Free-sets : a Condensed Representation of Boolean Data for the Approximation of Frequency Queries

To appear in Data Mining and Knowledge Discovery, an International Journal

Jean-François Boulicaut, Artur Bykowski, and Christophe Rigotti

Laboratoire d'Ingénierie des Systèmes d'Information
INSA Lyon, Bâtiment 501
F-69621 Villeurbanne Cedex, France
Tel: +33 4 72 43 85 88 Fax: +33 4 72 43 87 13
{Jean-François.Boulicaut,Artur.Bykowski,Christophe.Rigotti}@insa-lyon.fr

Abstract. Given a large collection of transactions containing items, a basic common data mining problem is to extract the so-called frequent itemsets (i.e., sets of items appearing in at least a given number of transactions). In this paper, we propose a structure called free-sets, from which we can approximate any itemset support (i.e., the number of transactions containing the itemset) and we formalize this notion in the framework of ϵ-adequate representations [10]. We show that frequent free-sets can be efficiently extracted using pruning strategies developed for frequent itemset discovery, and that they can be used to approximate the support of any frequent itemset. Experiments on real dense data sets show a significant reduction of the size of the output when compared with standard frequent itemset extraction. Furthermore, the experiments show that the extraction of frequent free-sets is still possible when the extraction of frequent itemsets becomes intractable, and that the supports of the frequent free-sets can be used to approximate very closely the supports of the frequent itemsets. Finally, we consider the effect of this approximation on association rules (a popular kind of patterns that can be derived from frequent itemsets) and show that the corresponding errors remain very low in practice.

1 Introduction

Several data mining tasks (e.g., association rule mining [1]) are based on the evaluation of frequency queries to determine how often a particular pattern occurs in a large data set. We consider the problem of frequency query evaluation, when patterns are itemsets or conjunctions of properties, in dense data sets[1] like, for instance in the

[1] e.g., data sets containing many strong correlations.

context of census data analysis [5] or log analysis [8]. In these important but difficult cases, there is a combinatorial explosion of the number of frequent itemsets and computing the frequency of all of them turns out to be intractable. In this paper, we present an efficient technique to approximate closely the result of the frequency queries, and formalize it within the ϵ-adequate representation framework [10]. Intuitively, an ϵ-adequate representation is a representation of data that can be substituted to another representation to answer the same kind of queries, but eventually with some lost of precision (bounded by the ϵ parameter). First evidences of the practical interest of such representations have been given in [10, 6].

In this paper, we propose a new ϵ-adequate representation for the frequency queries. This representation, called *free-sets*, is more condensed than the ϵ-adequate representation based on itemsets [10]. The key intuition of the free-set representation is illustrated on the following example. Consider the binary attributes A, B, C, D in the relational table r depicted in Table 1 and suppose we are interested in the support of $\{A, B, C\}$ in r (i.e., the number of rows in r in which A, B and C are true). If we know that the rule $A, B \Rightarrow C$ nearly holds in r (i.e., when A and B are true in a row then, excepted in a few cases, C is also true) then we can approximate the support of itemset $\{A, B, C\}$ using the support of $\{A, B\}$. In Table 1 the rule $A, B \Rightarrow C$ has only one exception. So, we can use the support of $\{A, B\}$ as a value for the support of $\{A, B, C\}$. Moreover, we can approximate the support of any itemset X such that $\{A, B, C\} \subseteq X$ by the support of $X \setminus \{C\}$ because the rule $(X \setminus \{C\}) \Rightarrow C$ also holds with at most a few exceptions. For instance, the support of $\{A, B, C, D\}$ can be approximated by the support of $\{A, B, D\}$ since the rule $A, B, D \Rightarrow C$ can not have more exceptions than $A, B \Rightarrow C$. Furthermore, the support of $\{A, B, D\}$ does not need to be known directly, but can also be approximated itself. For example, the rule $A, D \Rightarrow B$ holds in Table 1 with one exception, so the support of $\{A, D\}$ can be used as an approximation of the support of $\{A, B, D\}$ and then also of the support of $\{A, B, C, D\}$. It should be noticed that the framework presented in this paper can be restricted to rules with no exceptions. In this case, we still benefit of a significant condensation and speed-up when compared with frequent itemset extraction.

A	B	C	D
1	1	0	0
1	0	0	1
0	1	1	1
1	1	1	0
1	0	0	0
1	0	0	0
0	1	0	0
0	1	0	0
0	0	1	0
0	0	1	0
0	0	0	1
0	0	0	1
1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

Table 1. A relational table over four binary attributes.

In the representation proposed in this paper, we call free-set an itemset Y such that the items in Y can not be used to form a nearly exact rule. For example, if we consider only rules having at most one exception, then the free-sets in Table 1 are $\{\emptyset, \{A\}, \{B\}, \{C\}, \{D\}, \{A, B\}, \{A, C\}, \{A, D\}, \{B, C\}, \{B, D\}, \{C, D\}\}$. All other subsets of $\{A, B, C, D\}$ contain items that can be used to form rules with zero or one exception (e.g., $A, B \Rightarrow C$ for $\{A, B, C\}$, $B, D \Rightarrow A$ for $\{A, B, D\}$, $A, C \Rightarrow D$ for $\{A, C, D\}$, $B, C \Rightarrow D$ for $\{B, C, D\}$, and $A, B, C \Rightarrow D$ for $\{A, B, C, D\}$) and thus are not free.

The freeness of itemsets is anti-monotonic in that sense that if a set is not a free-set then none of its supersets can be a free-set. The algorithm proposed to extract the free-sets takes advantage of this property. It first considers sets of size 0 (i.e., the empty itemset), then sets of size 1, and so on. When it determines that a set X is not free then it prunes the search space since there is no need to consider any of the supersets of X. For example, if the algorithm is executed on Table 1 and takes into account rules having at most one

exception, then it will never consider the set $\{A, B, C, D\}$ because several sets among its subsets are not free (e.g., $\{A, B, C\}$).
The experiments show that *frequent* free-sets are an ϵ-adequate representation for frequency queries that can be extracted efficiently, even on dense data sets. They also show that the error made when approximating itemset support using frequent free-sets remains very low in practice.
Finally, we consider a popular application of frequent itemset discovery: the production of the so-called association rules [1]. We determine bounds for the errors propagated on association rule characteristics when we use frequent free-sets to approximate the support of frequent itemsets, and we show that these bounds are very tight in practice.
This paper is a significant extension of a preliminary work presented in [7]. It includes proofs of the theorems, an in-depth error evaluation.

