Mining Frequent Sequential Patterns under a Similarity Constraint

Matthieu Capelle, Cyrille Masson, and Jean-François Boulicaut*

Institut National des Sciences Appliquées de Lyon
Laboratoire d'Ingéniérie des Systèmes d'Information
F-69621 Villeurbanne Cedex, France
{cmasson,jfboulic}@lisi.insa-lyon.fr

Abstract. Many practical applications are related to frequent sequential pattern mining, ranging from Web Usage Mining to Bioinformatics. To ensure an appropriate extraction cost for useful mining tasks, a key issue is to push the user-defined constraints deep inside the mining algorithms. In this paper, we study the search for frequent sequential patterns that are also similar to an user-defined reference pattern. While the effective processing of the frequency constraints is well-understood, our contribution concerns the identification of a relaxation of the similarity constraint into a convertible anti-monotone constraint. Both constraints are then used to prune the search space during a levelwise search. Preliminary experimental validations have confirmed the algorithm efficiency.

1 Introduction

Many applications domains need for the analysis of sequences of events, like the design of personalized interface agents [5]. The extraction of frequent sequential patterns in huge databases of sequences has been heavily studied since the design of `apriori`-like algorithms [1,6]. Recent contributions consider the use of other criteria for the objective interestingness of the mined sequential patterns. Other kinds of user-defined constraints (e.g., enforcing a minimal gap between events) have been defined [3,10]. Provided a conjunction of constraints specifying the potential interest of patterns, the algorithmic challenge is to make use of these constraints in order to efficiently prune the search space.

In this paper, we are interested in the conjunction of two constraints: a *frequency constraint* and a *similarity constraint*. Two patterns are considered similar if the similarity measure between them is smaller than some threshold. Many research have been done in that field (for a survey, see, e.g., [7]), and the similarity measure we use allows us to identify a constraint that can be efficiently used inside our levelwise mining algorithm. Indeed, by mining sequential patterns satisfying a conjunction of an anti-monotone constraint (the frequency one) and a convertible anti-monotone constraint (the similarity one), we improve the global pruning efficiency during a levelwise exploration of the candidate patterns.

* Research partially funded by the European contract cInQ IST 2000-26469.

H. Yin et al. (Eds.): IDEAL 2002, LNCS 2412, pp. 1–6, 2002.

In Section 2, we introduce the sequential patterns and some useful properties of constraints. In Section 3, we define the distance measure and the similarity constraint we use. In Section 4, we demonstrate the relevancy of the approach by practical experiments. Section 5 is a short conclusion.

2 Basic Notions

Given a finite alphabet Σ, a sequential pattern M is an ordered list of symbols of Σ. L_M denotes the set of patterns that can be built using symbols of Σ. A sequential pattern M will be denoted $M_1 \rightarrow \ldots \rightarrow M_n$ where M_i is the i^{th} symbol of M and n is the *length* of M. $M_i \rightarrow M_{i+1}$ means that the element M_i preceeds the element M_{i+1}. $M' = M'_1 \rightarrow \ldots \rightarrow M'_m$ is a *sub-pattern* of $M = M_1 \rightarrow \ldots \rightarrow M_n$ if there exist some integers $i_1 < \ldots < i_m$ such that $M'_1 = M_{i_1}, \ldots, M'_m = M_{i_m}$. An *event* is denoted by a pair (A, t) where $A \in \mathcal{P}(\Sigma)\backslash\emptyset$ and $t \in \mathbb{N}$ (occurrence time of A). Several events can occur at the same time. An *event sequence* is a list of events sorted by increasing occurrence times. Generally, the *support* of a pattern M is its occurring rate in the data and a pattern is said *frequent* if its support exceeds an user-defined threshold.

Classical methods for frequent sequential pattern mining relies on an adaptation of the `apriori` paradigm [1,6]. The extraction process consists in several iterations of a levelwise algorithm, composed of three steps: generation of new candidates of length k, safe pruning of this set by using the frequency constraint (i.e., the extension of an infrequent pattern cannot be frequent), and counting the support of the remaining candidates. At each iteration, we extract longer patterns and it stops when no more candidates can be generated. Tractability can then be obtained by increasing the frequency threshold. However, considering only a frequency constraint often leads to a lack of focus. Indeed, if the user has an idea of some specific properties on the desired patterns, he might be overwhelmed by many useless results. A naive approach would be to check them in a post-processing phase: it means that the mined patterns will be among the frequent ones for a given frequency threshold. The challenge is to make an active use of the constraints, i.e., pushing them into the different steps of the extraction process. For that purpose, the constraints must satisfy some properties.

