MINI: Mining Informative Non-redundant Itemsets

Arianna Gallo[1], Tijl De Bie[1], and Nello Cristianini[1,2]

[1] University of Bristol, Department of Engineering Mathematics, UK
[2] University of Bristol, Department of Computer Science, UK

Abstract. Frequent itemset mining assists the data mining practitioner in searching for strongly associated items (and transactions) in large transaction databases. Since the number of frequent itemsets is usually extremely large and unmanageable for a human user, recent works have sought to define condensed representations of them, e.g. *closed* or *maximal* frequent itemsets. We argue that not only these methods often still fall short in sufficiently reducing of the output size, but they also output many redundant itemsets. In this paper we propose a philosophically new approach that resolves both these issues in a computationally tractable way. We present and empirically validate a statistically founded approach called MINI, to compress the set of frequent itemsets down to a list of informative and non-redundant itemsets.

1 Introduction

Frequent itemsets (or patterns) mining has been a focused research theme in data mining due to its broad applications at mining association, correlation, sequential patterns, episodes, multidimensional-patterns, max patterns, partial periodicity, emerging patterns, and many other important data mining tasks.

Since their introduction in 1993 in [1], hundreds of new scalable methods have been proposed to solve the mining frequent itemsets problems. Typically, while a too high support threshold leads to generate only commonsense patterns (or none), mining all the itemsets having a low support, or dealing with high correlated data, may generate an explosive number of results, often hard to examine for a user. In order to solve this problem, several methods were proposed to compress (or *summarize*) the set of frequent itemsets, i.e. to find a *concise representation* [2] of the whole collection of patterns. In general, a concise (or condensed) representation must enable to regenerate not only the patterns, but also the values of an evaluation function like the support without accessing the data. If these regenerated values are only approximated, the condensed representation is called *approximate*, *exact* otherwise.

Among these methods, closed [3] [4], non-derivable [5], closed non-derivable itemsets [6] (and 0-Free Sets [7]) have been suggested for finding an exact representation of the data. However, the number of closed itemsets, non-derivable and closed non-derivable itemsets can still be very large, thus an additional effort is essential to allow the user to better understanding the data.

J.N. Kok et al. (Eds.): PKDD 2007, LNAI 4702, pp. 438–445, 2007.

While most of the aforementioned state-of-the-art algorithms make use of the *restoration error* to measure the accuracy of the found patterns, here we develop a new probabilistic and *objective* [8] measure of interestingness. We describe MINI, a new scalable algorithm which discovers interesting and non redundant insights in the data. One of the novelties is that both the computation of the interestingness measure and the redundancy reduction are carried out by considering both the domain of the items and that of the transactions. Moreover, MINI does not require the user to manually choose any parameter, but a value which allows to manage the memory consumption without affecting the quality of the result. The experiments show the efficiency of the MINI algorithm which outputs effectively only the informative and non redundant itemsets that matter.

2 Problem Statement

A transactional database $\mathcal{T} = \{\mathtt{t}_i\}_{i=1}^{n}$ consists of a set of n transactions having an unique identifier. Let $\mathcal{I}$ be a set of items $\{\mathtt{i}_1, \mathtt{i}_2, ..., \mathtt{i}_m\}$. A transaction is a couple $\mathtt{t} = (tid, X)$ where tid is the transaction identifier and $X \subseteq \mathcal{I}$ is an *itemset*. A transaction $\mathtt{t} = (tid, X)$ contains an item (or an itemset) $\mathtt{i}$, if $\mathtt{i} \in X$ (or $I \in X$), and we write it $\mathtt{i} \in \mathtt{t}$ (or $I \in t$) for convenience. For any itemset I, its *tidset* T is defined as the set of identifiers of the transactions containing I. For any itemset I in the database, its *support* is defined as the number of transactions containing I: $supp(I) := |\{\mathtt{t} = (tid, X) | I \subseteq X, \mathtt{t} \in \mathcal{T}\}|$.

Similarly, we can define the support $supp(T)$ of a tidset T being the number of items shared by all the transactions $\mathtt{t} \in T$: $supp(T) := |\{i \mid \forall \mathtt{t} = (tid, X) \in T, i \in X\}|$.