Organization of the paper. In the next section we introduce preliminary definitions used in this paper. In Section 3, we present the notion of free-set, and show that it can be used as an ϵ-adequate representation for the frequency queries. In section 4, we present an algorithm to extract the frequent free-sets. In Section 5, we give practical evidences that frequent free-sets can be extracted efficiently and that the estimation of the supports of frequent itemsets using frequent free-sets leads in practice to very low errors. In Section 6, we show that these errors are propagated in a very favorable way when we produce association rules. We review related work in Section 7. Finally, we conclude with a summary and directions for future work.

2 Preliminary definitions

When applicable, we use the notational conventions and definitions from [10, 11].

2.1 Frequent sets

In this section, we recall standard definitions.

Definition 1 (binary database). Let R be a set of symbols called *items*. A *row* (also called *transaction*) is a subset of R. A *binary database* r *over* R is a multiset of transactions.

Definition 2 (support and frequency). We note $\mathcal{M}(r, X) = \{t \in r | X \subseteq t\}$ the multiset of rows matched by the itemset X and $Sup(r, X) = |\mathcal{M}(r, X)|$ the *support* of X in r, i.e., the number of rows matched by X. The *frequency* of X in r is $Sup(r, X)/|r|$. Let σ be a frequency threshold, $Freq(r, \sigma) = \{X | X \subseteq R$ and $Sup(r, X)/|r| \geq \sigma\}$ is the set of all σ-frequent itemsets in r.

For notational convenience, we also need the following specific definition.

Definition 3 (frequent sets). $FreqSup(r, \sigma)$ is the set of all pairs containing a frequent itemset and its support, i.e., $FreqSup(r, \sigma) = \{\langle X, Sup(r, X)\rangle | X \subseteq R$ and $Sup(r, X)/|r| \geq \sigma\}$.

2.2 ϵ-adequate representation

Definition 4 (ϵ-adequate representation [10]). Let $\mathcal{S}$ be a class of structures. Let $\mathcal{Q}$ be a class of queries for $\mathcal{S}$. The value of a query $Q \in \mathcal{Q}$ on a structure $s \in \mathcal{S}$ is assumed to be a real number in $[0, 1]$ and is denoted by $Q(s)$. An ϵ-adequate representation for $\mathcal{S}$ w.r.t. a class of queries $\mathcal{Q}$, is a class of structures $\mathcal{C}$, a representation mapping $rep : \mathcal{S} \to \mathcal{C}$ and a query evaluation function $m : \mathcal{Q} \times \mathcal{C} \to [0, 1]$ such that $\forall Q \in \mathcal{Q}, \forall s \in \mathcal{S}, |Q(s) - m(Q, rep(s))| \leq \epsilon$.

Example 1. An example of a class of structures is the set noted $\mathcal{DB}_R$ of all possible binary databases over a set of items R. An interesting query class is $\mathcal{Q}_R$, the set of all queries retrieving the frequency of an itemset $\subseteq R$. If we denote Q_X the query in $\mathcal{Q}_R$ asking for the frequency of itemset X then $\mathcal{Q}_R = \{Q_X | X \subseteq R\}$ and the value of Q_X on a database instance $r \in \mathcal{DB}_R$ is defined by $Q_X(r) = Sup(r, X)/|r|$.
An example of ϵ-adequate representation for $\mathcal{DB}_R$ w.r.t. $\mathcal{Q}_R$ is the representation of $r \in \mathcal{DB}_R$ by means of $Freq(r, \epsilon)$. The corresponding $rep, \mathcal{C}$ and m are as follows. $\forall r \in \mathcal{DB}_R, rep(r) = FreqSup(r, \epsilon)$,

$\mathcal{C} = \{rep(r) | r \in \mathcal{DB}_R\}$, $\forall Q_X \in \mathcal{Q}_R, \forall c \in \mathcal{C}$, if $\exists \langle X, \alpha \rangle \in rep(r)$ then $m(Q_X, c) = \alpha / |r|$ else $m(Q_X, c) = 0$. It is straightforward to see that this is an ϵ-adequate representation for $\mathcal{DB}_R$ w.r.t. $\mathcal{Q}_R$ since $\forall Q_X \in \mathcal{Q}_R, \forall r \in \mathcal{DB}_R, |Q_X(r) - m(Q_X, rep(r))| \leq \epsilon$.

Interesting ϵ-adequate representations are *condensed representations*, i.e., ϵ-adequate representations where structures have a smaller size than the original structures.

3 The free-sets as a condensed representation

First, we recall the notion of *association rule*, and then define a class of rules called δ-*strong rules* in order to introduce the concept of free-set in a concise way.

Definition 5 (association rule). Let R be a set of items, an *association rule* based on R is an expression of the form $X \Rightarrow Y$, where $X, Y \subseteq R$, $Y \neq \emptyset$ and $X \cap Y = \emptyset$.

Definition 6 (δ-strong rule). A δ-*strong rule*[2] in a binary database r over R is an association rule $X \Rightarrow Y$ such that $Sup(r, X) - Sup(r, X \cup Y) \leq \delta$, i.e., the rule is violated in no more than δ rows.

In this definition, δ is supposed to have a small value, so a δ-strong rule is intended to be a rule with very few exceptions.

3.1 Free-sets

Definition 7 (δ-free-set). Let r be a binary database over R, $X \subseteq R$ is a δ-*free-set* w.r.t. r if and only if there is no δ-strong rule based on X in r. The set of all δ-free-sets w.r.t. r is noted $Free(r, \delta)$.

Since δ is supposed to be rather small, informally, a free-set is a set of items such that its subsets (seen as conjunction of properties) are not related by any very strong positive correlation.
One of the most interesting properties of *freeness* is its *anti-monotonicity* w.r.t. itemset inclusion.

[2] Stemming from the notion of *strong rule* of [15]

Definition 8 (anti-monotonicity). A property ρ is *anti-monotone* if and only if for all itemsets X and Y, $\rho(X)$ and $Y \subseteq X$ implies $\rho(Y)$.

