

Constraint-Based Discovery and Inductive Queries: Application to Association Rule Mining

Baptiste Jeudy and Jean-François Boulicaut

Institut National des Sciences Appliquées de Lyon
Laboratoire d'Ingénierie des Systèmes d'Information
Bâtiment Blaise Pascal

F-69621 Villeurbanne cedex, France

{Baptiste.Jeudy, Jean-Francois.Boulicaut}@lisi.insa-lyon.fr

Abstract. Recently inductive databases (IDBs) have been proposed to afford the problem of knowledge discovery from huge databases. Querying these databases needs for primitives to: (1) select, manipulate and query data, (2) select, manipulate and query “interesting” patterns (i.e., those patterns that satisfy certain constraints), and (3) cross over patterns and data (e.g., selecting the data in which some patterns hold). Designing such query languages is a long-term goal and only preliminary approaches have been studied, mainly for the association rule mining task. Starting from a discussion on the MINE RULE operator, we identify several open issues for the design of inductive databases dedicated to these descriptive rules. These issues concern not only the offered primitives but also the availability of efficient evaluation schemes. We emphasize the need for primitives that work on more or less condensed representations for the frequent itemsets, e.g., the (frequent) δ -free and closed itemsets. It is useful not only for optimizing single association rule mining queries but also for sophisticated post-processing and interactive rule mining.

1 Introduction

In the cInQ¹ project, we want to develop a new generation of databases, called “*inductive databases*” (IDBs), suggested by Imielinski and Mannila in [15] and formalized in, e.g., [10]. This kind of databases integrate *raw data* with *knowledge* extracted from *raw data*, materialized under the form of patterns into a common framework that supports the knowledge discovery process within a database framework. In this way, the process of KDD consists essentially in a querying process, enabled by an ad-hoc, powerful and universal query language that can deal either with raw data or patterns and that can be used throughout the whole KDD process across many different applications. We are far from an understanding of fundamental primitives for such query languages when considering various kinds of knowledge discovery processes. The so-called *association rule mining*

¹ This research is part of the cInQ project (IST 2000-26469) that is partially funded by the European Commission IST Programme – Future and Emergent Technologies.

process introduced in [1] has received a lot of attention these last five years and it provides an interesting context for studying the inductive database framework and the identification of promising concepts. Indeed, when considering this kind of local pattern, a few query languages can be considered as candidates like the MINE RULE operator [23], MSQL [16], and DMQL [14] (see also [5] for a critical evaluation of these three languages).

A *query language* for IDBs, is an extension of a query language that includes primitives for supporting every step of the mining process. When considering association rule mining, it means that the language enables to specify:

- The selection of data to be mined. It must offer the possibility to select (e.g., via standard queries but also by means of sampling), to manipulate and to query data and views in the database. Also, primitives that support typical preprocessing like quantitative value discretization are needed.
- The specification of the type of rules to be mined. It often concerns syntactic constraints on the desired rules (e.g., the size of the body) but also the specification of the sorts of the involved attributes.
- The specification of the needed background knowledge (e.g., the definition of a concept hierarchy).
- The definition of constraints that the extracted patterns must satisfy. Among others, this implies that the language allows the user to define constraints that specify the interestingness (e.g., using measures like frequency, confidence, etc) on the patterns to be mined.
- The post-processing of the generated results. The language must allow to browse the collection of patterns, apply selection templates, *cross over* patterns and data, e.g., by selecting the data in which some patterns hold, or to manipulate results with some aggregate functions.

The satisfaction of a *closure property*, i.e., the user queries an inductive database instance and the result is again an inductive database instance is crucial for supporting the dynamic aspect of a discovery process and its inherent interactivity. This closure property can be achieved by means of the storage of the extracted patterns in the same database.

Contribution. Relating the inductive database framework with constraint-based mining enables to widen the scope of interest of this framework to various contributions in the data mining and machine learning communities. Then, we consider that it is useful to emphasize the interest of several condensed representations for frequent patterns that have been studied the last three years. However, algorithms for mining these representations have been already published and will not be discussed here. Only a few important conceptual issues like regeneration or the need for constrained condensed representations are discussed. Last, we sketch why these concepts are useful not only for the optimization of a single association rule mining query but also for sophisticated rule post-processing and interactive association rule mining.

Organization of the paper. First, we discuss the MINE RULE proposal to identify several open issues for the design of inductive databases dedicated to association rule mining (Section 2). It concerns not only the offered primitives but also the availability of efficient evaluation schemes. In Section 3, we provide a formalization of the association rule mining task and the needed notations. Then, in Section 4, we emphasize the need for primitives that work on more or less condensed representations for the frequent itemsets, e.g., the (frequent) δ -free and closed itemsets. In Section 5, the use of these condensed representations for both the optimization of inductive queries and sophisticated rule post-processing is discussed.

2 The MINE RULE Operator [23]

Throughout this paper, we use the MINE RULE query example of Figure 1 on the relational database of Table 1. The database records several transactions made by three customers in a store on different dates. The result of such a query is a set of frequent and valid association rules. A rule like **Coffee Boots** \Rightarrow **Darts** is frequent if enough customers buy within a same transaction **Coffee**, **Boots** and **Darts**. This rule is said valid if a customer who buys **Coffee** and **Boots** tends to buy **Darts** either.

Association rules are mined from a so-called *transactional database* that must be specified within the query. The **FROM** clause of the query specifies which part of the relational database (using any valid SQL query) is considered to construct the transactional database (e.g., given the used **WHERE** clause, only the transactions done after Nov. 8 are used). The **GROUP BY** clause specifies that the rows of the **purchase** table are grouped by transactions to form the rows of the transactional database (e.g., another choice would have been to group the rows by customers). In our query example, the result of this grouping/selection step is the transactional database T of Figure 2.