A constraint C_{am} is **anti-monotone** if, for all pattern M verifying C_{am}, all its sub-patterns verify C_{am}. For instance, the frequency constraint is anti-monotone thanks to safe pruning. Pushing anti-monotone constraints leads to quite effective algorithms [8,6]. However, this strong property only characterizes a few constraints. The concept of convertible constraint [9] allows to relax this definition while keeping an efficient pruning. In the following, we consider an equivalence relation R on the patterns. A constraint C_{cam} is **convertible anti-monotone** if, for all pattern M verifying C_{cam}, all its sub-patterns equivalents to M by R verify C_{cam}. When considering the prefix equivalence relation, such a constraint can be characterized with prefix decreasing functions [9] defined on L_M. Let $M, M' \in L_M$ such that M' is a prefix of M, f is a prefix decreasing function iff $f(M') \leq f(M)$.

Pushing non (convertible) anti-monotone constraints has been studied but the benefit cannot be guaranteed [3]. In this paper, given a collection of event sequences D and $C = C_{freq} \wedge C_{sim}(M_R)$ where C_{freq} denotes the frequency constraint and $C_{sim}(M_R)$ denotes a similarity constraint w.r.t. a reference pattern M_R, find the collection of patterns from L_M occurring in D and verifying C.

3 Similarity Constraint

Let us consider an user-defined reference pattern M_R and a set of operations $O_S = \{Ins, Del, Sub\}$ to which we associate costs depending on the position and the symbol on which they are applied. $C_{Ins}(X, i)$ is the insertion cost of symbol X at position $i, 0 \leq i \leq |M_R|$, $C_{Del}(X, i)$ is the deletion cost of X at position $i, 1 \leq i \leq |M_R|$, and $C_{Sub}(X, Y, i)$ is the substitution cost of X by Y at the position $i, 1 \leq i \leq |M_R|$. Costs values belong to the interval $[0, 1]$. The more its cost is close to 0, the more we consider the operation costly. Let $M \in L_M$, the cost $c(a_{M,M_R})$ of a_{M,M_R}, an alignment of M and M_R, is the product of all costs of its operations. The similarity score of a pattern M w.r.t. a pattern M_R is $sim(M, M_R) = max\{c(a_{M,M_R}) | a_{M,M_R}$ is an alignment of M and $M_R\}$. Notice that the maximal similarity of two patterns is 1. The algorithm that computes $sim(M, M_R)$ looks like the one used for editing distances [4]. It uses a matrix $\mathcal{M}$. However, we consider more complex operations and the cell $\mathcal{M}(0, 0)$ is initialized to 1 (we need a maximization criterion instead of a minimization criterion). More precisely, computing $\mathcal{M}$ relies on the following equations:

$$\forall j \in \{1, \ldots, |M|\}, \mathcal{M}(0, j) = \mathcal{M}(0, j-1) * C_{ins}(M[j], 0)$$
$$\forall i \in \{1, \ldots, |M_R|\}, \mathcal{M}(i, 0) = \mathcal{M}(i-1, 0) * C_{del}(M_R[i], i)$$
$$\mathcal{M}(i, j) = max \begin{cases} \mathcal{M}(i-1, j-1) * C_{sub}(M_R[i], M[j], i) \\ \mathcal{M}(i, j-1) * C_{ins}(M[j], i) \\ \mathcal{M}(i-1, j) * C_{del}(M_R[i], i) \end{cases}$$

Finally, the similarity score is the value $\mathcal{M}(|M_R|, |M|)$. To compute it, we can use dynamic programming: $\mathcal{M}(i, j)$ is determined by $\mathcal{M}(i-1, j-1), \mathcal{M}(i-1, j)$ and $\mathcal{M}(i, j-1)$. Thus, we can compute $sim(M, M_R)$ given the similarity score of its prefixes of length $|M| - 1$ The complexity of the algorithm is in $\mathcal{O}(|M_R|)$.