The anti-monotonicity has been identified as a key property for efficient pattern mining [11, 12], since it is the formal basis of a safe pruning criterion. Indeed, efficient frequent set mining algorithms like APRIORI [2] make use of the (anti-monotone) property *"is frequent"* for pruning.
The anti-monotonicity of freeness follows directly from the definition of free-set and is stated by the following theorem.

Theorem 1. *Let X be an itemset. For all $Y \subseteq X$ if $X \in Free(r, \delta)$ then $Y \in Free(r, \delta)$.*

3.2 Free-sets as an ϵ-adequate representation

We show now that δ-free-sets can be used to answer frequency queries with a bounded error. The following lemma states that the support of any itemset can be approximated using the support of one of the free-sets.

Lemma 1. *Let r be a binary database over a set of items R, $X \subseteq R$ and $\delta \in [0, |r|]$, then there exists $Y \subseteq X$ such that $Y \in Free(r, \delta)$ and $Sup(r, Y) \geq Sup(r, X) \geq Sup(r, Y) - \delta|X|$.*

Proof. We show this using a recurrence on $|X|$. The statement is true for $|X| = 0$ if we take $Y = \emptyset$. Suppose the statement is true for $|X| = i$. Let X be a subset of R such that $|X| = i + 1$. If $X \in Free(r, \delta)$ then we can simply choose $Y = X$. If $X \notin Free(r, \delta)$ then by definition of $Free(r, \delta)$ there exists a δ-strong rule $Z_1 \rightarrow Z_2$ based on X. Let A be an item in Z_2 and $Z_3 = X \setminus \{A\}$. As $|Z_3| = |X| - 1$ using the recurrence hypothesis we know that there exists $Y \subseteq Z_3$ such that $Y \in Free(r, \delta)$ and $Sup(r, Z_3) \geq Sup(r, Y) - \delta|Z_3|$. Since $Z_1 \rightarrow Z_2$ is a δ-strong rule, then $Sup(r, Z_1) - Sup(r, Z_1 \cup Z_2) \leq \delta$. $Sup(r, Z_1) - Sup(r, Z_1 \cup Z_2)$ is the number of rows not matched by Z_2 but matched by Z_1, thus $Sup(r, Z_1) - Sup(r, Z_1 \cup Z_2)$ is greater or equal to $Sup(r, Z_1 \cup Z_3) - Sup(r, Z_1 \cup Z_2 \cup Z_3)$ (i.e., the number of rows not matched by Z_2 but matched by Z_1 and Z_3). So we have $Sup(r, Z_1 \cup Z_3) - Sup(r, Z_1 \cup Z_2 \cup Z_3) \leq \delta$ which simplifies

to $Sup(r, Z_3) - Sup(r, X) \leq \delta$. Since $Sup(r, Z_3) \geq Sup(r, Y) - \delta|Z_3|$ and $|Z_3| = |X| - 1$ we deduce $Sup(r, X) \geq Sup(r, Y) - \delta|X|$. The other inequality $Sup(r, Y) \geq Sup(r, X)$ is straightforward because $Y \subseteq Z_3 \subseteq X$. □

This lemma states that the support of an itemset X can be approximated using the support of one of the free-sets, but it does not determine which free-set to use. We now show that this can be done by simply choosing among the free-sets included in X any free-set with a minimal support value. This is stated more formally by the following theorem.

Theorem 2. *Let r be a binary database over a set of items R, $X \subseteq R$ and $\delta \in [0, |r|]$, then for any $Y \subseteq X$ such that $Y \in Free(r, \delta)$ and $Sup(r, Y) = min(\{Sup(r, Z) | Z \subseteq X \text{ and } Z \in Free(r, \delta)\})$ we have $Sup(r, Y) \geq Sup(r, X) \geq Sup(r, Y) - \delta|X|$.*

Proof. Let Y be a subset of X such that $Y \in Free(r, \delta)$ and satisfying $Sup(r, Y) = min(\{ Sup(r, Z) | Z \subseteq X \text{ and } Z \in Free(r, \delta)\})$. Since $Y \subseteq X$ we have immediately that $Sup(r, Y) \geq Sup(r, X)$. By Lemma 1, there exists $Z \subseteq X$ such that $Z \in Free(r, \delta)$ and $Sup(r, Z) \geq Sup(r, X) \geq Sup(r, Z) - \delta|X|$. Since Y has the minimal support among all subsets of X in $Free(r, \delta)$, then $Sup(Z) \geq Sup(Y)$. Thus $Sup(Z) - \delta|X| \geq Sup(Y) - \delta|X|$. As $Sup(r, X) \geq Sup(r, Z) - \delta|X|$, we have $Sup(r, X) \geq Sup(Y) - \delta|X|$. □

In practice, computing the whole collection of δ-free-sets is often intractable. We show now that such an exhaustive mining can be avoided since an ϵ-adequate representation to answer frequency queries can be obtained if we extract only *frequent* free-sets together with a subset of the corresponding negative border [11].

Definition 9 (frequent free-set). Let r be a binary database over a set of items R, we denote $FreqFree(r, \sigma, \delta) = Freq(r, \sigma) \cap Free(r, \delta)$ the set of σ-frequent δ-free-sets w.r.t. r.

Let us adapt the concept of negative border from [11] to our context.

Definition 10 (negative border of frequent free-sets). Let r be a binary database over a set of items R, the negative border of

$FreqFree(r, \sigma, \delta)$ is denoted by $\mathcal{B}d^-(r, \sigma, \delta)$ and is defined as follows: $\mathcal{B}d^-(r, \sigma, \delta) = \{X | X \subseteq R, X \notin FreqFree(r, \sigma, \delta) \wedge (\forall Y \subset X, Y \in FreqFree(r, \sigma, \delta))\}$.

Informally, the negative border $\mathcal{B}d^-(r, \sigma, \delta)$ consists of the smallest itemsets (w.r.t. set inclusion) that are not σ-frequent δ-free. Our approximation technique only needs a subset of the negative border $\mathcal{B}d^-(r, \sigma, \delta)$. This subset, denoted by $Free\mathcal{B}d^-(r, \sigma, \delta)$, is the set of all free-sets in $\mathcal{B}d^-(r, \sigma, \delta)$.