The specified transactional database is used to perform association rule mining under constraints. The **SELECT** clause specifies that the body and head of the rules are **products** (a rule has the form **body** \Rightarrow **head** where **body** and **head** are sets of **products**) and that their size is greater than one (with no upper bound). This query also defines the constraints that must be fulfilled by the rules. The rules must be frequent (with a frequency threshold of 0.5), valid (with a confidence threshold of 0.7), and must satisfy the other constraints expressed in the **SELECT** clause: $\mathcal{C}_a(X \Rightarrow Y) \equiv \forall A \in Y, A.price > 100$ and $\mathcal{C}_b(X \Rightarrow Y) \equiv |(X \cup Y) \cap \{\mathbf{Album}, \mathbf{Boots}\}| \leq 1$. \mathcal{C}_a means that all products in the head of the rule must have a price greater than 100 and \mathcal{C}_b means that the rule must contain at most one product out of $\{\mathbf{Album}, \mathbf{Boots}\}$. Finally, the answer to this query is the set of rules that satisfy $\mathcal{C}_{\text{freq}} \wedge \mathcal{C}_{\text{conf}} \wedge \mathcal{C}_a \wedge \mathcal{C}_b$ on the transactional database T of Figure 2.

Let us now review some of the open problems with the MINE RULE proposal.

- *Data selection and preprocessing.* Indeed, query languages based on SQL enable to use the full power of this standard query language for data selection.

Table 1. Part of the purchase table.

tr.	cust.	product	date	price
1	cust ₁	Coffee	Nov. 8	20
1	cust ₁	Darts	Nov. 8	50
2	cust ₂	Album	Nov. 9	110
2	cust ₂	Boots	Nov. 9	120
2	cust ₂	Coffee	Nov. 9	20
2	cust ₂	Darts	Nov. 9	50
3	cust ₁	Boots	Nov. 9	120
3	cust ₁	Coffee	Nov. 9	20
4	cust ₃	Album	Nov. 10	110
4	cust ₃	Coffee	Nov. 10	20
⋮	⋮	⋮	⋮	⋮

```

MINE RULE result AS
SELECT DISTINCT 1..n product AS BODY,
    1..n product AS HEAD, SUPPORT, CONFIDENCE
WHERE HEAD.price > 100 AND
    |(HEAD ∪ BODY) ∩ {Album, Boots}| ≤ 1
FROM purchase WHERE date > Nov. 8
GROUP BY transaction
EXTRACTING RULES WITH SUPPORT: 0.5, CONFIDENCE: 0.7

```

Fig. 1. A MINE RULE query on the purchase database.

It is out of the scope of this paper to discuss this phase but it is interesting to note that MINE RULE offers no specific primitive for data preprocessing (e.g., discretization) and that the other languages like MSOL offer just a few [16]. Preprocessing remains ad-hoc for many data mining processes and it is often assumed that it is performed beforehand by means of various software tools.

- The specification of the type of rules to be mined is defined in MINE RULE by the SELECT clause. It enables the definition of simple syntactic constraints, the specification of the sorts of attributes, and the definition of the so-called *mining conditions* that can make use of some background knowledge. Using MINE RULE, it is assumed that this knowledge has been encoded within the relational database.
- In MINE RULE, it is possible to define minimal frequency and minimal confidence for the desired rules.
- *Rule post-processing.* When using MINE RULE, no specific post-processing primitive is offered. This contrasts with the obvious needs for pattern post-processing in unsupervised data mining processes like association rule mining. Indeed, extracted rules can be stored under a relational schema and then be queried by means of SQL. However, it has been shown (see, e.g., [5]) that simple post-processing queries are then quite difficult to express.

To the best of our knowledge, in the MINE RULE architecture, the collection of frequent itemsets and their frequencies is not directly available for further use. It means that the computation of other interestingness measures like the J-measure [29] is not possible without looking again at the data. For rule post-processing, MSQL is richer than the other languages in its offer of few built-in post-processing primitives (it reserves a dedicated operator, `SelectRules` for these purposes and primitives for crossing over the rules to the data). However, none of the proposed languages supports complex post-processing processes (e.g., the computation of non redundant rules) as needed in real-life association rule mining.

It is useful to abstract the meaning of such mining queries. A simple model has been introduced in [22] that considers a data mining process as a sequence of queries over the data but also the *theory* of the data. Given a language \mathcal{L} of patterns (e.g., association rules), the theory of a database \mathbf{r} with respect to \mathcal{L} and a selection predicate q is the set $Th(\mathbf{r}, \mathcal{L}, q) = \{\phi \in \mathcal{L} \mid q(\mathbf{r}, \phi) \text{ is true}\}$. The predicate q indicates whether a pattern ϕ is considered interesting (e.g., q specifies that ϕ is “frequent” in \mathbf{r}). The selection predicate can be defined as a combination (boolean expression) of atomic constraints that have to be satisfied by the patterns. Some of them refer to the “behavior” of a pattern in the data (e.g., its “frequency” in \mathbf{r} is above a threshold), some others define syntactical restrictions on the desired patterns (e.g., its “length” is below a threshold). Preprocessing defines \mathbf{r} , the mining phase is often the computation of the specified theory while post-processing can be considered as a querying activity on a materialized theory or the computation of a new theory. A “generate and test” approach that would enumerate the sentences of \mathcal{L} and then test the selection predicate q is generally impossible. A huge effort has concerned the “active” use of the primitive constraints occurring in q to have a tractable evaluation of useful mining queries.

