Frequent Closures as a Concise Representation for Binary Data Mining

Jean-François Boulicaut and Artur Bykowski

Laboratoire d'Ingénierie des Systèmes d'Information
Institut National des Sciences Appliquées de Lyon, Bâtiment 501
F-69621 Villeurbanne cedex, France
{Jean-Francois.Boulicaut,Artur.Bykowski}@lisi.insa-lyon.fr

Abstract. Frequent set discovery from binary data is an important problem in data mining. It concerns the discovery of a concise representation of large tables from which descriptive rules can be derived, e.g., the popular association rules. Our work concerns the study of two representations, namely frequent sets and frequent closures. N. Pasquier and colleagues designed the `close` algorithm that provides frequent sets via the discovery of *frequent closures*. When one mines highly correlated data, `apriori`-based algorithms clearly fail while `close` remains tractable. We discuss our implementation of `close` and the experimental evidence we got from two real-life binary data mining processes. Then, we introduce the concept of *almost-closure* (generation of every frequent set from frequent almost-closures remains possible but with a bounded error on frequency). To the best of our knowledge, this is a new concept and, here again, we provide some experimental evidence of its add-value.

1 Context and Motivations

One of the obvious hot topics of data mining research in the last five years has been frequent set discovery from binary data. It concerns the discovery of set of attributes from large binary tables such that these attributes are true within a same line often enough. It is then easy to derive rules that describe the data e.g., the popular association rules [2] though the interest of frequent sets goes further [8]. In this paper, we discuss the computation and the use of frequent sets considered as an interesting descriptive representation of binary table for typical rule mining processes.

When looking for a generic statement, it is possible to formulate a data mining task as a query over an intensionally defined collection of patterns [4]. Given a schema $\mathbf{R}$ for a database, let $(\mathcal{P}_\mathbf{R}, \mathcal{E}, \mathcal{V})$ denote the pattern domain where $\mathcal{P}_\mathbf{R}$ is a language of patterns, $\mathcal{E}$ is an evaluation function that defines pattern semantics, and $\mathcal{V}$ is a set of result values. Given $\mathbf{r}$, an instance of $\mathbf{R}$, $\mathcal{E}$ maps each $\theta \in \mathcal{P}_\mathbf{R}$ to an element of $\mathcal{V}$. Then, a mining task is the computation of the subset of $\mathcal{P}_\mathbf{R}$ that fulfil interestingness requirements. This can be formalized as the computation of $Th(\mathbf{r}, \mathcal{P}_\mathbf{R}, q) = \{\theta \in \mathcal{P}_\mathbf{R} \mid q(\mathbf{r}, \theta) \text{ is true}\}$ where predicate q indicates whether a sentence is considered interesting. Typically, this predicate is

© Springer-Verlag Berlin Heidelberg 2000

r =

	A	B	C	D	E
1	1	1	1	1	1
2	1	0	1	0	0
3	1	1	1	1	0
4	0	1	1	0	0
5	1	1	1	0	0
6	0	0	0	0	1

$\mathcal{E}$.support(C, **r**) = 5/6 = 0.83
$\mathcal{E}$.support(AC, **r**) = 4/6 = 0.67
$\mathcal{E}$.support($A \Rightarrow C$, **r**) = 0.67
$\mathcal{E}$.confidence($A \Rightarrow C$, **r**) = 4/4 = 1
$\mathcal{E}$.confidence($C \Rightarrow A$, **r**) = 4/5 = 0.8

Fig. 1. A binary dataset **r** and the behavior of some patterns

a conjunction of constraints that involves the evaluation function. This approach has been more or less explicitely used for various data mining tasks [13].

Example 1. Given a schema $\mathbf{R}=\{A_1,\ldots,A_n\}$ of attributes with domain {0,1} and a relation **r** over **R**, the support of a set X $\subseteq$ **R**, $\mathcal{E}$.support(X,**r**), denotes the fraction of rows of **r** that have a 1 in each column of X. Frequent set discovery in **r** consists in computing every subset from **R** such that its support is higher than a given threshold σ. Here, $\mathcal{P}_{\mathbf{R}}$ is $2^{\mathbf{R}}$, $\mathcal{V}$ is [0,1] and the predicate q is $\mathcal{E}$.support$(\theta, \mathbf{r}) \geq \sigma$. For instance, in Figure 1, supports of $\{C\}$ and $\{A,C\}$ in a dataset are given. Notice that we often use a string notation (e.g., AC) to denote a set of attributes. □

An explicit interestingness evaluation of all the patterns of $\mathcal{P}_{\mathbf{R}}$ in a dataset is not tractable in general. Though an exponential search space is concerned, frequent sets can be computed in real-life large datasets thanks to the support threshold on one hand and safe pruning criteria that drastically reduces the search space on the other hand (e.g., the so-called apriori trick [2]). However, there is still an active research on algorithms, not only for the frequent set discovery task when `apriori`-based algorithms fail (e.g., in the case of highly correlated data) but also for new related mining tasks, e.g., the discovery of maximal (long) frequent sets only [3].

