iZi: A New Toolkit for Pattern Mining Problems

Frédéric Flouvat, Fabien De Marchi, and Jean-Marc Petit

Université de Lyon, LIRIS, UMR5205 CNRS, F-69621, France
firstname.lastname@liris.cnrs.fr

Abstract. Pattern mining problems are useful in many applications. Due to a common theoretical background for such problems, generic concepts can be re-used to easier the development of algorithms. As a consequence, these problems can be implemented with only minimal effort, i.e. programmers do not have to be aware of low-level code, customized data structures and algorithms being available for free. A toolkit, called *iZi*, has been devised and applied to several problems such as itemset mining, constraint mining in relational databases and query rewriting in data integration systems. According to our first results, the programs obtained using our library offer a very good tradeoff between performances and development simplicity.

1 Introduction

In the last decade, many algorithms have been devised for pattern mining problems (such as for **F**requent **I**temset **M**ining). This is especially true for pattern mining problems known to be representable as set [1], as for instance, frequent itemset mining and variants [2,3], functional or inclusion dependency inference [4] or learning monotone boolean function [1]. Recently, other application domains have been identified such as discovery of schema matching [5] or query rewriting in integration systems [6]. In this setting, a common idea is to say that algorithms devised so far should be useful to answer these tasks and available open source implementations are a great source of know-how. Unfortunately, it seems rather optimistic to envision the application of most of publicly available implementations of frequent itemset mining algorithms, even for closely related problems. Indeed, sophisticated data structures specially devised for monotone predicates turn out to give very efficient algorithms but limit their application to other data mining tasks. As a consequence, low-level implementations hamper the rapid advances in the field.

Paper contribution. This paper takes advantage of the common theoretical background of problems isomorphic to boolean lattices. We provide a generic architecture and an implementation framework for this family of pattern mining problems. It encompasses efficient data structures for boolean lattice representation and several generic implementations of well known algorithms, such as a levelwise algorithm and a dualization-based algorithm. By the way, any problem can be implemented with only minimal effort, i.e. the programmers do not have to be aware of low-level code, customized data structures and algorithms being available for free. To the best of our knowledge, our contribution is the

A. An et al. (Eds.): ISMIS 2008, LNAI 4994, pp. 131–136, 2008.

only one providing a generic theoretical and implementation framework for this family of pattern mining problems. A toolkit called *iZi* has been devised and applied to several problems such as itemset mining, constraint mining in relational databases and query rewriting in data integration systems. According to our first results, the programs obtained using our toolkit have very interesting performances regarding simplicity of their development.

2 Related Work

Several packages and libraries have also been proposed for data mining. However, most of them do not focus on interesting pattern discovery problems and address more specifics data mining tasks (classification, clustering,...).

To our knowledge, only the DMTL library has objectives close to *iZi* w.r.t. code reusability and genericity. DMTL (Data Mining Template Library) is a C++ library for frequent pattern mining which supports any types of patterns representable as graphs (sets, sequences, trees and graphs). However, the motivations are quite different: while DMTL focuses on patterns genericity w.r.t. the frequency criteria only, *iZi* focuses on a different class of patterns but on a wider class of predicates. Moreover, *iZi* is based on a well established theoretical framework, whereas DMTL does not rely on such a theoretical background. However, DMTL encompasses problems that cannot be integrated into *iZi*, for instance frequent sequences or graphs mining since such problems are not isomorphic to a boolean lattice.

3 Theoretical Framework

We recall in this section the theoretical KDD framework defined in [1] for interesting pattern discovery problems. Given a database d, a finite language $\mathcal{L}$ for expressing patterns or defining subgroups of the data, and a predicate Q for evaluating whether a pattern $\varphi \in \mathcal{L}$ is true or "interesting" in d, the discovery task is to find the theory of d with respect to $\mathcal{L}$ and Q, i.e. the set $Th(\mathcal{L}, d, Q) = \{\varphi \in \mathcal{L} \mid Q(d, \varphi) \text{ is true}\}$.