Indeed, given the (restricted) collection of primitives offered by the MINE RULE operator, it is possible to have an efficient implementation thanks to the availability of efficient algorithms for computing frequent itemsets from huge but sparse databases [2,26]. To further extend both the efficiency of single query evaluation (especially in the difficult contexts where expressive constraints are used and/or the data are highly-correlated), the offered primitives for post-processing and the optimization of sequence of queries, we now consider an abstract framework in which the impact of the so-called *condensed representations* of frequent patterns can be emphasized.

3 Association Rule Mining Queries

Assume that `Items` is a finite set of symbols denoted by capital letters, e.g., `Items = {A, B, C, ...}`. A *transactional database* is a collection of rows where each row is a subset of `Items`. An *itemset* is a subset of `Items`. A row r *supports* an itemset S if $S \subseteq r$. The *support* (denoted $\text{support}(S)$) of an itemset S is

the multi-set of all rows of the database that support S . The *frequency* of an itemset S is $|\text{support}(S)|/|\text{support}(\emptyset)|$ and is denoted $\mathcal{F}(S)$. Figure 2 provides an example of a transactional database and the supports and the frequencies of some itemsets. We often use a string notation for itemsets, e.g., AB for $\{A, B\}$.

An association rule is denoted $X \Rightarrow Y$ where $X \cap Y = \emptyset$ and $X \subseteq \text{Items}$ is the *body* of the rule and $Y \subseteq \text{Items}$ is the *head* of the rule. The *support* and *frequency* of a rule are defined as the support and frequency of the itemset $X \cup Y$. A row r *supports* a rule $X \Rightarrow Y$ if it supports $X \cup Y$. A row r is an *exception* for a rule $X \Rightarrow Y$ if it supports X and it does not support Y . The *confidence* of the rule is $\mathcal{CF}(X \Rightarrow Y) = \mathcal{F}(X \Rightarrow Y)/\mathcal{F}(X) = \mathcal{F}(X \cup Y)/\mathcal{F}(X)$. The confidence of the rule gives the conditional probability that a row supports $X \cup Y$ when it supports X . A rule with a confidence of one has no exception and is called a *logical rule*. Frequency and confidence are two popular *evaluation functions* for association rules. Interestingly, the association rule mining task is not well specified as a theory computation. Indeed, in this kind of process, we need not only the patterns that satisfy q but also the results of some evaluation functions for each of these selected patterns.

We now define constraints on itemsets and rules.

Definition 1 (constraint). *If \mathcal{B} denotes the set of all transactional databases and 2^{Items} the set of all itemsets, an itemset constraint \mathcal{C} is a predicate over $2^{\text{Items}} \times \mathcal{B}$. Similarly, a rule constraint is a predicate over $\mathcal{R} \times \mathcal{B}$ where \mathcal{R} is the set of association rules. An itemset $S \in 2^{\text{Items}}$ (resp. a rule R) satisfies a constraint \mathcal{C} in the database $B \in \mathcal{B}$ iff $\mathcal{C}(S, B) = \text{true}$ (resp. $\mathcal{C}(R, B) = \text{true}$). When it is clear from the context, we write $\mathcal{C}(S)$ (resp. $\mathcal{C}(R)$). Given a subset I of Items , we define $\text{SAT}_{\mathcal{C}}(I) = \{S \in I, S \text{ satisfies } \mathcal{C}\}$ for an itemset constraint (resp. if J is a subset of \mathcal{R} , $\text{SAT}_{\mathcal{C}}(J) = \{R \in J, R \text{ satisfies } \mathcal{C}\}$ for a rule constraint). $\text{SAT}_{\mathcal{C}}$ denotes $\text{SAT}_{\mathcal{C}}(2^{\text{Items}})$ or $\text{SAT}_{\mathcal{C}}(\mathcal{R})$.*

We can now define the frequency constraint for itemsets and the frequency and confidence constraints for association rules. $\mathcal{C}_{\text{freq}}(S) \equiv \mathcal{F}(S) \geq \gamma$, $\mathcal{C}_{\text{freq}}(X \Rightarrow Y) \equiv \mathcal{F}(X \Rightarrow Y) \geq \gamma$, $\mathcal{C}_{\text{conf}}(X \Rightarrow Y) \equiv \mathcal{CF}(X \Rightarrow Y) \geq \theta$ where γ is the frequency threshold and θ the confidence threshold. A rule that satisfies $\mathcal{C}_{\text{freq}}$ is said frequent. A rule that satisfies $\mathcal{C}_{\text{conf}}$ is said valid.

Example 1 *Consider the dataset of Figure 2 where $\text{Items} = \{A, B, C, D\}$. If the frequency threshold is 0.5, then with the constraint $\mathcal{C}_b(S) \equiv |S \cap \{A, B\}| \leq 1$, $\text{SAT}_{\mathcal{C}_{\text{freq}} \wedge \mathcal{C}_b} = \{\emptyset, A, B, C, AC, BC\}$ If the confidence threshold is 0.7, then the rules satisfying the constraint of Figure 1 are $\text{SAT}_{\mathcal{C}_a \wedge \mathcal{C}_b \wedge \mathcal{C}_{\text{freq}} \wedge \mathcal{C}_{\text{conf}}} = \{\emptyset \Rightarrow A, C \Rightarrow A\}$.*

Let us now formalize that inductive queries that return itemsets or rules must also provide the results of the evaluation functions for further use.