An itemset is *frequent* in the database if its support in is at least a certain support threshold σ. According to this definition, any subset of a frequent itemset is frequent. This Apriori property leads to an explosive number of frequent patterns. For example, a frequent itemset with c items may generate 2^c sub-itemsets, all of which are frequent. This redundancy can be solved with the introduction of the notion of *closed* itemset, i.e. *an itemset with no frequent superset with the same support.*

Definition 1. *An itemset I is called* closed *if it has no frequent superset with the same support.*

Theorem 1. *The support of non closed itemsets is uniquely determined by the support of the closed itemsets.*

Because of theorem 1, the set of frequent closed itemsets forms a lossless representation of all frequent itemsets. Unfortunately, real applications are often subjected to noise. As long as there is a small noise on the transactions containing an itemset I, hundreds of sub-itemsets can be still generated with different supports. Since their itemsets strongly overlap with each other, these sub-itemsets are considered *redundant*. For this reason, dealing with real domains requires a more sophisticated technique to minimize this redundancy.

3 Measuring Surprise as an Interestingness Measure

Measuring the interestingness of discovered patterns is an active area of data mining research, and there is no widespread agreement on a formal definition. Most itemset summarization methods proposed so far are based on the *restoration error*, which measures the average relative error between the estimated support of a pattern and its true support. In this paper, we introduce a radically different interestingness measure, and show its practical relevance and applicability. In addition to that, we provide an efficient and scalable framework to compute this measure of interestingness of frequent itemsets.

Our interestingness measure is based on hypothesis testing ideas. In particular, we formulate a model for the database that represents the "uninteresting" situation in which no item associations are present, i.e. in which items occur independently from each other in transactions. This model is known as the *null model*. Then, for each itemset discovered, we compute its probability to be "surprising" under the null model. This probability is known as the *p-value* of the itemset. The smaller the p-value, the more surprising (informative, interesting) the itemset is, and the more interesting we consider the itemset to be.

Possible Null Models for the Data. There are two possible null models we consider appropriate. The first possible null model considers all items to be independent random variables. By this we mean that they are contained in any particular transaction with a certain probability $f_{\mathtt{i}_k} = \frac{c_{\mathtt{i}_k}}{n}$ for item $\mathtt{i}_k$ (where $c_{\mathtt{i}_k} = supp(\mathtt{i}_k)$ is the total item count in the database), independently from the presence or absence of other items. The independence assumption implies that the probability of all items $\mathtt{i} \in I$ to be jointly present in a given transaction is equal to $p_I = \prod_{\mathtt{i} \in I} f_{\mathtt{i}}$, the product of their individual probabilities. We can then compute the probability that the support of an itemset is exactly $supp(I)$ under the null model by means of the binomial distribution:

$$P(supp(I); n, p_I) = \binom{n}{supp(I)} p_I^{supp(I)} (1 - p_I)^{n - supp(I)}. \tag{1}$$

The p-value for an itemset I under this null model is then computed as the probability to observe an itemset at least as "surprising" I, which we define here as having a support at least as large as $supp(I)$. Given that all items in I occur jointly in a transaction with probability p_I, we can compute the probability of a support larger than or equal to $supp(I)$ out of n transactions by means of the cumulative binomial distribution function, as

$$P_c^I = \sum_{s=supp(I)}^{n} \binom{n}{s} p_I^s (1 - p_I)^{n-s}, \text{ with } p_I = \prod_{\mathtt{i} \in I} f_{\mathtt{i}} \tag{2}$$

This is the p-value under the first null model we consider.

A different null model is obtained by considering the transactions as independent random variables, containing each of the items with the same transaction-dependent probability $f_{\mathtt{t}_k} = \frac{c_{\mathtt{t}_k}}{m}$ for transaction $\mathtt{t}_k$ (where $c_{\mathtt{t}_k} = supp(\mathtt{t}_k)$ is the

total number of items in the transaction $\mathtt{t}_k$). Using a reasoning analogous to the above, we obtain the following p-value for a transaction set T:

$$P_c^T = \sum_{s=supp(T)}^{m} \binom{m}{s} p_T^s (1-p_T)^{m-s}, \text{ with } p_T = \prod_{\mathtt{t}\in T} f_{\mathtt{t}} \tag{3}$$

Note that, because of the Definition 1, $supp(T)$ is equal to the *cardinality* $|I|$ of the itemset I.