Example 2. Association rules have been extensively studied since their introduction in [1]. Given the schema $\mathbf{R}=\{A_1,\ldots,A_n\}$, an association rule is an expression $X \Rightarrow Y$ where $X \subseteq \mathbf{R}$ and $Y \in \mathbf{R} \setminus X$. $\mathcal{P}_{\mathbf{R}}$ is the (finite) collection of such sentences. The typical "behavior" of these rules in an instance **r** over **R** is evaluated by means of two interestingness measures called "support" or "confidence". The support of a rule $X \Rightarrow Y$ is equal to the support of $X \cup Y$ (as defined in Example 1) while its confidence is equal to its support divided by the support of X. $\mathcal{V}$ is [0,1] $\times$ [0,1] and the evaluation function provides the support ($\mathcal{E}$.support) and the confidence ($\mathcal{E}$.confidence). The "classical" association rule mining task concerns the discovery of rules whose support and confidence are greater or equal to user-given thresholds, resp., σ and ϕ. The predicate q is defined as $\mathcal{E}$.support$(\theta, \mathbf{r}) \geq \sigma \wedge \mathcal{E}$.confidence$(\theta, \mathbf{r}) \geq \phi$. For example, with σ=0.5 and ϕ=0.9, $A \Rightarrow C$ is discovered in the data of Figure 1 while $C \Rightarrow A$ is not. □

In the case of association rules, left-hand and right-hand sides denote conjunctions of properties. We can consider the case of generalized rules where other boolean operators, like negation and disjunction, are allowed.

Example 3. The rule $A \wedge \neg E \Rightarrow C$ is an example of a generalized rule which might be extracted from the data in Figure 1. Its support is 0.5 and its confidence is 1. Mining such rules is very complex and we do not know any efficient strategy to explore the search space for generalized rules. □

As we are interested in very large datasets, an important issue is whether the explicit interestingness evaluation of a collection of patterns remains tractable. The answer can come from the computation of concise representations as defined in [8]. Given a database schema **R**, a dataset **r** and a language of patterns $\mathcal{P}_{\mathbf{R}}$, a concise representation for **r** and $\mathcal{P}_{\mathbf{R}}$ is a structure that makes possible to answer queries of the form "How many times p $\in \mathcal{P}_{\mathbf{R}}$ occur in **r**" approximately correctly and more efficiently than by looking at **r** itself. By the way, some concise representations might enable to provide exact answers.

This paper deals with two related concise representations of binary data, namely frequent sets and *frequent closures*. Not only the extraction of these representations is discussed but we also point out their specific add-value when considered as concise representations for rule mining. Beside well-studied `apriori`-based algorithms, we consider the `close` algorithm that provides frequent closures [10]. We implemented it and made experiments over real data. Furthermore, we propose the new concept of *almost-closure* and sketch the `min-ex` algorithm to mine it. The main idea here is to accept a small incertitude on set frequency since, at that cost, more useful mining tasks become tractable.

2 Frequent Sets As a Concise Representation of Binary Data

At first, we adapt the formal definition of [8] to the kind of concise representation we need. Formally, if an evaluation function $\mathcal{Q}$, a member of Θ (the class of evaluation functions), is an application from a class of structures $\mathcal{S}=\{s_i \mid i \in I\}$ into the interval [0,1], an ϵ-adequate representation for $\mathcal{S}$ with respect to Θ is a class of structures $\mathcal{H}=\{r_i \mid i \in I\}$ and an alternative evaluation function m: $\Theta \times \mathcal{H} \rightarrow [0, 1]$ such that for all $\mathcal{Q} \in \Theta$ and $s_i \in \mathcal{S}$ we have: $\mid \mathcal{Q}(s_i) - m(\mathcal{Q}, r_i) \mid \leq \epsilon$. I denotes a finite (or infinite) index set of $\mathcal{S}$.