Definition 2 (itemset query). *A itemset query is a pair $(\mathcal{C}, \mathbf{r})$ where \mathbf{r} is a transactional database and \mathcal{C} is an itemset constraint. The result of a query $Q = (\mathcal{C}, \mathbf{r})$ is defined as the set $\text{Res}(Q) = \{(S, \mathcal{F}(S)) \mid S \in 2^{\text{Items}} \wedge \mathcal{C}(S) = \text{true}\}$.*

$T =$	t_2	ABCD	Itemset	Support	Frequency
	t_3	BC	A	$\{t_2, t_4, t_5, t_6, t_7\}$	0.83
	t_4	AC	B	$\{t_2, t_3, t_6, t_7\}$	0.67
	t_5	AC	AB	$\{t_2, t_6, t_7\}$	0.5
	t_6	ABCD	AC	$\{t_2, t_4, t_5, t_6, t_7\}$	0.83
	t_7	ABC	CD	$\{t_2, t_6\}$	0.33
			ACD	$\{t_2, t_6\}$	0.33

Fig. 2. Supports and frequencies of some itemsets in a transactional database. This database is constructed during the evaluation of the MINE RULE query of Figure 1 from the purchase table of Table 1

Definition 3 (association rule query). *An association rule query is a pair $(\mathcal{C}, \mathbf{r})$ where \mathbf{r} is a transactional database and \mathcal{C} is an association rule constraint. The result of a query $Q = (\mathcal{C}, \mathbf{r})$ is defined as the set $Res(Q) = \{(R, \mathcal{F}(R), \mathcal{CF}(R)) \mid R \in \mathcal{R} \wedge \mathcal{C}(R) = true\}$.*

The classical association rule mining problem can be stated in an association rule query where the constraint part is the conjunction of the frequency and confidence constraint [1]. Our framework enables more complex queries and does not require that the frequency and/or frequency constraints appear in \mathcal{C} . However, if the constraint \mathcal{C} is not enough selective, the query will not be tractable. Selectivity can not be predicted beforehand. Fortunately, when the constraint has some nice properties, e.g., it is a conjunction of anti-monotone and monotone atomic constraints, efficient evaluation strategies have been identified (see the end of this section).

Our definition of an association rule query can also be modified to include other quality measures (e.g., the J-measure [29]) and not only the frequency and the confidence.

Computing the result of the classical association rule mining problem is generally done in two steps [2]: first the computation of all the frequent itemsets and their frequency and then the computation of every valid association rule that can be made from disjoint subsets of each frequent itemset. This second step is far less expensive than the first one because no access to the database is needed: only the collection of the frequent itemsets and their frequencies are needed.

To compute the result of an arbitrary association rule query, the same strategy can be used. First, derive an itemset query from the association rule query, then compute the result of this query using the transactional database and finally generate the association rules from the itemsets. For the first step, there is no general method. This is generally done in an ad-hoc manner (see Example 2) and supporting this remains an open problem. The generation of the rules can be performed by the following algorithm:

Algorithm 1 (Rule_Gen) *Given an association rule query $(\mathbf{r}, \mathcal{C})$ and the result Res of the itemset query, do:*

For each pair $(S, \mathcal{F}(S)) \in Res$ and for each $T \subset S$
 Construct the rule $R := T \Rightarrow (S - T)$
 Compute $\mathcal{F}(R) := \mathcal{F}(S)$ and $\mathcal{CF}(R) := \mathcal{F}(S)/\mathcal{F}(T)$.
 Output $(R, \mathcal{F}(R), \mathcal{CF}(R))$ if it satisfies the rule constraint \mathcal{C} .

Since the database is used only during the computation of itemsets, the generation of rules is efficient.

Example 2 The constraint used in the query of Figure 1 is: $\mathcal{C}_{ar}(X \Rightarrow Y) = \mathcal{C}_{freq} \wedge \mathcal{C}_{conf} \wedge \mathcal{C}_a(X \Rightarrow Y) \wedge \mathcal{C}_b(X \Rightarrow Y)$ where $\mathcal{C}_a(X \Rightarrow Y) \equiv \forall A \in Y, A.price > 100$ and $\mathcal{C}_b(X \Rightarrow Y) \equiv |(X \cup Y) \cap \{\text{Album, Boots}\}| \leq 1$. \mathcal{C}_b can be rewritten as an itemset constraint: $\mathcal{C}_b(S) \equiv |S \cap \{\text{Album, Boots}\}| \leq 1$. Furthermore, since (as specified in the MINE RULE query) rules cannot have an empty head, $\mathcal{C}_a(X \Rightarrow Y) \equiv \forall A \in Y, A.price > 100 \wedge \mathcal{C}'_a(X \cup Y)$ where $\mathcal{C}'_a(S) \equiv |S \cap \{I \in \text{Items}, I.price > 100\}| \geq 1$ is a useful itemset constraint.

Finally, we can derive an itemset query $Q_i = (\mathcal{C}_i, \mathbf{r})$ with the constraint $\mathcal{C}_i = \mathcal{C}_{freq} \wedge \mathcal{C}_b \wedge \mathcal{C}'_a$ and be sure that the result of this itemset query will allow the generation of the result of the association rule query $Q = (\mathcal{C}_{ar}, \mathbf{r})$ using Algorithm 1.

The efficiency of the extraction of the answer to the itemset query relies on the possibility to use constraints during the itemset computation. A classical result is that effective safe pruning can be achieved when considering anti-monotone constraints [22,26]. It relies on the fact that if an itemset violates an anti-monotone constraint then all its supersets violate it as well and therefore this itemset and its supersets can be pruned.