Let us suppose a specialization/generalization relation between patterns of $\mathcal{L}$. Such a relation is a partial order $\preceq$ on the patterns of $\mathcal{L}$. We say that φ is more general (resp. more specific) than θ, if $\varphi \preceq \theta$ (resp. $\theta \preceq \varphi$). Let $(I, \preceq)$ be a partially ordered set of elements. A set $S \subseteq I$ is *closed downwards* (resp. *closed upwards*) if, for all $X \in S$, all subsets (resp. supersets) of X are also in S. The predicate Q is said to be monotone (resp. anti-monotone) with respect to $\preceq$ if for all $\theta, \varphi \in \mathcal{L}$ such that $\varphi \preceq \theta$, if $Q(d, \varphi)$ is true (resp. false) then $Q(d, \theta)$ is true (resp. false). As a consequence, if the predicate is monotone (resp. anti-monotone), the set $Th(\mathcal{L}, d, Q)$ is upward (resp. downward) closed, and can be represented by either his *positive border* or his *negative border*. The *positive border*, denoted by $\mathcal{B}d^+(Th(\mathcal{L}, d, Q))$, made up of the MOST SPECIALIZED true patterns when $Th(\mathcal{L}, d, Q)$ is downward closed, and the MOST SPECIALIZED false patterns when $Th(\mathcal{L}, d, Q)$ is upward closed. The *negative border*, denoted

by $\mathcal{B}d^-(Th(\mathcal{L}, d, Q))$, made up of the MOST GENERALIZED false patterns when $Th(\mathcal{L}, d, Q)$ is downward closed, and the MOST GENERALIZED true patterns when $Th(\mathcal{L}, d, Q)$ is upward closed.

The last hypothesis of this framework is that the problem must be representable as sets via an isomorphism, i.e. the search space can be represented by a boolean lattice (or subset lattice). Let $(\mathcal{L}, \preceq)$ be the ordered set of all the patterns defined by the language $\mathcal{L}$. Let E be a finite set of elements. The problem is said to be *representable as sets* if a bijective function $f : (\mathcal{L}, \preceq) \rightarrow (2^E, \subseteq)$ exists, and its inverse function f^{-1} is computable, such that: $X \preceq Y \iff f(X) \subseteq f(Y)$.

Example 1: Key mining. *Let us consider the key discovery problem in a relation, which can be enounced as follows: Let r be a relation over a schema R, extract the (minimal) keys satisfied in r. The patterns are : $\{X \mid X \subseteq R\} = \mathcal{P}(R)$. X is true if X is a superkey, i.e. if $|\pi_X(r)| = |r|$, where $\pi_X(r)$ is the projection onto X over r. It is clear that any superset of a superkey is also a superkey, it justifyes that only minimal keys are really interested. One can deduce that* minimal keys constitute the positive border of superkeys, with natural set inclusion. *The transformation function here is the identity since patterns are sets.*

4 A Generic Toolkit for Pattern Discovery

Based on the theoretical framework presented in section 3, we propose a C++ library, called *iZi*, for these pattern mining problems. The basic idea is to offer a toolbox for a rapid development of efficient and robust programs. The development of this toolkit takes advantage of the past experience to solve particular problems such as frequent itemsets mining, functional dependency mining, inclusion dependency mining and query rewriting...

4.1 Generic Algorithms and Data Structures

Even if this framework has been frequently used at a theoretical level, it has never been exploited at a technical point of view. One of our goal is to factorize some technical solutions which can be common to any pattern mining problem representable as sets. We are interested in algorithms and data structures that apply directly on sets, since they can be applied without any change for any problem, exploiting the isomorphic transformation. Our solution reuse some previous works done for frequent itemset mining, which is a problem "directly" representable as sets.

Currently, many algorithms from the multitude that has been proposed for the FIM problem could be implemented into *iZi*, from classical levelwise [7] and depth-first approaches, to more sophisticated dualization-based algorithms [8,4]. Since the generic part of our library only manipulates sets, we use a data structure based on prefix-tree (also called *trie*) specially devoted to this purpose [9]. They have not only a power of compression by factorizing common prefix in a set collection, but are also very efficient for candidate generation. Moreover,

prefix-trees are well adapted for inclusion and intersection tests, which are basic operations when considering sets.