Example 4. Let us illustrate the definition on classical concepts from programming languages. Assume $\mathcal{S}$ is a class, e.g. float, s_i is an instance of $\mathcal{S}$, e.g. 0.02, and Θ is the set of proper functions on that class, e.g. $\{sin, cos\}$. A concise representation can be the couple $(\mathcal{H}, m)$, $\mathcal{H}$ being another class, e.g. short, and m an alternative way to evaluate $\mathcal{Q}$, e.g. using a table of values of *sin* and *cos* for all angles from $\{0, 1, \ldots, 359\}$. Now, there is an alternative way to compute sin(x) and cos(x). Instead of s_i=0.02, we store $r_i = round(0.02 \times 360/2\pi)\ mod\ 360$, i.e., 1. When the value of $sin(0.02)$ is needed, we can use $m(sin, 1)$ that returns the value stored in the table associated to *sin*. Clearly, the result is approximate, but the error is bound and the result is known at a much lower cost. □

If the functions from Θ share a lot of intermediate results, and the number of evaluations justifies it, a concise representation can be made of the intermediate results from which all functions from Θ can be evaluated. Such a concise representation avoids going back to the data. The alternative data representation memory requirement might be smaller as well.

Let us now consider the class $\mathcal{S}$ of binary relational schema over the set of attributes $\mathbf{R}$. Instances $s_i \in \mathcal{S}$ are relational tables. A query $\mathcal{Q} \in \Theta$ over an instance s_i of $\mathcal{S}$, denoted $\mathcal{Q}(s_i)$, is a function whose result is to be found with an alternative (ϵ-adequate) representation. $\mathcal{H}$ denotes the alternative class of structures and the counterpart of evaluations, denoted by m, must be a mapping from $\Theta \times \mathcal{H}$ into [0,1]. The error due to the new representation r_i of s_i (thus compared to the result of $\mathcal{Q}(s_i)$ on the original structure) must be at most ϵ for any instance of s_i.

Example 5. Let $\mathbf{r}$ denote a binary relation over $\mathbf{R}=\{A_1, \ldots, A_n\}$ and consider the set $\Theta=\{\mathcal{E}.\text{support}(X, \mathbf{r}) \mid X \subseteq \mathbf{R}\}$, where $\mathcal{E}.\text{support}(X, \mathbf{r})$ is the function that returns the support of X in $\mathbf{r}$ (see Example 1). Given a frequency threshold σ, let FS_σ denote the collection of all frequent sets with their supports. Let $AltSup(X, FS_\sigma)$ denote the support of a frequent set X. FS_σ and the function $m(\mathcal{E}.\text{support}(X, r), FS_\sigma) = AltSup(X, FS_\sigma)$ for $X \in FS_\sigma$, 0 elsewhere, is a σ-adequate representation for Θ over the binary relations defined on $\mathbf{R}$. □

Let us discuss the use of FS_σ as a concise representation for the rule mining task we introduced in Example 2. The support and the confidence of $X \Rightarrow Y$ are exactly known if the support of the rule is at least σ, because the first equals to $AltSup(X \cup Y, FS_\sigma)$ (since $X \cup Y \in FS_\sigma$) and the second equals to $AltSup(X \cup Y, FS_\sigma)/AltSup(X, FS_\sigma)$ (since $X \in FS_\sigma$, too). If it is not the case ($\mathcal{E}.\text{support}(X \Rightarrow Y, \mathbf{r}) < \sigma$), the support is bounded by [0, σ]. If moreover the left-hand side (X) of the rule is frequent, we can bound the confidence of the rule by [0, $\sigma/AltSup(X, FS_\sigma)$]. Otherwise, the confidence can be any number from [0, 1]). FS_σ turns to be a σ-adequate representation for rule support evaluation and a 0.5-adequate representation for rule confidence evaluation. 0.5-adequacy for confidence is clearly insufficient for most of the applications. But if we are interested only in frequent rules (support $\geq \sigma$), we get a 0-adequate representation (so an equivalent representation) for both, the support and the confidence evaluation functions. It explains the effective strategy for extracting all the potentially interesting association rules (w.r.t. frequency and confidence thresholds) from FS_σ: for each $X \in FS_\sigma$ and for each $Y \subset X$, the rule $X \setminus Y \Rightarrow Y$ is kept iff it satisfies the minimum confidence criterion.

Generalized rules (see Example 3) can be evaluated using FS_σ, too. The problem is that the collection FS_σ might not provide some of the needed supports for the computation of rule support and confidence even if the support of the rule is above the support threshold.