Definition 11. $Free\mathcal{B}d^-(r, \sigma, \delta) = \mathcal{B}d^-(r, \sigma, \delta) \cap Free(r, \delta)$

As in the case of an ϵ-adequate representation for $\mathcal{DB}_R$ w.r.t. $\mathcal{Q}_R$ using frequent itemsets (see Section 2.2), we need the free-sets and their supports.

Definition 12. $FreqFreeSup(r, \sigma, \delta)$ is the set of all pairs containing a frequent free-set and its support, i.e., $FreqFreeSup(r, \sigma, \delta) = \{\langle X, Sup(r, X)\rangle | X \in FreqFree(r, \sigma, \delta)\}$.

We can now define the ϵ-adequate representation w.r.t. the frequency queries.

Definition 13. The *frequent free-sets representation* w.r.t. σ, δ and a query class $\mathcal{Q} \subseteq \mathcal{Q}_R$, is defined by a class of structures $\mathcal{C}$, a representation mapping rep and a query evaluation function m, where $\forall r \in \mathcal{DB}_R, rep(r) = \langle FreqFreeSup(r, \sigma, \delta), Free\mathcal{B}d^-(r, \sigma, \delta)\rangle$, $\mathcal{C} = \{rep(r) | r \in \mathcal{DB}_R\}$, $\forall Q_X \in \mathcal{Q}, \forall c \in \mathcal{C}$, if $\exists Y \in Free\mathcal{B}d^-(r, \sigma, \delta), Y \subseteq X$ then $m(Q_X, c) = 0$ else $m(Q_X, c) = min(\{\alpha | \exists Z \subseteq X, \langle Z, \alpha\rangle \in FreqFreeSup(r, \sigma, \delta)\})/|r|$.

Using this representation, the frequency of an itemset X is approximated as follows. If X has a subset Y which is free but not frequent then the frequency of X is considered to be 0. Otherwise we take the smallest support value among the supports of the subsets of X that are free and frequent.
We now establish that this representation is an ϵ-adequate representation for the following database class and query class.

Definition 14. $\mathcal{DB}_{R,s} = \{r | r \in \mathcal{DB}_R$ and $|r| \geq s\}$, i.e., the set of all binary databases having at least s rows. $\mathcal{Q}_{R,n} = \{Q_X | X \subseteq R$ and $|X| \leq n\}$, i.e., the set of frequency queries on itemsets having no more than n items.

Theorem 3. *A* frequent free-sets representation *w.r.t.* σ, δ *and a query class* $\mathcal{Q}_{R,n}$ *is an* ϵ*-adequate representation for* $\mathcal{DB}_{R,s}$ *w.r.t.* $\mathcal{Q}_{R,n}$ *where* $\epsilon = max(\sigma, n\delta/s)$.

Proof. Let Q_X be a query in $\mathcal{Q}_{R,n}$ and r an database in $\mathcal{DB}_{R,s}$. If there exists $Y \in Free\mathcal{B}d^-(r, \sigma, \delta)$ such that $Y \subseteq X$ then X is not σ-frequent so $Q_X(r) \leq \sigma$. Since $m(Q_X, c) = 0$ we have $|Q_X(r) - m(Q_X, rep(r)| \leq \sigma$.
In the case where no $Y \in Free\mathcal{B}d^-(r, \sigma, \delta)$ is a subset of X, this means that all δ-free-set included in X are σ-frequent. Whence $min(\{Sup(r, Z)|Z \subseteq X \text{ and } Z \in Free(r, \delta)\}) = min(\{\alpha|\exists Z \subseteq X, \langle Z, \alpha\rangle \in FreqFreeSup(r, \sigma, \delta)\})$ which is equal to $m(Q_X, rep(r))$. Thus, by Theorem 2, $m(Q_X, rep(r)) \geq Q_X(r) \geq m(Q_X, rep(r)) - \delta|X|/|r|$. So we have $|Q_X(r) - m(Q_X, rep(r))| \leq n\delta/s$. □

4 Discovering all frequent free-sets

In this section, we describe an algorithm, called MinEx, that generates all frequent free-sets. For clarity, we omit the fact that it outputs their supports as well. Implementation issues are presented in Section 4.2.

4.1 The algorithm - an abstract version

MinEx can be seen as an instance of the levelwise search algorithm presented in [11]. It explores the itemset lattice (w.r.t. set inclusion) levelwise, starting from the empty set and stopping at the level of the largest frequent free-sets. More precisely, the collection of candidates is initialized with the empty set as single member (the only set of size 0) and then the algorithm iterates on candidate evaluation and larger candidate generation. At each iteration of this loop, it scans the database to find out which candidates of size i are frequent free-sets. Then, it generates candidates for the next iteration, taking every set of size $i + 1$ such that all proper subsets are frequent free-sets. The algorithm finishes when there is no more candidate. The algorithm is given below as Algorithm 1.

Algorithm 1 (MinEx)
Input: *r a binary database over a set of items R, σ and δ two thresholds.*
Output: $FreqFree(r, \sigma, \delta)$

1. $\mathcal{C}_0 := \{\emptyset\}$;
2. $i := 0$;
3. **while** $\mathcal{C}_i \neq \emptyset$ **do**
4. $\quad \mathcal{F}req\mathcal{F}ree_i := \{X | X \in \mathcal{C}_i$ *and* X *is a* σ*-frequent* δ*-free-set in* $r\}$;
5. $\quad \mathcal{C}_{i+1} := \{X | X \subseteq R$ *and* $\forall\, Y \subset X, Y \in \bigcup_{j \leq i} \mathcal{F}req\mathcal{F}ree_j\} \setminus \bigcup_{j \leq i} \mathcal{C}_j$;
6. $\quad i := i + 1$;
7. **od**;
8. **output** $\bigcup_{j<i} \mathcal{F}req\mathcal{F}ree_j$;

Using the correctness result of the levelwise search algorithm given in [11] the following theorem is straightforward.

Theorem 4 (Correctness). Algorithm MinEx computes the sets of all σ-frequent δ-free-sets.