The first model takes effective account of the fact that associations between more frequent items are less surprising (even though they may be relevant in terms of restoration error), and hence may not be as interesting to a user (the user would *expect* to see these associations, even by chance). The second model accounts for the fact that two large transactions are likely to share items, even by mere chance, so that again such associations are less interesting. Therefore, for any itemset I supported by transactions T, we define our measure of interestingness as the largest p-value obtained by these null two models:

$$\text{p-value}(I) = \max(P_c^I, P_c^T) \tag{4}$$

In accordance with Definition 1, it is interesting to note the following fact:

Theorem 2. *For any non-closed itemset there exists a closed itemset that has a lower p-value, i.e. that is more significant.*

Because of this theorem, we can safely restrict our attention to closed itemsets only and sort them by their interestingness instead of considering all the frequent itemsets. Additionally, all the p-values are efficiently and effectively updated in a second step, described in detail in the next sections, further minimizing the redundancy.

4 The MINI Algorithm for Mining Informative Non-redundant Itemsets

The set of closed itemsets represent a concise representation of the set of all frequent itemsets. However, it could still contains redundancy. Several solution to this problem were proposed, such as relaxing the requirement that a supporting transaction contains exactly all the items in the itemset [9], or applying traditional k-means clustering algorithm [10]. In this paper we adopt a completely different approach to address the problem. The list of closed itemsets is sorted by increasing p-value, and then an iterative procedure to this list is applied to updating the p-values ignoring itemsets that are redundant with more highly listed ones. This is done by penalizing itemsets that overlap with highly ranked itemsets by increasing their p-value in a statistically principled way, so that they go down in the sorted list.

Let I_k be the itemset at position k whose p-value is going to be updated. In our framework, all the itemsets $\hat{\mathcal{I}}_{k-1} = \bigcup_k I_k, (k = 1, .., N-1)$ are said to

be *covered*, as well as their respective tidset $\hat{\mathcal{T}}_{k-1} = \bigcup_k T_k, (k = 1, .., N-1)$. Note that for each covered item $\hat{\mathtt{i}} \in \hat{\mathcal{I}}_{k-1}$, we can always individuate a set of covered tids $\{\hat{\mathtt{t}} = (tid, X) | \hat{\mathtt{t}} \in \hat{\mathcal{T}}_{k-1}, \hat{\mathtt{i}} \in X\} = \{\hat{\mathtt{t}}\}_{\hat{\mathtt{i}}}$, being the tids of the covered itemsets containing $\mathtt{i}$ in their set of items.

An itemset I_k could share some items with j covered itemsets. Its success probability p_{I_k} is updated as follows:

$$p'_{I_k} = \sum_j \frac{supp(\hat{I}_j)}{n} \prod_{\mathtt{i} \in \hat{I} \setminus I_k} f_{\mathtt{i}} + \frac{\sum_j supp(\hat{I}_j) - supp(I_k)}{n} \prod_{\mathtt{i} \in I_k \setminus \hat{\mathcal{I}}_{k-1}} f'_{\mathtt{i}} \qquad (5)$$

where for each item $\mathtt{i} \in I_k$ already covered by some more interesting itemsets, its probabilities of being contained in a transaction ($f_{\mathtt{i}}$) is updated to be $f'_{\mathtt{i}} = f_{\mathtt{i}} - \frac{|\{\hat{\mathtt{t}}\}_{\mathtt{i}}|}{n}$.

Again, using an analogous to the above, we can update the probability $p_{\mathtt{T}}$ for the other null model discussed so far.

The two p-values in Equations 2 and 3 are then computed using these two updated probabilities as new parameters, and the new interestingness measure for I_k is then chosen accordingly to Equation 4.

An Iterative Algorithm. The sketch of the MINI algorithm is given below. First of all, the set R of closed itemsets is sorted in ascending order by p-values (steps 1-3). Since there are no itemsets more interesting than it, the first itemset is assumed to be covered.

Property 1. The p-value of the itemset I_k is updated considering only the highly listed (i.e. most interesting) $k-1$ itemsets in the set.

The algorithm starts thus from the 2*nd* position (step 4) and, at each step (labeled `current_step`), it updates the p-value of the itemset I_k (accordingly to the Property 1), using the cumulative binomial probability functions in 2 and 3 with the updated probabilities discussed in the last section as parameters. This means that at each step, the iterative algorithm searches for the *k-th* most interesting itemset in the set.

If after the updating of its p-value the itemset I_k still detains the same position k in R, I_k, it is considered to be "covered" as well as its items and tids, and, in the next step, the algorithm will search for the $(k+1)$-*th* most interesting itemset (steps 13-14). Hence, at the end of the algorithm, the number of covered itemsets is at most $k \leq 1 + \mathtt{max\_steps}$, where `max_steps` is an user defined parameter defining the maximum number of steps to carried out. Usually, the user does not know apriori how many exactly interesting patterns will well summarize the data. In our framework, instead of an exact number of interesting patterns, the user needs to choose only the maximum number `c_max` of itemsets that he/she wants to try to cover, and the number $\mathtt{max\_steps} \geq \mathtt{c\_max}$ of steps to carry out.