Definition 4 (Anti-monotonicity). An anti-monotone constraint is a constraint \mathcal{C} such that for all itemsets S, S' : $(S' \subseteq S \wedge \mathcal{C}(S)) \Rightarrow \mathcal{C}(S')$.

The prototypical anti-monotone constraint is the frequency constraint. The constraint \mathcal{C}_b of Example 2 is another anti-monotone constraint and many other examples can be found, e.g., in [26]. Notice that the conjunction or disjunction of anti-monotone constraints is anti-monotone.

The monotone constraints can also be used to improve the efficiency of itemset extraction (optimization of the candidate generation phase that prevents to consider candidates that do not satisfy the monotone constraint) [17]. However, pushing monotone constraints sometimes increases the computation times since it prevents effective pruning based on anti-monotone constraints [30,9,12].

Definition 5 (Monotonicity). A monotone constraint is a constraint \mathcal{C} such that for all itemsets S, S' : $(S \subseteq S' \wedge S \text{ satisfies } \mathcal{C}) \Rightarrow S' \text{ satisfies } \mathcal{C}$.

Example 3 \mathcal{C}'_a (see Example 2), $\mathcal{C}(S) \equiv \{\text{A, B, C, D}\} \subset S$, $\mathcal{C}(S) \equiv \text{Sum}(S.price) > 100$ (the sum of the prices of items from S is greater than 100) and $\mathcal{C}(S) \equiv S \cap \{\text{A, B, C}\} \neq \emptyset$ are examples of monotone constraints.

4 Condensed Representations of Frequent Sets

To answer an association rule query, we must be able to provide efficiently the frequency of many itemsets (see Algorithm 1). Computing the frequent itemsets is a first solution. Another one is to use condensed representations with respect to frequency queries. Condensed representation is a general concept (see, e.g., [21]). In our context, the intuition is to substitute to the database or the collection of the frequent itemsets, another representation from which we can derive the whole collection of the frequent itemsets and their frequencies. In this paper, given a set \mathcal{S} of pairs $(X, \mathcal{F}(X))$, we are interested in condensed representations of \mathcal{S} that are subsets of \mathcal{S} with two properties: (1) It is much smaller than \mathcal{S} and faster to compute, and (2), the whole set \mathcal{S} can be generated from the condensed representation with no access to the database, i.e., efficiently. User-defined constraints can also be used to further optimize the computation of condensed representations [17].

Several algorithms exist to compute various condensed representations of frequent itemsets: CLOSE [27], CLOSET [28], CHARM [31], MIN-EX [6], or PASCAL [4]. These algorithms compute different condensed representations: the frequent closed itemsets (CLOSE, CLOSET, CHARM), the frequent free itemsets (MIN-EX, PASCAL), or the frequent δ -free itemsets for MIN-EX. Also, a new promising condensed representation, the disjoint-free itemsets, has been proposed in [11]. These algorithms enable tractable extractions from dense and highly-correlated data, i.e., extractions for frequency thresholds on which APRIORI-like algorithms are intractable. Let us now discuss two representations on which we have been working: the closed itemsets and the δ -free itemsets.

Definition 6 (closures and closed itemsets). *The closure of an itemset S (denoted by $\text{closure}(S)$) is the maximal (for set inclusion) superset of S which has the same support than S . A closed itemset is an itemset that is equal to its closure.*

The next proposition shows how to compute the frequency of an itemset using the collection of the frequent closed itemsets efficiently, i.e., with no access to the database.

Proposition 1 *Given an itemset S and the set of frequent closed itemsets,*

- *If S is not included in a frequent closed itemset then S is not frequent.*
- *Else S is frequent and $\mathcal{F}(S) = \text{Max}\{\mathcal{F}(X), S \subseteq X \text{ and } X \text{ is closed}\}$.*

Using this proposition, it is possible to design an algorithm to compute the result of a frequent itemset query using the frequent closed itemsets. This algorithm is not given here (see, e.g., [27,4]). As a result, γ -frequent closed itemsets are like the γ -frequent itemsets a $\gamma/2$ -adequate representation for frequency queries [6], i.e., the error on the exact frequency for any itemset is bound by $\gamma/2$ (the $\gamma/2$ value is given to infrequent itemsets and the frequency of any frequent itemset is known exactly).

Example 4 *In the transactional database of Figure 2, if the frequency threshold is 0.2, every itemset is frequent (16 frequent itemsets). The frequent closed sets are C, AC, BC, ABC, and ABCD and we can use the previous property to get the frequency of non-closed itemsets from closed ones (e.g., $\mathcal{F}(\text{AB}) = \mathcal{F}(\text{ABC})$ since ABC is the smallest closed superset of AB).*

We can compute the closed sets from the free sets.

Definition 7 (free itemset). *An itemset S is free if no logical rule holds between its items, i.e., it does not exist two distinct itemsets X, Y such that $S = X \cup Y$, $Y \neq \emptyset$ and $X \Rightarrow Y$ is a logical rule.*

Example 5 *In the transactional database of Figure 2, if the frequency threshold is 0.2, the frequent free sets are \emptyset, A, B, D , and AB.*

The closed sets are the closure of the free one. Freeness is an anti-monotone property and thus can be used efficiently, e.g., within a level-wise algorithm.