Example 6. Assume we want to compute the support and the confidence of the rule $A \wedge \neg E \Rightarrow D$. Applying well-known transformations, we can write the equations: $\mathcal{E}.\text{support}(A \wedge \neg E \Rightarrow D, \mathbf{r})= \mathcal{E}.\text{support}(AD, \mathbf{r}) - \mathcal{E}.\text{support}(ADE, \mathbf{r})$ and $\mathcal{E}.\text{confidence}(A \wedge \neg E \Rightarrow D, \mathbf{r}) = \mathcal{E}.\text{support}(A \wedge \neg E \Rightarrow D, \mathbf{r})$ / ($\mathcal{E}.\text{support}(A, \mathbf{r}) - \mathcal{E}.\text{support}(AE, \mathbf{r})$). These measures can be computed exactly only if A, AD, AE and ADE are frequent sets. □

If we consider several negations and disjunctions, the number of terms will increase and the need for the support of infrequent sets will increase too. Since the computation of the support of all sets is clearly untractable, infrequent conjuncts will give rise to an incertitude [8]. However, this might be acceptable for practical applications. It becomes clear that the adequacy of frequent sets as a concise representation depends on how frequent are the patterns of interest, i.e., the more a pattern is frequent, the less an incertitude will affect the result.

3 Computing Frequent Sets and Frequent Closures

The `apriori` algorithm is defined in [2] and we assume that the reader is familiar with it. It is a levelwise method based on the itemset lattice (i.e., the sets of attributes ordered by set inclusion). The algorithm searches in the lattice starting from singletons and identifies level by level larger frequent sets until the maximal frequent sets are found, i.e., the collection of sets that are frequent while none of their supersets is frequent. This collection is denoted by $Bd^+(FS_\sigma)$ and is called the positive border of FS_σ [13]. A safe pruning strategy (supersets of infrequent sets can not be frequent) has been shown to be the very efficient for the computation of FS_σ in many real-life datasets. One of the identified drawbacks of `apriori`-based algorithms is their untractability for highly correlated data mining. Data are correlated when the truth value of an attribute or a set of attributes determine the truth value of another one (in other terms, association rules with high confidence hold in it). The problem with correlated data originates from the fact that each rule with high confidence pushes the positive border back by one level for a significant part of the itemset lattice (when σ does not change). Highly correlated data contain several such rules, thus pushing back the positive border by several levels. Consequently, the extraction slows down drastically or can even turn to be untractable. An algorithm that would avoid counting support for a great part of frequent sets would accelerate the process. This is the assumption of useful algorithms like `max-miner` [3] that provides $Bd^+(FS_\sigma)$ but not FS_σ. We will consider hereafter an algorithm that avoids counting support for many frequent sets though it provides FS_σ, i.e., every frequent set and its support.

The experiment summarized in Table 1 emphasizes the influence of high correlation of data. We provide the output of the frequent set discovery tool `freddie` that implements an `apriori` algorithm. The left column corresponds to a real dataset from ANPE [1], the right one corresponds to census data (c20d10k) preprocessed at the University of Stanford [2]. We kept in both cases the first 10000 objects and for each object, their 17 first variables (each variable might be encoded in a number of binary attributes). In each column of Table 1, the first information provides the iteration counter (at level k, the level k of the itemset

[1] ANPE is the French national unemployment agency: data10K contains data about unemployed people in december 1998.

[2] ftp://ftp2.cc.ukans.edu/pub/ippbr/census/pums/pums90ks.zip.

Table 1. Mining frequent sets using `freddie` (`apriori`)

```
Input file : data10K              | Input file : base17.txt
Frequency threshold : 0.05        | Frequency threshold : 0.05
    Candidate Frequent Time       |     Candidate  Frequent Time
      sets      sets     (s)      |     sets       sets     (s)
Iter1 :  214     65      0.14     | Iter1 :    317     51      0.15
Iter2 : 2080    602     18.58     | Iter2 :   1275    544     14.60
Iter3 : 2991   2347     78.76     | Iter3 :   3075   2702     92.12
Iter4 : 5738   4935    223.95     | Iter4 :   8101   7940    376.87
Iter5 : 7203   6623    367.86     | Iter5 :  15454  15365    965.41
Iter6 : 6359   5957    391.79     | Iter6 :  20720  20705   1564.63
Iter7 : 3733   3558    257.88     | Iter7 :  19973  19968   1777.45
Iter8 : 1395   1359    105.20     | Iter8 :  13859  13857   1429.21
Iter9 :  304    302     23.13     | Iter9 :   6811   6811    798.39
Iter10 :  32     32      2.70     | Iter10 :  2277   2277    292.68
Iter11 :   1      1      0.48     | Iter11 :   479    479     58.83
Iter12 : No more.                 | Iter12 :    54     54      5.89
Total : 34836 25781   1470.47     | Iter13 :     2      2      0.74
                                  | Iter14 : No more.
                                  | Total : 97080  90755   7376.97
```

lattice is processed). Then, we get the number of candidates, the number of frequent sets and finally the duration of the iteration (CPU time).

