X) < \mathrm{supp}(Y)$. Let $\perp = \mathrm{Clo}_{\mathcal{D}}(\emptyset)$, which is the possibly empty closed itemset consisting of the items occurring in all transactions. It is shown in [9] that any closed itemset $X \neq \perp$ is the ppc-extension of *exactly one* closed itemset Y. Hence, all closed itemsets can be conceptually organized in a tree whose root is $\perp$, and where the children of a closed itemset Y are its ppc-extensions.

TopKMiner generates the frequent closed itemsets in order of decreasing support by performing a best-first (i.e., highest-support-first) exploration of the nodes of the tree defined above. Specifically, the algorithm receives in input the dataset $\mathcal{D}$, the value K for which the top-K f.c.i. are sought, and a value $K^* \geq K$. During the course of the algorithm, the user is allowed to dynamically raise K up to K^*. The algorithm makes use of a priority queue Q (implemented as a max heap), whose entries correspond to closed itemsets. Let $\mathrm{E}(Y, s)$ denote an entry of Q relative to a closed itemset Y with support s. The value s is the key for the entry. Q is initialized with entries corresponding to the ppc-extension of $\perp$. Then, a main loop is executed where in each iteration the entry $\mathrm{E}(Y, s)$ with maximum s is extracted from Q, the corresponding itemset Y is generated and returned in output, and entries for all ppc-extensions of Y are inserted into Q.

A variable σ is used to maintain an approximation from below to the support σ_{K^*} of the K^*-th most frequent closed itemset. This variable is initialized by suitable heuristics similar to those employed in [10], and it is updated in each iteration of the main loop to reflect the support of the K^*-th most frequent closed itemset seen so far. The value σ is used to avoid inserting entries with support smaller that σ into Q. Also, after each update of σ, entries with support smaller than σ, previously inserted into Q, can be removed from the queue.

After the K-th closed itemset is generated, its support is stored in a variable σ'. The main loop ends when the last closed itemset of support σ' is generated. At this point the user may decide to raise K to a new value K_{new}. In this case, the main loop is reactivated and the termination condition will depend now on K_{new}. It can be shown that TopKMiner processes (i.e., inserts into Q) at most nK^* closed itemsets[2], while one such bound is not known for TFP [10].

[2] In fact, with a slight modification of the algorithm the bound can be lowered to nK.

Implementation. The efficient implementation of TopKMiner is a challenging task. For lack of space, we will limit ourself to briefly mention the key ingredients of our implementation. More details can be found in [6]. As in [7] the dataset $\mathcal{D}$ is represented through a Patricia trie $T_{\mathcal{D}}$ built on the set of transactions with items sorted by decreasing support. An entry $\mathrm{E}(Y, s)$ of Q, corresponding to some closed itemset Y, is represented by the quadruple $(\mathcal{D}_Y, s, i, Y(i-1))$, where i is the core index of Y. For space and time efficiency, we represent $\mathcal{D}_Y$ through the list $L_{\mathcal{D}}(Y)$ of nodes of $T_{\mathcal{D}}$ which contain the core index item a_i of Y and belong to paths associated with transactions in $\mathcal{D}_Y$. We observe that $(\mathcal{D}_Y, s, i, Y(i-1))$ contains only a prefix, $Y(i-1)$, of Y. The actual generation of Y is delayed to the time when the entry $(\mathcal{D}_Y, s, i, Y(i-1))$ is extracted from Q. At this point a clever traversal of the subtrie of $T_{\mathcal{D}}$ whose leaves are the nodes of $L_{\mathcal{D}}(Y)$, is employed to generate Y and the quadruples for all of its ppc-extensions to be inserted into Q.

4 Experimental evaluation

We experimentally compared the performance of TopKMiner and TFP [10] on all real and synthetic datasets from the FIMI repository (`http://fimi.cs.helsinki.fi`). The experiments have been conducted on a HP Proliant, using one AMD Opteron 2.2GHz processor, with 8GB main memory, 64KB L1 cache and 1MB L2 cache. Both TopKMiner and TFP have been coded in `C++` and the source code for TFP has been provided to us by its authors. Due to lack of space, we report only a few representative results relative to datasets kosarac and accidents. The results for the other datasets are consistent with those reported here. The characteristics of the datasets are reported in the following table, including the values $\sigma_K/|\mathcal{D}|$ for $K = 1000$ and $K = 10000$:

Dataset	#Items	Avg. Trans. Length	# Transactions	$\sigma_{1000}/\lvert\mathcal{D}\rvert$	$\sigma_{10000}/\lvert\mathcal{D}\rvert$
accidents	468	33.8	340,183	0.656	0.483
kosarac	41,270	8.1	990,002	0.006	0.002

Figure 1.(a) and 1.(b) show the relative running times of TopKMiner and TFP on kosarac and accidents, respectively, for values of K ranging from 1000 to 10000 with step 1000. For TopKMiner, we imposed $K = K^*$ so to assess the relative performance of the two algorithms when focused on the basic task of mining top-K f.c.i. In another experiment, we tested the effectiveness of the TopKMiner's feature which allows the user to dynamically raise the value K up to a maximum value K^*. To this purpose we simulated a scenario where K is raised from 1000 to 10000 with step 1000 and run TopKMiner with $K^* = 10000$ measuring the running time after the computation for each value K ended. We compared these running times with those attained by executing TFP from scratch for each value K and accumulating the running times of previous executions. The results are shown in Figure 1.(c) for dataset accidents.

We also analyzed the memory usage of TFP and TopKMiner on all datasets, for K between 1000 and 10000. TopKMiner requires less memory than TFP in almost all cases and, in the worst case, requires a factor 1.5 more memory.

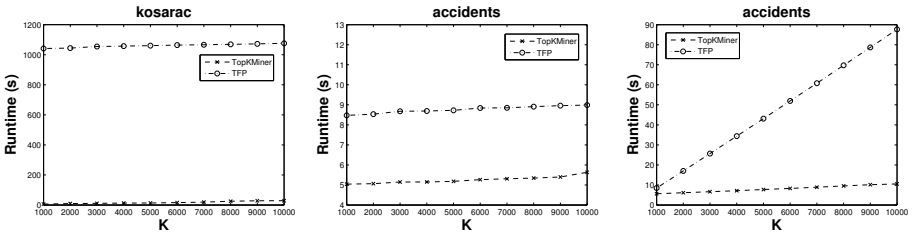

Fig. 1. Running times of TopKMiner and TFP for: (a) kosarac without dynamic update of K; (b) accidents without dynamic update of K; and (c) accidents with dynamic update of K.

References

1. R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. of the ACM SIGMOD Intl. Conference on Management of Data*, pages 207–216, 1993.
2. E. Boros, V. Gurvich, L. Khachiyan, and K. Makino. On maximal frequent and minimal infrequent sets in binary matrices. *Annals of Mathematics and Artificial Intelligence*, 39:211–221, 2003.
3. Y. Cheung and A. Fu. Mining frequent itemsets without support threshold: with and without item constraints. *IEEE Trans. on Knowledge and Data Engineering*, 16(9):1052–1069, 2004.
4. A. Fu, R. Kwong, and J. Tang. Mining n-most interesting itemsets. In *Proc. of the Intl. Symp. on Methodologies for Intelligent Systems*, pages 59–67, 2000.
5. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *Proc. of the 7th Int. Conference on Database Theory*, pages 398–416, Jan. 1999.
6. A. Pietracaprina and F. Vandin. Efficient Incremental Mining of Top-K Frequent Closed Itemsets. Technical Report available at `http://www.dei.unipd.it/~capri/PietracaprinaVTR07.pdf`
7. A. Pietracaprina and D. Zandolin. Mining frequent itemsets using Patricia tries. In *Proc. of the Workshop on Frequent Itemset Mining Implementations (FIMI03)*, Vol. 90, Melbourne, USA, Nov. 2003. CEUR-WS Workshop On-line Proceedings.
8. J. Seppanen and H. Mannila. Dense itemsets. In *Proc. of the 10th ACM SIGKDD Intl. Conference on Knowledge Discovery and Data Mining*, pages 683–688, 2004.
9. T. Uno, T. Asai, Y. Uchida, and H. Arimura. An efficient algorithm for enumerating closed patterns in transaction databases. In *Proc. of 7th Intl. Conf. Discovery Science*, pages 16–31, 2004.
10. J. Wang, J. Han, Y. Lu, and P. Tzvetkov. TFP: An efficient algorithm for mining top-k frequent closed itemsets. *IEEE Trans. on Knowledge and Data Engineering*, 17(5):652–664, 2005.
11. G. Yang. The complexity of mining maximal frequent itemsets and maximal frequent patterns. In *Proc. of the 10th ACM SIGKDD Intl. Conference on Knowledge Discovery and Data Mining*, pages 344–353, 2004.