4.2 Architecture

Figure 1 represents the architecture of our library. The figure 2 presents how the library solves the inclusion dependency (IND) mining problem using the levelwise strategy. The *algorithm* is initialized (*initialization* function) with patterns corresponding to singletons in the set representation, using the data (*data access* component). Then, during the execution of the algorithm, the *predicate* is used to test each pattern against the data. Before testing a element, the algorithm uses the *set transformation* function to transform each set generated into the corresponding pattern. This architecture is directly derived from the studied framework and has the main advantage of decoupling algorithms, patterns and data. Only the *predicate*, *set transformation* and *initialization* components are specifics to a given problem. Consequently, to solve a new problem, users may have to implement or reuse with light modifications some of these components.

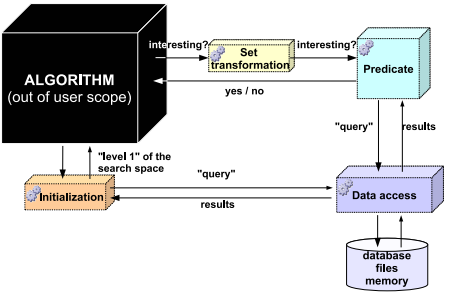

Fig. 1. *iZi* architecture

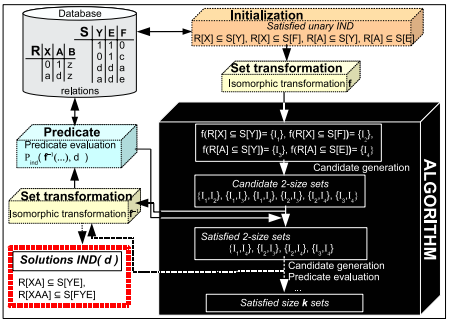

Fig. 2. IND mining example

As shown in figure 1, algorithms are **decoupled of the problems** and are **black box for users**. Each algorithm can be used directly to solve any problem fitting in the framework without modifications. This leads to the rapid construction of robust programs without having to deal with low-level details. Currently, the library offers two bottom-up algorithms (an *Apriori*-like [7,1] and *ABS* [4]), two top-down algorithms (top-down versions of *Apriori* and *ABS*) and depth-first strategies are currently being integrated.

Another important aspect of our library is that data access is totally decoupled of all other components (see figure 1). Currently, data access in most of the implementations is tightly coupled with algorithm implementations and predicates. Consequently, algorithms and "problem" components can be used with different data formats without modifications.

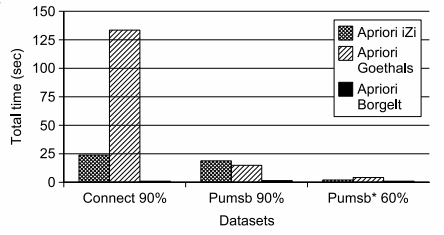

Fig. 3. Comparison of three *Apriori* implementations

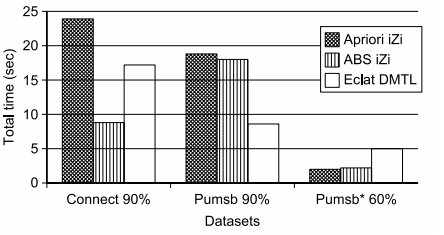

Fig. 4. Comparison of iZi and DMTL implementations

5 Experimentation

From our past experience in the development of pattern mining algorithms, we note that the adaptation of existing implementations is extremely difficult. In some cases, it implies the redevelopment of most of the implementation and could take more time than developing a new program from scratch.

Many problems have been implemented in our library along with several components. Our library was tested against 5 problems (and 5 different data formats): frequent and frequent essential itemset mining (FIMI file format [10,11]), inclusion dependency and key mining (FDEP file format [12], CSV file format or *MySQL* DBMS), and query rewriting in integration systems (specific file format). As indication, the use of our library to implement a program for the key mining problem has been done in less than one working day. Note that thanks to our library, an external module has been developed and integrated into a query rewriting prototype, allowing the scalability with respect to the number of views. From our point of view, this is a typical case where our library is very useful, providing a scalable component, almost for free, for the data-centric problems of a larger problem/application.