The "independance analysis" of the data has shown that ANPE data are slightly correlated while census data are highly correlated. However, the average level of correlation in ANPE data is not low. Typical basket analysis data are much less correlated and would bring down the execution time to a few minutes (and the number of frequent sets would certainly be smaller for $\sigma = 0.05$).

The problem is clearly that a user might want to mine (highly) correlated data with rather low support thresholds while `apriori`-based algorithms become untractable (time, memory) in that cases.

`close` is an algorithm that computes frequent closures in binary data [10]. A set X is a closure in $\mathbf{r}$ when there is no attribute in $\mathbf{R} \setminus X$ that is always true when attributes in X are true. In other words, for each property p not in X, there is a tuple in $\mathbf{r}$ that has all properties of X and does not have the property p. A closure is called a frequent closure when its support in the database is greater than a given threshold σ.

Example 7. In the data from Figure 1, BC is closed while BD is not closed. Indeed, the objects 1 and 3 (the only ones that verify B and D) verify A and C, as well. Furthermore, if σ=0.6, BC is a frequent closure in that data. □

By reducing the number of candidates considered during the extraction (the lattice of closures is generally quite much smaller than the lattice of itemsets, see for instance Figure 2 on the left), `close` can be more efficient than `apriori`. It is straightforward to derive all the frequent sets and their supports from frequent closures.

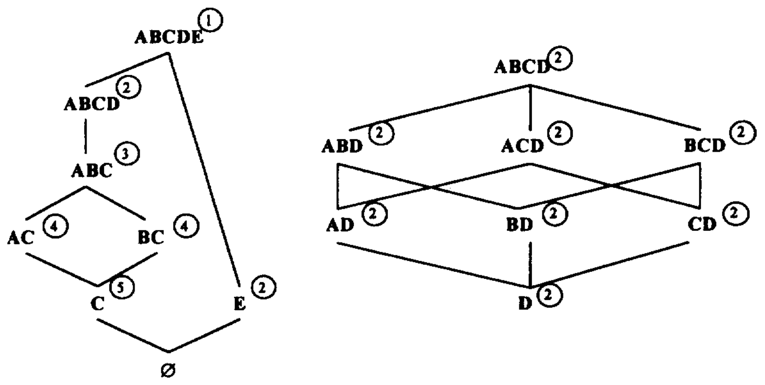

Fig. 2. Closed set lattice (left) and sub-lattice of itemset lattice w.r.t. generator D (right) for the data from Example 1

We now sketch the `close` algorithm and introduce our implementation `close`$_2$. Formal definitions and proofs of properties about `close` are in [10]. Mining closures as a formal basis for association rule mining has also been suggested in [12] though no algorithm was proposed in that paper.

Let FC_σ denote the collection of all frequent closures and their supports. The positive border of FC_σ, $Bd^+(FC_\sigma)$, is the set containing all frequent closures for which no superset of each of them is in FC_σ. It has been proven that, for a given dataset, $Bd^+(FC_\sigma) = Bd^+(FS_\sigma)$.

There are two properties of the itemset lattice on which substantial optimisations can rely. First, the supports of a set and of its closure are the same (see the right part of Figure 2 for an example derived from Example 1). Thus, once identified the closure of a set to be different from this set, we can exclude the closure and all intermediate sets from the support counting procedure since they all have the same support. The sets that go through the support counting procedure are called *generators*. In Figure 2 on the right, it is emphasized that counting the support of generator D, whose closure is $ABCD$, enables to derive the support for the whole sub-lattice. Second, if the closure of X is $X \cup C$, the closure of $X \cup Y$ is a superset of $X \cup Y \cup C$. These properties are used as a base of a safe pruning strategy integrated in `close` [10].

In our implementation `close`$_2$, the extraction of frequent sets is performed in two steps. The first step extracts frequent closures from a binary relation. The extracted closures correspond to all generators. There may be some duplicates, in terms of closures, because different generators may have a same closure. The second step takes that collection of frequent closures, removes duplicates, stores FC_σ set and derives FS_σ. In Table 2, we compare the execution of `close`$_2$ with **`freddie`** on ANPE and census data. The given time is the average CPU time for 2 executions. For `close`$_2$, the time of each step is given. The I/O overhead is provided as the number of scans on the data. We notice that the relative advantage of `close`$_2$ over **`freddie`** is much higher in case of highly correlated data. However, in both cases, the use of `close`$_2$ is worthwhile.