Algorithm MINI(R, max_steps)
1: for each $I \in R$ do
2: p-value(I) := $\max\{P_c^I(supp(I); n, p_I), P_c^T(supp(T); m, p_T)\}$
3: sort R by p-values in ascending order
4: $k := 2$
5: current_step= 1
6: while (current_step<max_steps) do
7: current_step+=1
8: $I_k := I$ at position k in R
9: p-value(I_k) := $\max\{P_c^{I_k}(supp(I_k); n, p'_I), P_c^{T_k}(supp(T); m, p'_T)\}$
10: last_updated_itemset:= I_k
11: update ordering in R
12: $I_k := I$ at position k in R
13: if (I_k ==last_updated_itemset) do
14: k+=1

5 Experiments and Results

All the experiments were performed on a 1.60GHz Intel(R) Pentium(R)M with 512 MB of memory. We show the performance of our methods on 5 datasets. The first 2 consist of news articles in different periods: one month (`August 2006`), one year (`Year2006`). Another dataset, `Iraq` was constructed to contain news articles with keyword query "iraq". We will also show experiments performed on two other different kinds of dataset, containing all the titles and titles+abstracts of the papers published at the PKDD conferences of the last years (from 2000 to 2006), labeled `TitlesPkdd` and `AbstractsPkdd`. We run a closed itemsets mining algorithm to extract all the frequent closed itemsets with different support thresholds for each dataset. Details of the datasets are shown in Figure 1. In

	rows	different words	min_supp	closeds
Year2006	65754	43100	0.001	5103
August 2006	17085	23630	0.0018	5522
Iraq	5069	8593	0.001	45396
AbstractsPkdd	423	4693	0.012	51494
TitlesPkdd	423	1069	0.012	143

Fig. 1. Details of the datasets used in the experiments

Figure 2 are shown some experimental results. Figure 2(a) shows the number of covered itemsets (on the left) and the number of penalized itemsets at each step (on the right). Looking at the results, we can deduce that the ordered closed itemsets set of `August2006` contains several very disjoined (i.e. non redundant) patterns in its first positions. On the other hand, the `Year2006` initial list contains a lot of redundancy. Indeed, the slope of the functions of `August2006` is higher than that of `Year2006` in Figure 2(a) on the left, while it is lower in Figure 2(a) on the right.

We also evaluated the running time of MINI w.r.t. the parameters `c_max` (with dataset `Year2006`) and `max_steps` (with dataset `AbstractsPkdd`) (Figure 2(b)). As we already pointed out in Section 4, the number of closed itemsets is often very large, so much so that it is hard to store entirely in the memory. One solution

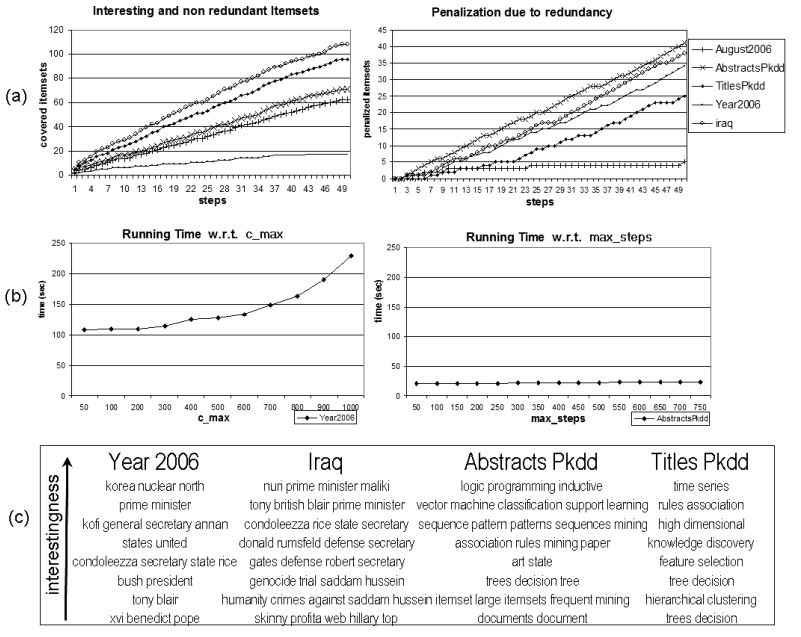

Fig. 2. Experimental Results

could be choosing a higher support threshold σ. However, not only σ is a parameter typically hard to choose a priori by the user, but it could also delete some interesting itemsets. In general, it is desirable to mine all closed itemsets without any support threshold and then perform the summarization with our interesting measure, which we claim is often more appropriate than a support threshold.