When they can be computed, closed itemsets constitute a good condensed representation (see, e.g., [6] for experiments with real-life dense and correlated data sets). The free sets can be generalized to δ -free itemsets². Representations based on δ -free itemsets are quite interesting when it is not possible to mine the closed sets, i.e., when the computation is intractable given the user-defined frequency threshold. Indeed, algorithms like CLOSE [27] or PASCAL [4] use logical rule to prune candidate itemsets because their frequencies can be inferred from the frequencies of free/closed itemsets. However, to be efficient, these algorithms need that such logical rules hold in the data. If it is not the case, then the frequent free sets are exactly the frequent itemsets and we get no improvement over APRIORI-like algorithms.

The MIN-EX algorithm introduced in [6,8] computes δ -free itemsets. The concept of closure is extended, providing new possibilities for pruning. However, we must trade this efficiency improvement against precision: the frequency of the frequent itemsets are only known within a bounded error. The MIN-EX algorithm uses rules with few exceptions to further prune the search space. Given an itemset $S = X \cup Y$ and a rule $Y \Rightarrow Z$ with less than δ exceptions, then the frequency of $X \cup Y \cup Z$ can be approximated by the frequency of S . More formally, MIN-EX uses an extended notion of closure.

Definition 8 (δ -closure and δ -free itemsets). *Let δ be an integer and S an itemset. The δ -closure of S , $\text{closure}_\delta(S)$ is the maximal (w.r.t. the set inclusion) superset Y of S such that for every item $A \in Y - S$, $|\text{Support}(S \cup \{A\})|$ is at least $|\text{Support}(S)| - \delta$. An itemset S is δ -free if no association rule with less than δ exceptions holds between its subsets.*

Example 6 *In the transactional database of Figure 2, if the frequency threshold is 0.2 and $\delta = 1$, the frequent 1-free sets are \emptyset, A, B , and D.*

² There is no such generalization for closed sets

Notice that with $\delta = 0$, it is the same closure operator than for CLOSE, i.e., $\text{closure}_0 = \text{closure}$. Larger values of δ leads to more efficient pruning (there are less δ -free itemsets) but also larger errors on the frequencies of itemsets when they are regenerated from the δ -free ones (see below).

The output of the MIN-EX algorithm is formally given by the three following sets: $FF(\mathbf{r}, \gamma, \delta)$ is the set of the γ -frequent δ -free itemsets, $IF(\mathbf{r}, \gamma, \delta)$ is the set of the minimal (w.r.t. the set inclusion) infrequent δ -free itemsets (i.e., the infrequent δ -free itemsets whose all subsets are γ -frequent). $FN(\mathbf{r}, \gamma, \delta)$ is the set of the minimal γ -frequent non- δ -free itemsets (i.e., the γ -frequent non- δ -free itemsets whose all subsets are δ -free). The two pairs (FF, IF) and (FF, FN) are two condensed representations based on δ -free itemsets. The next proposition shows that it is possible to compute an approximation of the frequency of an itemset using one of these two condensed representations.

Proposition 2 *Let S be an itemset. If there exists $X \in IF(\mathbf{r}, \gamma, \delta)$ such that $X \subseteq S$ then S is infrequent. If $S \notin FF(\mathbf{r}, \gamma, \delta)$ and there does not exist $X \in FN(\mathbf{r}, \gamma, \delta)$ such that $X \subseteq S$ then S is infrequent. In these two cases, the frequency of S can be approximated by $\gamma/2$. Else, let F be the δ -free itemset such that: $\mathcal{F}(F) = \text{Min}\{\mathcal{F}(X), X \subseteq S \text{ and } X \text{ is } \delta\text{-free}\}$. Assuming that $n_S = |\text{support}(S)|$ and $n_F = |\text{support}(F)|$, then $n_F \geq n_S \geq n_F - \delta(|S| - |F|)$, or, dividing this by n , the number of rows in \mathbf{r} , $\mathcal{F}(F) \geq \mathcal{F}(S) \geq \mathcal{F}(F) - \frac{\delta}{n}(|S| - |F|)$.*

Typical δ values range from zero to a few hundreds. With a database size of several tens of thousands of rows, the error made is below few percents [8].

Using Proposition 2, it is also possible to regenerate an approximation of the answer to a frequent itemset query from one of the condensed representation (FF, IF) or (FF, FN) :

- The frequency of an itemset is approximated with an error bound given by Proposition 2 (notice that this error is computed during the regeneration and thus can be presented to the user with the frequency of each itemset).
- Some of the computed itemsets might be infrequent because the uncertainty on their frequencies does not enable to classify them as frequent or infrequent (e.g., if $\gamma = 0.5$ and the $\mathcal{F}(X) = 0.49$ with an error of 0.02).

If $\delta = 0$, then the two condensed representations enable to regenerate exactly the answer to a frequent itemset query.

Given an arbitrary itemset query $Q = (\mathcal{C}, \mathbf{r})$, there are therefore two solutions to compute its answer:

- Pushing the anti-monotone and monotone part of the constraint \mathcal{C} as sketched in Section 3.
- Using condensed representation to answer a more general query (with only the frequency constraint) and then filter the itemsets that do not verify the constraint \mathcal{C} .

We now consider how to combine these two methods for mining condensed representations that satisfy a conjunction of an anti-monotone and a monotone

constraint. In [17], we presented an algorithm to perform this extraction. This algorithm uses an extension of the δ -free itemsets, the *contextual δ -free itemsets*.

Definition 9 (contextual δ -free itemset). *An itemset S is contextual δ -free with respect to a monotone constraint \mathcal{C}_m if it does not exist two distinct subsets X, Y of S such that X satisfies \mathcal{C}_m and $X \Rightarrow Y$ has less than δ exceptions.*

The input and output of this algorithm are formalized as follows:

Input: a query $Q = (\mathcal{C}_{am} \wedge \mathcal{C}_m, \mathbf{r})$ where \mathcal{C}_{am} is an anti-monotone constraint and \mathcal{C}_m a monotone constraint.