Table 2. Comparison of `freddie (apriori)` and `close2`

	`freddie (apriori)`			`close2`		
Dataset/σ	Time (s)	$\|FS_\sigma\|$	DB scans	Time (s)	$\|FC_\sigma\|$	DB scans
ANPE/0.05	1463.9	25 781	11	69.2/6.2	11 125	9
census/0.05	7377.6	90 755	13	61.7/25.8	10 513	9
ANPE/0.1	254.5	6 370	10	25.5/1.1	2 798	8
census/0.1	2316.9	26 307	12	34.6/6.0	4 041	9
ANPE/0.2	108.4	1 516	9	11.8/0.2	638	7
census/0.2	565.5	5 771	11	18.0/1.1	1 064	9

As it is possible to generate FS_σ from the corresponding FC_σ and $||FS_\sigma|| \geq ||FC_\sigma||$, FC_σ can be considered as a concise representation of the binary relation which is more compact than FS_σ, without any loss of information. Beside efficiency, notice that the postprocessing of frequent closures to get rules can also give rise to a faster computation of useful rules. A first study in that direction concerns the computation of non redundant rules [11].

4 A New Concise Representation: Mining Almost-Closures

This section concerns the concept of almost-closure in binary data. To the best of to our knowledge, this is an original concept. Details about the formalization and the algorithm are available in [6,5].

A fundamental property of set lattices which is used in `close`, is that the same support of the sub-lattice's bottom and top implies the same support for all sets of that sub-lattice. The more the data is correlated (many association rules with confidence 1), the more the collection of frequent closures is compact compared to the collection of frequent sets. We decided to relax the constraint equality of supports, which seems to be a very exigent one, with an "almost-equality" constraint. The new algorithm, called `min-ex`, does not require any association rule with confidence 1 to be present in the mined data. Instead, it can take advantage of a correlation even if it is approximate (the confidence of association rules holding in the data should be however close to 1). These situations might correspond to exceptions in regular behaviours and/or to erroneous tuples that survived preprocessing steps. We expect that, in case of real-life data mining, we will remove much more candidates (w.r.t. `close`) from the support counting procedure, given that `min-ex` pruning strategy is similar to `close` pruning strategy. The trade-off consists in accepting a small incertitude on supports though being able to mine correlated data with lower frequency thresholds. In the following, we consider that the support of a set is the (absolute) number of objects (tuples) in which all the attributes of the set are true. This is different from the definition in Example 1.

Formally, if X (an itemset) "occurs" in t objects within the database, we say that an attribute A is in the almost-closure of X if the support of $X \cup \{A\}$ is at least $t - \delta$ (δ should be small, not to loose the practical relevancy of the extracted information). The almost-closure of X is the set containing all such attributes. Conceptually, a closure is a special case of an almost-closure when δ=0.

Example 8. In data from Figure 1, considering the generator C, one finds that A and B are in the almost-closure of C for δ=1 while none of them was in its closure. □

Now, let us explain where the incertitude comes from. Assume that the almost-closure of X equals to $X \cup \{A, B, C\}$. Let the support of X be s_X, and the supports of $X \cup \{A\}$, $X \cup \{B\}$ and $X \cup \{C\}$ be respectively $s_X - s_A$, $s_X - s_B$ and $s_X - s_C$ where s_A, s_B and s_C are positive numbers lower than δ. We have considered two possibilities for output content. The first stores for each frequent almost-closure: generator items (elements of X, in the example), generator support (s_X) and almost-closure's supplement items (A, B and C). The second adds to each item a from the almost-closure supplement the difference of support between X and $X \cup \{A\}$ (this difference is called miss-number hereafter). In our example that part corresponds to s_A, s_B and s_C. These values have to be known, because to decide if an item is in the almost-closure, they must be at hand. Miss-numbers are values of miss-counters at the end of the corresponding database pass.

The fact, that, for instance, B and C are in the almost-closure of X only implies that they occur almost always with X. Assume that we are in the second case of output (miss-numbers stored). From the supports of X, $X \cup \{B\}$ and $X \cup \{C\}$ we can not infer the support of $X \cup \{B, C\}$, because we do not know if the misses occurred on the same objects (support would be $s_X - max(s_A, s_B)$) or on disjoint ones (support would be $s_X - s_A - s_B$). All intermediate cases are allowed, too. Storing miss-numbers greatly improves the precision of the resulting supports, above all when they are small, compared to δ. Therefore, we choose this solution, even if it increases the volume of output (in terms of quantity of information, not in terms of number of elements). FaC_σ denotes the collection of all frequent almost-closures for threshold σ and is the output of `min-ex`.