4.2 Implementation issues

We used techniques similar to the ones described in [3] for frequent itemset mining. The candidate generation is made using a join-based function, and the itemset support counters are updated w.r.t. a row of the database using a *prefix-tree* data structure.
The key point that needs a new specific technique is the freeness test in step 4 of the algorithm. An efficient computation of this test can be done, based on the following remark: Z is not a δ-free-set if and only if there exist $A \in Z$ and $X = Z \setminus \{A\}$ such that X is not δ-free or X is δ-free and $X \Rightarrow \{A\}$ is a δ-strong rule. Furthermore, the step 5 of the algorithm guarantees that if Z is a candidate then X must be δ-free since X is a subset of Z. Therefore, during the i^{th} iteration, we might first compute the δ-strong rules of the form $X \Rightarrow \{A\}$, where $X \in \mathcal{F}req\mathcal{F}ree_i$ and $A \in R \setminus X$, and then use them to remove candidates in $\mathcal{C}_{i+1}$ that are not δ-free. Thus, at the beginning of an iteration, only free-sets are candidates.

This is incorporated in the algorithm by replacing steps 4 and 5 with the following steps:

4.1 $\mathcal{F}req\mathcal{F}ree_i := \{X | X \in \mathcal{C}_i$ and X is a σ-frequent$\}$;
4.2 $\mathcal{N}ot\mathcal{F}ree_{i+1} := \{Z | Z = X \cup \{A\}$ where $X \in \mathcal{F}req\mathcal{F}ree_i$, $A \in R \setminus X$ and $X \Rightarrow \{A\}$ is a δ-strong rule $\}$;
5.1 $\mathcal{C}^g_{i+1} := \{X | X \subseteq R$ and $\forall\, Y \subset X, Y \in \bigcup_{j \leq i} \mathcal{F}req\mathcal{F}ree_j\} \setminus \bigcup_{j \leq i} \mathcal{C}_j$;
5.2 $\mathcal{C}_{i+1} := \mathcal{C}^g_{i+1} \setminus \mathcal{N}ot\mathcal{F}ree_{i+1}$;

The steps 4.1 and 4.2 can be computed efficiently within the same database scan as follows. For each candidate X considered in step 4.1, we maintain a node n (in the prefix-tree) containing an integer denoted by $n.count$ to count the support of X and a set denoted by $n.rhs$ to determine the δ-strong rule having a left hand side equal to X. More precisely, $n.rhs$ is a set of pairs of the form $\langle A, e\rangle$. Such a pair $\langle A, e\rangle$ means that the rule $X \Rightarrow \{A\}$ has e exceptions.
Steps 4.1 and 4.2 are performed by first initializing for each candidate $X \in \mathcal{C}_i$ the corresponding node n in the prefix-tree with $n.count := 0$ and $n.rhs := \emptyset$. Then the database r is scanned, and for each row t the prefix-tree is used to find all candidates matching t. For each such candidate X, corresponding to a node n in the tree, we call $matched(t, X, n, \delta)$ to update $n.count$ and $n.rhs$. The description of *matched* is given below as Algorithm 2.

Algorithm 2 (*matched*)
Input: *a row t, a candidate X, a node n of the prefix-tree and the threshold δ.*
Output: *n updated.*

1. **if** $n.count \leq \delta$ **then**
2. **for** *all* $i \in t \setminus X$ **do**
3. **if** $\nexists \langle j, e\rangle \in n.rhs$ *with* $j = i$ **then**
4. $n.rhs := n.rhs \cup \{\langle i, n.count\rangle\}$;
5. **fi**
6. **od**
7. **fi**
8. **for** *all* $\langle j, e\rangle \in n.rhs$ **do**
9. **if** $j \notin t$ **then**

```
10.                 n.rhs := n.rhs \ {⟨j, e⟩};
11.                 if e < δ then
12.                         n.rhs := n.rhs ∪ {⟨j, e + 1⟩};
13.                 fi
14.         fi
15.   od
16.   n.count := n.count +1;
17.   output n;
```

The key idea is that the set $n.rhs$ is created lazily, in the sense that a pair in $n.rhs$ is created for an item A only when the algorithm finds A in a transaction t matched by X. Moreover, when δ rows matched by X have been encountered, there is no need to create new entries for new items in $n.rhs$ since these items will lead obviously to rules with more than δ exceptions.

5 Experiments

The running prototype is implemented in C++. We use a PC with 512 MB of memory and a 500 MHz Pentium III processor under Linux operating system.
For an experimental evaluation, we chose the PUMSB* data set, a PUMS census data set[3] preprocessed by researchers from IBM Almaden Research Center. The particularity of PUMS data sets is that they are very dense and make the mining of all frequent itemsets together with their supports intractable for low frequency thresholds, because of the combinatorial explosion of the number of frequent itemsets [5].

5.1 Frequent free-set vs. frequent set condensation

Table 2 shows a comparison of the extraction of frequent sets and frequent free-sets for different frequency thresholds and different values of δ. The collections $FreqFree(r, \sigma, \delta)$ are significantly smaller than the corresponding $Freq(r, \sigma)$. For frequency thresholds of 15% and 20% $Freq(r, \sigma)$ is so large that it is clearly impossible to provide it on our platform, while the extraction of $FreqFree(r, \sigma, \delta)$

[3] http://www.almaden.ibm.com/cs/quest/data/long_patterns.bin.tar

remains tractable. For these two frequency thresholds of 15% and 20%, we use lower-bound estimations of $|Freq(r, \sigma)|$. These lower-bounds are computed using the δ-strong rules collected by MinEx (see Section 4.2) to find the size of the largest frequent itemset. If this size is m then there are a least 2^m frequent itemsets. Figure 1 (left) emphasizes, using logarithmically scaled axes, the difference of the size of the various representations. We observe a brutal change between the size of $Freq(r, 0.25)$ and of $Freq(r, 0.20)$: 1000 times more frequent itemsets than expected by extrapolating the trend given by $Freq(r, 0.25)$ and $Freq(r, 0.30)$. If we look at the trend of the number of frequent free-sets it seems to be unchanged. The reason for this, is that between 0.25 and 0.20 we reach a support threshold where the number of strong rules increases significantly and then leads to the explosion of the number of frequent itemsets, but not to the explosion of the number of frequent free-sets.

Using also logarithmically scaled axes, Figure 1 (right) shows that the extraction time for MinEx grows up exponentially when the frequency threshold is reduced. This is due to the combinatorial explosion of the number of frequent free-sets. Apriori-based algorithms have a similar exponential evolution of the extraction time, due in this case to the combinatorial explosion of the number of frequent sets.