Output: three collections FF, IF, FN and, if $\delta = 0$, the collection \mathcal{O} .

- $FF = \{(S, \mathcal{F}(S)) \mid S \text{ is contextual } \delta\text{-free and } \mathcal{C}_{am}(S) \wedge \mathcal{C}_m(S) \text{ is true}\}$,
- IF and FN are defined as for MIN-EX and
- $\mathcal{O} = \{(\text{closure}(S), \mathcal{F}(S)) \mid S \text{ is free and } \mathcal{C}_{\text{freq}}(S) \wedge \mathcal{C}_{am}(S) \text{ is true}\}$.

These collections give three condensed representations \mathcal{O} (if $\delta = 0$), (FF, IF) and (FF, FN) . The regeneration of the answer to the query Q using the collection \mathcal{O} of closed itemsets can be done by:

Given an itemset S

If $\mathcal{C}_m(S)$ is true, then use Proposition 1 to compute $\mathcal{F}(S)$

If $\mathcal{C}_{am}(S)$ is true then output $(S, \mathcal{F}(S))$.

When considering (FF, IF) or (FF, FN) :

If $\mathcal{C}_m(S)$ is true, then use (FF, IF) or (FF, FN) as in Proposition 2 to compute $\mathcal{F}(S)$.

If $\mathcal{C}_{am}(S)$ is true or unknown then output $(S, \mathcal{F}(S))$

The result of $\mathcal{C}_{am}(S)$ can be unknown due to the uncertainty on the frequency (if $\delta \neq 0$).

5 Uses of Condensed Representations

Let us now sketch several applications of such condensed representations.

Optimization of MINE RULE queries. It is clear that the given condensed representations of the frequent patterns can be used, in a transparent way for the end-user, for optimization purposes. In such a context, we just have to replace the algorithmic core that concerns frequent itemset mining by our algorithms that compute free/closed itemsets and then derive the whole collection of the frequent itemsets. Also, the optimized way to push conjunction of monotone and anti-monotone constraints might be implemented.

Condensed representations have other interesting applications beyond the optimization of an association rule mining query.

Generation of non-redundant association rules. One of the problems in association rule mining from real-life data is the huge number of extracted rules. However, many of the rules are in some sense redundant and might be useless, e.g., $AB \Rightarrow C$ is not interesting if $A \Rightarrow BC$ has the same confidence. In [3], an algorithm is presented to extract a minimal cover of the set of frequent association rules. This set is generated from the closed and free itemsets. This cover can be generated by considering only rules of the form $X \Rightarrow (Y - X)$ where X is a free itemset and Y is a closed itemset containing X . It leads to a much smaller collection of association rules than the one computed from itemsets using Algorithm 1. In this volume, [19] considers other concise representations of association rules. In our practice, post-processing the discovered rules can clearly make use of the properties of the free and close sets. In other terms, materializing these collections can be useful for post-processing, not only the optimization of the mining phase. For instance, it makes sense to look for association rules that contain free itemsets as their left-hand sides and some targeted attributes on the right-hand sides without any minimal frequency constraint. It might remain tractable, thanks to the anti-monotonicity of freeness (extraction of the whole collection of the free itemsets), and need a reasonable amount of computation when computing the frequency and the confidence of each candidate rule.

Using condensed representation as a knowledge cache. A user generally submits a query, gets the results and refines it until he is satisfied by the extracted patterns. Since computing the result for one single query can be expensive (several minutes up to several hours), it is highly desirable that the data mining system optimizes sequences of queries. A classical solution is to cache the results of previous queries to answer faster to new queries. This has been studied by caching itemsets (e.g., [13,25]). Most of these works require that some strong relation holds between the queries like inclusion or equivalence. Caching condensed representations seems quite natural and we began to study the use of free itemsets for that purpose [18]. In [18], we assume the user defines constraints on closed sets and can refine them in a sequence of queries. Free sets from previous queries are put in a cache. A cache of free itemsets is much smaller than a cache containing itemsets and our algorithm ensures that the intersection between the union of the results of all previous queries and the result of the new query is not recomputed. Finally, we do not make any assumption on the relation between two queries in the sequence. The algorithm improves the performance of the extraction with respect to an algorithm that mines the closed sets without making use of the previous computations. The speedup is roughly equal to the relative size of the intersection between the answer to a new query and the content of the cache. Again, such an optimization could be integrated into the MINE RULE architecture in a transparent way.

6 Conclusion

Even though this paper has emphasized the use of frequent itemsets for association rule mining, the interest of inductive querying on itemsets goes far beyond

this popular mining task. For instance, constrained itemsets and their frequencies can be used for computing similarity measures between attributes and thus for clustering tasks (see, e.g., [24]). It can also be used for the discovery of more general kinds of rules, like rules with restricted forms of disjunctions or negations [21,7] and the approximation of the joint distribution [20]. Our future line of work will be, (1) to investigate the multiple uses of the condensed representations of frequent itemsets, and (2) to study evaluation strategies for association rule mining queries when we have complex selection criteria (i.e., general boolean expression instead of conjunctions of monotone and anti-monotone constraints).

Acknowledgments. The authors thank the researchers from the C_{IN}Q consortium for interesting discussions and more particularly, Luc De Raedt, Mika Klemettinen, Heikki Mannila, Rosa Meo, and Christophe Rigotti.