However, even if the set of closed itemsets is very large, MINE allows the user to manage the memory consumption, reducing the running time without affecting the summarization quality. In the experiments in Figure 2(b) on the left, the algorithm carries out 50 steps and covers always 17 itemsets, while the `c_max` parameter is varied. The running time grows exponentially with `c_max`, while the summarization result does not change.

In the experiments shown in Figure 2(b) (right) we varied `max_steps` on the dataset `AbstractsPkdd`. Here, the `c_max` was chosen to be equal to 1200. Note that, in this case, the summarization result varies, from 9 itemsets covered with 50 steps, to 53 in 750 steps. However, because of the MINI property 1, these results differ from each other only in their size. In each of these results, at each position the same itemset is returned, as well as its measure.

In Figure 2(c) the 8 most interesting and non redundant itemsets found in `Year2006`, `Iraq`, `AbstractsPkdd`, and `TitlesPkdd` are shown. We included the `Iraq` example to illustrate an advantage of our method over methods that focus on the restoration error. Such methods would usually be biased towards the *more frequent itemsets*, which in this case are likely to include the word iraq, perhaps even as an itemset of size 1. We argue that, in order to discover

interesting insights, the results of such techniques are often suboptimal: arguably the itemset `iraq` is not interesting considering the database we are looking at. In our framework, this pattern is automatically considered as *not* interesting and *not* surprising, as it occurs in each row of the dataset.

6 Conclusions

We propose MINI, a two-steps algorithm to mine interesting and non redundant itemsets from a database. In the first step, the cumulative binomial probability of each itemset is computed in both the items and transactions domains independently from the other itemsets in the set. In its second step, the MINI algorithm further reduces the redundancy updating their p-values, penalizing the itemsets which share some items and/or tids with more interesting itemsets. The experiments show that the MINI mines the interesting and non redundant itemsets from a dataset, *without requiring the user to define any parameter*, but `c_max` to reduce the memory consumption and `max_steps`, which can be chosen arbitrarily high without affecting the quality of the summarization. This makes MINI very scalable and thus applicable to many other interesting real world tasks.

Acknowledgment. This work was partially supported by NIH grant R33HG00 3070-01, and the EU Project SMART.

References

1. Agrawal, R., Imieliski, T., Swami, A.: Mining association rules between sets of items in large databases. In: SIGMOD, pp. 207–216. ACM Press, New York (1993)
2. Mannila, H., Toivonen, H.: Multiple uses of frequent sets and condensed representations. In: KDD, Portland, USA, pp. 189–194 (1996)
3. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Efficient mining of association rules using closed itemset lattices. 24(1), 25–46 (1999)
4. Chi, Y., Wang, H., Yu, P.S., Muntz, R.R.: Moment: Maintaining closed frequent itemsets over a stream sliding window. In: Perner, P. (ed.) ICDM 2004. LNCS (LNAI), vol. 3275, Springer, Heidelberg (2004)
5. Calders, T., Goethals, B.: Mining all non-derivable frequent itemsets, pp. 74–85. Springer, Heidelberg (2002)
6. Muhonen, J., Toivonen, H.: Closed non-derivable itemset. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS (LNAI), vol. 4213, pp. 601–608. Springer, Heidelberg (2006)
7. Boulicaut, J.F., Bykowski, A., Rigotti, C.: Free-sets: A condensed representation of boolean data for the approximation of frequency queries. Data Min. Knowl. Discov. 7(1), 5–22 (2003)
8. Geng, L., Hamilton, H.J.: Interestingness measures for data mining: A survey. ACM Comput. Surv. 38(3), 9 (2006)
9. Yang, C., Fayyad, U., Bradley, P.S.: Efficient discovery of error-tolerant frequent itemsets in high dimensions. In: SIGKDD, pp. 194–203. ACM Press, New York (2001)
10. Yan, X., Cheng, H., Han, J., Xin, D.: Summarizing itemset patterns: a profile-based approach. In: 11th ACM SIGKDD, pp. 314–323. ACM Press, New York (2005)