σ	15%			20%			25%			30%		
δ	0	10	20	0	10	20	0	10	20	0	10	20
Max frequent free-set size (≈MIN-EX DB scans)	12	11	10	12	10	9	11	9	9	10	9	8
$\lvert FreqFree(r, \sigma, \delta)\rvert$	909 806	324 743	232 887	253 107	105 615	76 413	78 220	36 310	27 137	26 972	14 631	11 079
$FreqFree(r, \sigma, \delta)$ extraction time in sec. (MIN-EX)	11 977	6 590	5 126	4 233	2 342	1 890	1 540	905	731	533	373	302
Max frequent set size (≈APRIORI DB scans)	35			32			18			16		
$\lvert Freq(r, \sigma)\rvert$	$>2^{35}$			$>2^{32}$			2 064 946			432 699		
$Freq(r, \sigma)$ extraction time in sec. (APRIORI)	N/A			N/A			14 559			3 469		

Table 2. Comparison of different representations at various frequency thresholds.

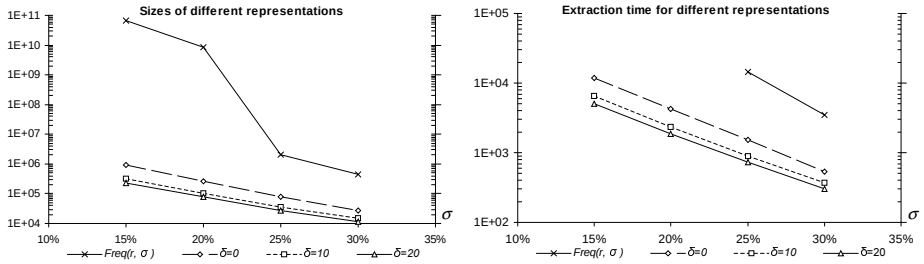

Fig. 1. Extraction time and sizes of different representations.

5.2 Scale-up experiment

On Figure 2, we report the extraction time (for $\sigma = 20\%$) when changing the number of rows or the number of items in the data set. We observe an exponential complexity w.r.t. the number of items and a linear complexity w.r.t. number of rows in the data set if the value of δ follows the number of tuples (e.g., if we double the number of rows then we double the value of δ). This is emphasized by a superimposed straight line on Figure 2 (left).

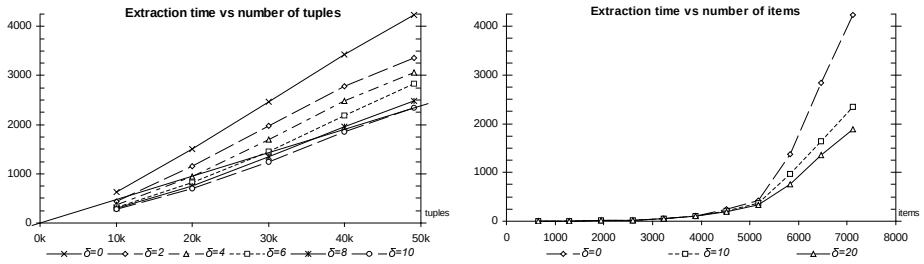

Fig. 2. Behavior of MinEx w.r.t. the number of rows and the number of items.

5.3 Approximation error in practice

In this section we report the practical error made on σ-frequent itemset supports when using the approximation based on σ-frequent δ-free-sets. This evaluation is made on the PUMSB* data set used in the previous experiments and also on a PUMS data set of Kansas in a less favorable case.

In the PUMSB* data set, for $\sigma = 0.3$, there are 432699 σ-frequent sets and the largest has $n = 16$ items. We computed the condensed representation $FreqFreeSup(r, 0.3, 20)$ which contains 11079 elements.
Theoretical error bounds for the frequent set support approximation can be determined using Theorem 2 as follows. In this experiment, the maximal absolute support error is $\delta * n = 20 * 16 = 320$ rows. The maximal relative support error can be obtained assuming that the maximal theoretical absolute error occurs on the σ-frequent set of minimal frequency (i.e., σ). The PUMSB* data set contains $N = 49046$ rows. So, the maximal relative support error is $\delta * n/(N * \sigma) = 2.18\%$.
The support of each of the 432699 σ-frequent itemsets is approximated using the collection $FreqFreeSup(r, 0.3, 20)$ and Theorem 2 and then compared to the exact support. The maximal observed absolute support error is 45 rows, and the maximal relative support error is 0.29%. The average absolute support error is 6.01 rows and the average relative support error is 0.037%. Tables 3 and 4 show that these errors remain very low even for frequent sets containing a lot of items and for low supports.

itemset size	1	2	3	4	5	6	7	8
average abs. sup. error	0	0.36	1.17	2.14	3.24	4.33	5.31	6.12
average rel. sup. error	0	0.002%	0.007%	0.012%	0.019%	0.026%	0.032%	0.038%
maximal abs. sup. error	0	18	20	37	37	39	39	45
maximal rel. sup. error	0	0.11%	0.13%	0.18%	0.22%	0.24%	0.24%	0.29%

itemset size	9	10	11	12	13	14	15	16
average abs. sup. error	6.80	7.40	7.92	8.39	8.82	9.22	9.58	9.86
average rel. sup. error	0.042%	0.046%	0.050%	0.054%	0.057%	0.060%	0.062%	0.064%
maximal abs. sup. error	45	45	45	44	38	31	24	15
maximal rel. sup. error	0.29%	0.29%	0.29%	0.28%	0.26%	0.19%	0.15%	0.10%

Table 3. Error observed on σ-frequent itemset supports by itemset size.

In this experiment the value of δ is small w.r.t. the minimal support required. The ratio is $20/(0.3 * 49046) = 0.136\%$. We now report another experiment where the value of δ represents more than 1% of the minimal support required, and thus is likely to greatly increase the value of the error.

itemset support (%)	[30,40]	(40,50]	(50,60]	(60,70]	(70,80]	(80,90]	(90,100]
average abs. sup. error	6.20	3.29	0.07	0	0	0	0
average rel. sup. error	0.039%	0.016%	2.8×10^{-6}	0	0	0	0
maximal abs. sup. error	45	38	9	0	0	0	0
maximal rel. sup. error	0.29%	0.19%	0.03%	0	0	0	0

Table 4. Error observed on σ-frequent itemset supports by interval of support.