References

1. R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings SIGMOD'93*, pages 207–216, Washington, USA, 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. AAAI Press, 1996.
3. Y. Bastide, N. Pasquier, R. Taouil, G. Stumme, and L. Lakhal. Mining minimal non-redundant association rules using frequent closed itemsets. In *Proceedings CL 2000*, volume 1861 of *LNCS*, pages 972–986, London, UK, 2000. Springer-Verlag.
4. Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, and L. Lakhal. Mining frequent patterns with counting inference. *SIGKDD Explorations*, 2(2):66–75, Dec. 2000.
5. M. Botta, J.-F. Boulicaut, C. Masson, and R. Meo. A comparison between query languages for the extraction of association rules. In *Proceedings DaWaK'02*, Aix en Provence, F, 2002. Springer-Verlag. To appear.
6. J.-F. Boulicaut and A. Bykowski. Frequent closures as a concise representation for binary data mining. In *Proceedings PAKDD'00*, volume 1805 of *LNAI*, pages 62–73, Kyoto, JP, 2000. Springer-Verlag.
7. J.-F. Boulicaut, A. Bykowski, and B. Jeudy. Towards the tractable discovery of association rules with negations. In *Proceedings FQAS'00*, Advances in Soft Computing series, pages 425–434, Warsaw, PL, Oct. 2000. Springer-Verlag.
8. J.-F. Boulicaut, A. Bykowski, and C. Rigotti. Approximation of frequency queries by means of free-sets. In *Proceedings PKDD'00*, volume 1910 of *LNAI*, pages 75–85, Lyon, F, 2000. Springer-Verlag.
9. J.-F. Boulicaut and B. Jeudy. Using constraint for itemset mining: should we prune or not? In *Proceedings BDA'00*, pages 221–237, Blois, F, 2000.
10. J.-F. Boulicaut, M. Klemettinen, and H. Mannila. Modeling KDD processes within the inductive database framework. In *Proceedings DaWaK'99*, volume 1676 of *LNCS*, pages 293–302, Florence, I, 1999. Springer-Verlag.
11. A. Bykowski and C. Rigotti. A condensed representation to find frequent patterns. In *Proceedings PODS'01*, pages 267–273, Santa Barbara, USA, 2001. ACM Press.

12. M. M. Garofalakis, R. Rastogi, and K. Shim. SPIRIT: Sequential pattern mining with regular expression constraints. In *Proceedings VLDB'99*, pages 223–234, Edinburgh, UK, 1999. Morgan Kaufmann.
13. B. Goethals and J. van den Bussche. On implementing interactive association rule mining. In *Proceedings of the ACM SIGMOD Workshop DMKD'99*, Philadelphia, USA, 1999.
14. J. Han and M. Kamber. *Data Mining: Concepts and techniques*. Morgan Kaufmann Publishers, San Francisco, USA, 2000. 533 pages.
15. T. Imielinski and H. Mannila. A database perspective on knowledge discovery. *Communications of the ACM*, 39(11):58–64, 1996.
16. T. Imielinski and A. Virmani. MSQL: A query language for database mining. *Data Mining and Knowledge Discovery*, 3(4):373–408, 1999.
17. B. Jeudy and J.-F. Boulicaut. Optimization of association rule mining queries. *Intelligent Data Analysis*, 6(5), 2002. To appear.
18. B. Jeudy and J.-F. Boulicaut. Using condensed representations for interactive association rule mining. In *Proceedings PKDD'02*, Helsinki, FIN, 2002. Springer-Verlag. To appear.
19. M. Kryszkiewicz. Concise representations of association rules. In *Proceedings of the ESF Exploratory Workshop on Pattern Detection and Discovery*, London, UK, 2002. Springer-Verlag. To appear in this volume.
20. H. Mannila and P. Smyth. Approximate query answering with frequent sets and maximum entropy. In *Proceedings ICDE'00*, page 309, San Diego, USA, 2000. IEEE Computer Press.
21. H. Mannila and H. Toivonen. Multiple uses of frequent sets and condensed representations. In *Proceedings KDD'96*, pages 189–194, Portland, USA, 1996. AAAI Press.
22. H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.
23. R. Meo, G. Psaila, and S. Ceri. An extension to SQL for mining association rules. *Data Mining and Knowledge Discovery*, 2(2):195–224, 1998.
24. P. Moen. *Attribute, Event Sequence, and Event Type Similarity Notions for Data Mining*. PhD thesis, Department of Computer Science, P.O. Box 26, FIN-00014 University of Helsinki, Jan. 2000.
25. B. Nag, P. M. Deshpande, and D. J. DeWitt. Using a knowledge cache for interactive discovery of association rules. In *Proceedings SIGKDD'99*, pages 244–253. ACM Press, 1999.
26. R. Ng, L. V. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained associations rules. In *Proceedings SIGMOD'98*, pages 13–24, Seattle, USA, 1998. ACM Press.
27. 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.
28. J. Pei, J. Han, and R. Mao. CLOSET an efficient algorithm for mining frequent closed itemsets. In *Proceedings of the ACM SIGMOD Workshop DMKD'00*, pages 21–30, Dallas, USA, 2000.
29. P. Smyth and R. M. Goodman. An information theoretic approach to rule induction from databases. *IEEE Transactions on Knowledge and Data Engineering*, 4(4):301–316, 1992.
30. R. Srikant, Q. Vu, and R. Agrawal. Mining association rules with item constraints. In *Proceedings KDD'97*, pages 67–73, Newport Beach, USA, 1997. AAAI Press.
31. M. J. Zaki. Generating non-redundant association rules. In *Proceedings ACM SIGKDD'00*, pages 34–43, Boston, USA, 2000. ACM Press.