Performances. Implementations for FIM are very optimized, specialized, and consequently very competitive. The most performant ones are often the results of many years of research and development. In this context, our experimentations aims at proving that our generic algorithms implementations behave well compared to specialized ones. Moreover, we compare *iZi* to the DTML library, which is also optimized for frequent pattern mining. Experiments have been done on some classical benchmark datasets [10,11]. We compared our *Apriori* generic implementation to two others devoted implementations: one by B. Goethals and one by C. Borgelt [13]. The first one is a quite natural version, while the second one is, to our knowledge, the best existing *Apriori* implementation, developed in *C* and strongly optimized. Then, we compared "*iZi Apriori* and *ABS*" to the eclat implementation provided with DMTL. As shown in figures 3 and 4, our generic version has good performances with respect to other implementations. The difference between the two libraries is mainly due to the algorithm used during the experimentations. These results are very encouraging, in regards of the simplicity to obtain an operational program.

6 Discussion and Perspectives

In this paper, we have considered a classical problem in data mining: the discovery of interesting patterns for problems known to be *representable as sets*, i.e. isomorphic to a boolean lattice. As far as we know, this is the first contribution trying to bridge the gap between fundamental studies in data mining and practical aspects of pattern mining discovery. Our work concerns plenty of applications from different areas such as databases, data mining, or machine learning. Many perspectives exist for this work. First, we may try to integrate the notion of *closure* which appears under different flavors in many problems. The basic research around concept lattices could be a unifying framework. Secondly, we are interested in integrating the library as a plugin for a data mining software such as Weka. Analysts could directly use the algorithms to solve already implemented problems or new problems by dynamically loading their own components. Finally, a natural perspective of this work is to develop a declarative version for such mining problems using query optimization techniques developed in databases [14].

References

1. Mannila, H., Toivonen, H.: Levelwise search and borders of theories in knowledge discovery. Data Min. Knowl. Discov. 1(3), 241–258 (1997)
2. Agrawal, R., Imielinski, T., Swami, A.N.: Mining association rules between sets of items in large databases. In: SIGMOD Conference, pp. 207–216 (1993)
3. Bastide, Y., Taouil, R., Pasquier, N., Stumme, G., Lakhal, L.: Mining frequent patterns with counting inference. SIGKDD Explorations 2(2), 66–75 (2000)
4. De Marchi, F., Flouvat, F., Petit, J.M.: Adaptive strategies for mining the positive border of interesting patterns: Application to inclusion dependencies in databases. In: Constraint-Based Mining and Inductive Databases, pp. 81–101 (2005)
5. He, B., Chang, K.C.C., Han, J.: Mining complex matchings across web query interfaces. In: Das, G., Liu, B., Yu, P.S. (eds.) DMKD, pp. 3–10. ACM, New York (2004)
6. Jaudoin, H., Petit, J.M., Rey, C., Schneider, M., Toumani, F.: Query rewriting using views in presence of value constraints. In: Description Logics (2005)
7. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: VLDB, pp. 487–499 (1994)
8. Gunopulos, D., Khardon, R., Mannila, H., Saluja, S., Toivonen, H., Sharm, R.S.: Discovering all most specific sentences. ACM Trans. Database Syst. 28(2) (2003)
9. Bodon, F.: Surprising results of trie-based fim algorithms [11]
10. Bayardo Jr., R.J., Zaki, M.J. (eds.): FIMI 2003, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, USA (November 2003)
11. Bayardo Jr., R.J., Goethals, B., Zaki, M.J. (eds.): FIMI 2004, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, UK (November 2004)
12. Flach, P.A., Savnik, I.: Database dependency discovery: A machine learning approach. AI Commun. 12(3), 139–160 (1999)
13. Borgelt, C.: Efficient implementations of Apriori and Eclat [10]
14. Chaudhuri, S.: Data mining and database systems: Where is the intersection? IEEE Data Eng. Bull. 21(1), 4–8 (1998)