The data set used in this experiment is a PUMS data set of Kansas state[4]. We use a version of this data set that has been preprocessed at the University of Clermont-Ferrand (France) in Prof. L. Lakhal's research group. We have reduced this data set to 10000 rows and 317 items to be able to extract all σ-frequent itemsets at a low frequency threshold. For $\sigma = 0.05$ (500 rows), there are 90755 σ-frequent sets and the largest has $n = 13$ items. We computed $FreqFreeSup(r, 0.05, 6)$ which contains 4174 elements.
In this experiment, the maximal absolute support error is $\delta * n = 6*13 = 78$ rows. The maximal relative support error is $\delta*n/(N*\sigma) = 15.6\%$ ($N = 10000$ rows in the experiment).
The supports of the σ-frequent itemsets are approximated using $FreqFreeSup(r, 0.05, 6)$ and compared to the exact supports. The maximal observed absolute support error is 18 rows, and the maximal relative support error is 3.1%. The average absolute support error is 2.12 rows and the average relative support error is 0.28%. A more detailed distribution of the error is given in Tables 5 and 6. These results show that the error remains low in practice even when the value of δ is high w.r.t. the minimal support.

6 Effect of errors on association rules

A popular application of the extraction of frequent itemsets is the discovery of association rules [1]. In this section, we give bounds for the error made on support and confidence of association rules when these rules are derived from frequent δ-free-sets instead of frequent itemsets. The notion of association rules has been recalled in Definition 5. Support and confidence are the two most widely used objec-

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

itemset size	1	2	3	4	5	6	7	8
average abs. sup. error	0	0.24	0.65	1.10	1.53	1.92	2.31	2.75
average rel. sup. error	0	0.03%	0.07%	0.13%	0.18%	0.24%	0.31%	0.38%
maximal abs. sup. error	0	6	10	12	14	18	18	18
maximal rel. sup. error	0	1.1%	1.3%	2.1%	2.7%	3.1%	3.1%	3.1%

itemset size	9	10	11	12	13
average abs. sup. error	3.28	3.90	4.58	5.20	5.50
average rel. sup. error	0.47%	0.58%	0.71%	0.83%	0.88%
maximal abs. sup. error	18	18	18	15	11
maximal rel. sup. error	3.1%	2.9%	2.9%	2.9%	2.0%

Table 5. Error observed on σ-frequent itemset supports by itemset size.

itemset support (%)	[5,10]	(10,20]	(20,30]	(30,40]	(40,50]	(50,60]	(60,70]	(70,80]	(80,90]	(90,100]
average abs. sup. error	2.16	2.03	2.22	2.03	1.25	1.70	0.66	0	0	0
average rel. sup. error	0.337%	0.159%	0.089%	0.063%	0.027%	0.031%	0.010%	0	0	0
maximal abs. sup. error	18	14	10	10	5	6	6	0	0	0
maximal rel. sup. error	3.11%	1.17%	0.47%	0.33%	0.12%	0.10%	0.10%	0	0	0

Table 6. Error observed on σ-frequent itemset supports by interval of support.

tive interestingness measures for association rules and are commonly defined as follows.

Definition 15 (support and confidence). Let $X \Rightarrow Y$ be an association rule based on the set of items R. The support and confidence of this rule in a database r over R are denoted by $Sup(r, X \Rightarrow Y)$ and $Conf(r, X \Rightarrow Y)$ and are defined respectively by $Sup(r, X \Rightarrow Y) = Sup(r, X \cup Y)$ and $Conf(r, X \Rightarrow Y) = Sup(r, X \cup Y)/Sup(r, X)$. The rule $X \Rightarrow Y$ is *frequent* in r w.r.t. a frequency threshold σ if $X \cup Y \in Freq(r, \sigma)$.

6.1 Error bounds for support approximation

The error on support of association rules is the same as the error on support of itemsets. For a frequent rule $X \Rightarrow Y$, if we use frequent δ-free-sets to determine its support, by Theorem 2 we always have an overestimate of its support with an error of at most $\delta|X \cup Y|$. In practice, we have the same approximation errors as those presented in Section 5.3.

6.2 Error bounds for confidence approximation

Let $X \Rightarrow Y$ be a frequent rule in r. Suppose we used frequent δ-free-sets to approximate $Sup(r, X \cup Y)$ and $Sup(r, X)$. These approximations are denoted respectively by $\overline{Sup}(r, X \cup Y)$ and $\overline{Sup}(r, X)$. Now, we can approximate $Conf(r, X \Rightarrow Y)$ by $\overline{Conf}(r, X \Rightarrow Y) = \overline{Sup}(r, X \cup Y)/\overline{Sup}(r, X)$. By Theorem 2, and since we have overestimated the supports, then $Sup(r, X \cup Y)/(Sup(r, X) + \delta|X|) \leq \overline{Conf}(r, X \Rightarrow Y) \leq (Sup(r, X \cup Y) + \delta|X \cup Y|)/Sup(r, X)$.
Thus a bound for the absolute error made on the confidence when we use $\overline{Conf}(r, X \Rightarrow Y)$ instead of $Conf(r, X \Rightarrow Y)$ is $max(Sup(r, X \cup Y)/(Sup(r, X) + \delta|X|) - Conf(r, X \Rightarrow Y), (Sup(r, X \cup Y) + \delta|X \cup Y|)/Sup(r, X) - Conf(r, X \Rightarrow Y))$.
Now, we derive values of this bound in practice, using the experiments reported in Section 5.3. We consider the PUMS data set of Kansas state, which is less favorable than the other (PUMSB*) since the error on the support was larger.
Let $ar(s, c)$ be the set of all association rules in this data set with support s and confidence c. For a given pair $\langle s, c \rangle$, we bound the error made on confidence for all rules in $ar(s, c)$ as follows. The support of the left hand side of any of these rules is $s' = s/c$. Using the experimental results of Section 5.3, we can find the maximal relative support error made on s and s', denoted respectively by rse and rse'. Then we bound the absolute error made on the confidence by $max(s/(s' + s' \times rse') - c, (s + s \times rse)/s' - c)$.

	support						
confidence	0.05	0.1	0.2	0.3	0.4	0.5	0.6
0.99	0.0308	0.0116	0.0047	0.0033	0.0012	0.0010	0.0010
0.95	0.0295	0.0111	0.0045	0.0031	0.0011	0.0010	0.0010
0.9	0.0280	0.0105	0.0042	0.0030	0.0011	0.0009	0.0009
0.85	0.0264	0.0099	0.0040	0.0028	0.0010	0.0008	0.0009

Table 7. Bounds for absolute error on rule confidence.