An important property about closures has been preserved. Still, if the almost-closure of X is $X \cup C$, the almost-closure of $X \cup Y$ is a superset of $X \cup Y \cup C$. Let us prove it. Attribute A is in the almost-closure of X iff $\mathcal{E}$.support$(X, \mathbf{r})$ − $\mathcal{E}$.support$(X \cup \{A\}, \mathbf{r}) \leq \delta$. In other words, the number of objects that have all properties of X and do not have the property A is at most δ. Clearly, the number of objects satisfying a set of properties can not grow if we enforce that property with a new constraint. Therefore, the number of objects that have all properties of X and all properties of Y and do not have the property A can not be greater than δ. So, all elements of the almost-closure of X (i.e. C) must be present in the almost-closure of $X \cup Y$.

This property may be used as a basis of an efficient safe pruning strategy, analogously to the pruning strategy of `close`. We have been looking for such a strategy. The one implemented in the actual implementation of `min-ex` seems to be reliable [6]. However, in spite of numerous tries, we did not establish a proof that it is safe. We have not found either a counterexample. We checked the completeness in our practical experiments. However, proving the incompleteness or the completeness of our algorithm remains an open problem though it does not prevent its use for practical applications.

Deriving frequent sets from frequent almost-closures is as straightforward as for `close`. The difference is that now there is an incertitude on the support of some frequent sets.

The sub-lattices (corresponding to almost-closures) of which the support range, due to δ, crosses the threshold is kept in the result set, leading to the collection FaC_σ that enables to derive a superset of FS_σ. This is a safety measure: we do not want to prune out sub-lattices of which some itemsets are known to be frequent, for the sake of completeness.

We did several experiments using `min-ex` on census and ANPE datasets (see Table 3). A first remark is that it confirms that `close` and `min-ex` with δ=0 are functionally equivalent. In the case of `close`$_2$, the reduction of the size of FC_σ w.r.t. the corresponding FS_σ highlights the tight-correlation level (relative number of rules with confidence 1) of the data. In the same way, the further reduction of output (——FaC_σ—— compared to ——FC_σ——) for different values of δ, points out the loose-correlation level (relative number of association rules that are nearly "logical" ones).

Let us now discuss the add-value of `min-ex` w.r.t. `close` for highly correlated data mining like census data mining. First of all, we must recall that a too high value of δ might provide a "fuzzy" FaC_σ collection, leading to, e.g., rules with too high incertitude on evaluation functions.

Consider the CPU time needed by the extraction of FaC_σ. It has been more than halved (census data) for δ=6 and the tested frequency thresholds. Next, the I/O activity (number of database passes) has been reduced, an important criterion if the I/O turns to be a bottleneck of the system. A third advantage is that the output collection size has shrunk and we assume that further subsequent knowledge extraction steps will be faster.

Another way to demonstrate the add-value of `min-ex` can be derived from Table 3. We can extract the following concise representations of census data: either $FC_{0.01}$ with `close` or $FaC_{0.005}$ with `min-ex` and $\delta = 2$. It took the same time (154.3 vs. 155.2 sec., 10 passes for both executions) and we got a similar-sized output collection (52166 vs. 55401 itemsets). It is possible without incertitude ($FC_{0.01}$) or with a very good precision (δ=2) on the frequent set supports ($FaC_{0.005}$). The difference is that, using `min-ex`, we gained knowledge about all phenomena of frequency between 0.5% and 1% at almost no price. However, we must notice that in case of uncorrelated data, the memory consumption and CPU load due to maintaining miss-counters may affect the performances (See in Table 3 the extraction time evolution for ANPE/σ=0.05). Only, with a significant reduction of number of candidates (thus only in case of correlated data), the memory consumption will recover (e.g., see ANPE/σ=0.005 or census/σ=0.05).