We consider support $s \in \{0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$ and confidence $c \in \{0.99, 0.95,\ 0.9, 0.85\}$. For each pair $\langle s, c \rangle$, we used the maximal relative error on support given in Table 6 to bound the error made on confidence for the set of rule $ar(s, c)$. The corresponding

values are presented in Table 7. For example, if we consider rules with confidence 0.99 and support 0.05, the maximal absolute error made on confidence is 0.0308. For higher rule supports the error decreases. This variation corresponds to the reduction of the maximal relative error for higher supports in Table 7. For lower confidence values the error also decreases. This is due to the fact that a lower confidence implies a higher support for the left hand side of the rule and thus a lower error on the left hand side support.

7 Related work

Using incomplete information about itemset frequencies for some mining task, e.g., Boolean rule mining, has been proposed in [10], and formalized in the general framework of ϵ-adequate representations. Probabilistic approaches to the problem of frequency queries have also been investigated (see [14]).

Several search space reductions based on nearly exact (or exact) association rules have been proposed. The use of the nearly exact association rules to estimate the confidence of other rules and then to prune the search space has been suggested in [4] but not investigated nor experimented. Efficient mining of nearly exact rules (more specifically rules with at most δ exceptions) with a single attribute in both the left and the right hand sides has been proposed in [9]. Search space pruning using exact association rules has been experimented in [4] in the context of rule mining and developed independently in the context of frequent itemset mining in [13]. [13] implicitly proposes a kind of condensed representation called *closed* itemsets which is strongly related to the notion of 0-free-sets (δ-free-sets with $\delta = 0$). Mining 0-free-sets or closed itemsets lead to similar gains, but mining δ-free-sets with $\delta \neq 0$ offers additional search space reductions (at the cost of an uncertainty on supports). It should also be noticed that by definition exact rules are very sensitive to noise. If we process a noisy data set (a very common case in practice) a few exceptions to the exact rules can appear easily. Then the pruning methods based on exact rules will be less effective, while the mining of δ-free-sets with $\delta \neq 0$ can still benefit of an important search space reduction.

The techniques mentioned in this section present important benefits on dense data sets, but if we consider very sparse data sets, we can

hardly expect to have many exact or nearly exact rules that hold, and thus all these techniques are likely to be less interesting. Moreover, on very sparse data sets, these techniques may be a little bit slower than the direct extraction of frequent itemsets without pruning, since they can not take advantage of important search space reductions, but have to pay for a little overhead due to the tests performed to detect the rules.

8 Conclusion and future work

We proposed a structure called free-sets that can be extracted efficiently, even on dense data sets, and that can be used to approximate closely the support of frequent itemsets. We formalized this approximation in the framework of ϵ-adequate representations [10] and gave a correct extraction algorithm formulated as an instance of the levelwise search algorithm presented in [11].
We reported experiments showing that frequent free-sets can be extracted even when the extraction of frequent itemsets turns out to be intractable. The experiments also show that the error made when approximating the support of frequent itemsets using the support of frequent free-sets remains very low in practice. Finally, we considered the effect of this approximation on the support and confidence of association rules. We bounded the corresponding errors and the experiments show that these bounds are very tight in practice.
Interesting future work includes applications of the notion of δ-free-set to the the approximation of the support of general Boolean formula as investigated in [10].

References

1. R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. of the 1993 ACM SIGMOD International Conference on Management of Data (SIGMOD'93)*, pages 207–216, Washington, D.C., 1993.
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. AAAI Press, 1996.
3. R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proc. of the 20th International Conference on Very Large Data Bases (VLDB'94)*, pages 487 – 499, Santiago de Chile, Chile, 1994.

4. R. J. Bayardo. Brute-force mining of high-confidence classification rules. In *Proc. of the Third International Conference on Knowledge Discovery and Data Mining (KDD'97)*, pages 123–126, Newport Beach, California, 1997.
5. R. J. Bayardo. Efficiently mining long patterns from databases. In *Proc. of the 1998 ACM SIGMOD International Conference on Management of Data (SIGMOD'98)*, pages 85–93, Seattle, Washington, 1998.
6. J.-F. Boulicaut and A. Bykowski. Frequent closures as a concise representation for binary data mining. In *Proc. of the Fourth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'00)*, volume 1805 of *LNAI*, pages 62–73, Kyoto, JP, 2000. Springer-Verlag.
7. J.-F. Boulicaut, A. Bykowski, and C. Rigotti. Approximation of frequency queries by mean of free-sets. In *Proc. of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'00)*, LNAI, Lyon, France, Sept. 2000. To appear. Springer-Verlag.
8. A. Bykowski and L. Gomez-Chantada. Frequent itemset extraction in highly-correlated data: a web usage mining application. In *Proc. of the 2000 International Workshop on Web Knowledge Discovery and Data Mining (WKDDM'00)*, pages 27–42, Kyoto, JP, Apr. 2000.
9. S. Fujiwara, J. D. Ullman, and R. Motwani. Dynamic miss-counting algorithms: Finding implication and similarity rules with confidence pruning. In *Proc. of the 16th International Conference on Data Engineering (ICDE'00)*, pages 501–511, San Diego, California, 2000.
10. H. Mannila and H. Toivonen. Multiple uses of frequent sets and condensed representations. In *Proc. of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96)*, pages 189–194, Portland, Oregon, 1996.
11. H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.
12. R. Ng, L. V. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimization of constrained association rules. In *Proc. of the 1998 ACM SIGMOD International Conference on Management of Data (SIGMOD'98)*, pages 13–24, Seattle, Washington, 1998.
13. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Efficient mining of association rules using closed itemset lattices. *Information Systems*, 24(1):25–46, 1999.
14. D. Pavlov, H. Mannila, and P. Smyth. Probalistic models for query approximation with large data sets. Technical Report 2000-07, Univsersity of California, Department of Information and Computer Science, Irvine, CA-92697-3425, Feb. 2000.
15. G. Piatetsky-Shapiro. Discovery, analysis, and presentation of strong rules. In *Knowledge Discovery in Databases*, pages 229 – 248. AAAI Press, Menlo Park, CA, 1991.