Applications. A promising application of `min-ex` would be to enable the discovery of repetitive but scarce behaviours. Another application concerns generalized rule mining. Generalized rules, if generated from FS_σ, have an incertitude on

Table 3. Evaluations of implementations `close`$_2$ and `min-ex`

	close$_2$			min-ex			
Dataset/σ	Time (s)	FC_σ	DB scans	δ	Time (s)	FaC_σ	DB scans
ANPE/0.005	816.7	412 092	11	0	851.3	412 092	11
				2	759.5	265 964	11
				4	639.7	182 829	10
				6	553.0	135 136	10
census/0.005	197.8	85 950	10	0	216.2	85 950	10
				2	155.2	55 401	10
				4	118.4	39 036	8
				6	98.5	29 848	8
ANPE/0.01	421.8	161 855	11	0	450.4	161 855	11
				2	466.8	130 765	11
				4	445.1	104 162	10
				6	416.4	84 318	10
census/0.01	154.3	52 166	10	0	166.2	52 166	10
				2	124.9	33 992	10
				4	95.0	24 109	8
				6	79.0	18 822	8
ANPE/0.05	69.2	11 125	9	0	71.5	11 125	9
				2	79.7	11 066	9
				4	85.3	10 931	9
				6	88.4	10 588	9
census/0.05	61.7	10 513	9	0	64.4	10 513	9
				2	50.2	7 294	9
				4	38.2	5 090	8
				6	32.2	4 086	8

measures like support and confidence due to unknown infrequent set supports [8]. Using `min-ex`, it is possible to reduce the bounds of error on evaluation value by supplying the support value for many more itemsets. The incertitude introduced by `min-ex` to some terms of generalized rule evaluation functions can be negligible (w.r.t. function result) compared to the contribution made by the larger number of known terms. Another interesting use is when an approximate result of the data mining step is sufficient. For instance, consider the "sampling" algorithm [7] during its "guess" phase. This phase is supposed to provide an approximation of the collection of frequent sets. An error is inherent to the use of sampling. If we keep the error introduced by the use of almost-closures negligible against the error due to sampling, the guess will be as good as before, but will be computed faster.

5 Conclusion

We studied several concise representations of binary data when data mining processes make use of set support (e.g., when looking for association rules). We studied the `close` algorithm and beside its introduction in [10], we provide a new implementation and experimental evidences about its add-value for the concise representation of (highly) correlated data. It has lead us to the definition of the concept of almost-closure and, here again, we provided experimental evidences of its interest when we are looking for concise representation in difficult cases (correlated data and low frequency thresholds). The discovery of almost-closed frequent sets gave rise to tricky problems w.r.t. the completeness of the mining task. Completeness of `min-ex` remains an open problem at that time and we are currently working on it.

Acknowledgements. The authors thank H. Toivonen from the University of Helsinki for letting us use the `freddie` software tool and the Rhône departmental direction of ANPE who provided data. Last but not least, we want to thank C. Rigotti for stimulating discussions.

References

1. R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In: Proc. *SIGMOD'93*, Washington DC (USA), pages 207 - 216, May 1993, ACM Press.
2. R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In: *Advances in Knowledge Discovery and Data Mining*, pages 307 - 328, 1996, AAAI Press.
3. R.J. Bayardo. Efficiently mining of long patterns from databases. In: Proc. *SIGMOD'98*, Seattle (USA), pages 85 - 93, June 1998, ACM Press.
4. J-F. Boulicaut, M. Klemettinen, and H. Mannila. Modeling KDD processes within the Inductive Database Framework. In: Proc. *DaWak'99*, Florence (I), pages 293 - 302, September 1999, Springer-Verlag, LNCS 1676.
5. J-F. Boulicaut, A. Bykowski, and C. Rigotti. Mining almost-closures in highly correlated data. Research Report LISI INSA Lyon, 2000, 20 pages.
6. A. Bykowski. Frequent set discovery in highly correlated data. Master of Science thesis, INSA Lyon, July 1999, 30 pages.
7. H. Toivonen. Sampling large databases for association rules. In: Proc. *VLDB'96*, Mumbay (India), pages 134 - 145, September 1996, Morgan Kaufmann.
8. H. Mannila and H. Toivonen. Multiple uses of frequent sets and condensed representations. In: Proc. *KDD'96*, Portland (USA), pages 189 - 194, August 1996, AAAI Press.
9. H. Mannila. Inductive databases and condensed representations for data mining. In: Proc. *ILPS'97*, Port Jefferson, Long Island N.Y. (USA), pages 21 - 30, October 1997, MIT Press.
10. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Efficient mining of association rules using closed itemset lattices. *Information Systems*, Volume 24 (1), pages 25 - 46, 1999.
11. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Closed set discovery of small covers for association rules. In: Proc. *BDA'99*, Bordeaux (F), pages 53 - 68, October 1999.
12. M. Zaki and M. Ogihara. Theoretical foundations of association rules. In: Proc. *Workshop post-SIGMOD DMKD'98*, Seattle (USA), pages 85 - 93, June 1998.
13. H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241 - 258, 1997.