Table 1. Computation of $sim(A \to B \to E \to C \to D, A \to B \to C)$

		A	B	E	C	D
(0)	1	$\to 0.01$	$\to 0.001$	$\to 10^{-4}$	$\to 10^{-5}$	$\to 10^{-6}$
A(1)	$\downarrow 0.01$	$\searrow 1$	$\to 0.75$	$\to 0.56$	$\to 0.42$	$\to 0.32$
B(2)	$\downarrow 7,5.10^{-3}$	$\downarrow 0.75$	$\searrow 1$	$\searrow 0.67$	$\downarrow 0.32$	$\to 0.24$
C(3)	$\downarrow 7,5.10^{-5}$	$\downarrow 7,5.10^{-3}$	$\downarrow 0.01$	$\searrow 0.01$	$\searrow 0.67$	$\to 0.60$
sim	$7,5.10^{-5}$	$7,5.10^{-3}$	0.01	0.01	0.67	0.60
sim-pot	1	1	1	0.67	0.67	0.60

Example. Let $M_R = A \rightarrow B \rightarrow C$ and assume the costs given below, table 1 describes a computation of $sim(M, M_R)$.

$\forall X \in \Sigma, c_{Del}(X,1) = 0.01, c_{Del}(X,2) = 0.75, c_{Del}(X,3) = 0.01$
$\forall X \in \Sigma, c_{Ins}(X,0) = 0.01, c_{Ins}(X,1) = 0.75, c_{Ins}(X,2) = 0.01, c_{Ins}(X,3) = 0.9$
$$\forall i \in \{1,2,3\}, c_{Sub}(X,Y,i) = \begin{cases} 1 \text{ if } X = Y \\ 0.9 \text{ if } (X,Y) \in \{(A,D),(D,A),(B,E),(E,B)\} \\ 0.01 \text{ else} \end{cases}$$

Given a reference pattern $M_R \in L_M$, operations costs and a similarity threshold *min-sim*$\in [0,1]$, a pattern M is similar to M_R if $sim(M, M_R) \geq$ *min-sim*. We can now define the similarity constraint $C_{sim}(M_R)$ as follows: $M \in L_M$ satisfies $C_{sim}(M_R)$ iff M is similar to M_R.

Our algorithm (see [2] for details) is a variant of cSPADE [10]. However, it differs from it on some points:

- It does not consider any non (convertible) anti-monotone constraints.
- It supports the potential similarity constraint (defined below), i.e. a relaxation of $C_{sim}(M_R)$.
- It uses the prefix equivalence classes, instead of the suffix ones. This choice is due to the fact that our similarity constraint is based on prefix relations.

However, our constraint $C_{sim}(M_R)$ is neither anti-monotone, nor convertible anti-monotone. We have been looking for a relaxation of $C_{sim}(M_R)$ that would be convertible anti-monotone. Assume R_p denotes the equivalence relation "is prefix of". We define the potential similarity of a pattern M w.r.t. M_R as the maximal similarity score that its extensions can reach. Formally, *sim-pot*$(M, M_R) = \max\{sim(M', M_R) | M' \in L_M$ and $R_p(M, M')\}$. A corollary of this definition is that for all $M' \in L_M$ *sim-pot*$(M', M_R) \geq sim(M', M_R)$. To compute *sim-pot*$(M, M_R)$, we can take the largest value of the $(|M|+1)th$ column of $\mathcal{M}$. Indeed, it is not possible to increase this value since the cost of a further editing operation on M is smaller than 1.

Our first idea was to consider the potential similarity of a pattern as the result of the *sim-pot* function. As it can be shown that *sim-pot* is a prefix anti-monotone function, that would have lead to a convertible anti-monotone constraint ensuring a safe pruning, i.e. without affecting the correction of the mining algorithm. However, we have shown that its completeness is lost (see [2] for details). Thus, we defined a new function *sim-pot-comp* based on *sim-pot*. The idea is to consider this "complete" potential similarity of a pattern M as the potential similarity of its prefix of length $|M|-1$. Let $M', M \in L_M$ such that $R_p(M', M)$ and $|M| = |M'| + 1$, the function *sim-pot-comp* is defined by: *sim-pot-comp*$(M, M_R) =$ *sim-pot*(M', M_R). Given a similarity threshold *min-sim*, we now say that a pattern $M \in L_M$ is potentially similar to M_R if *sim-pot-comp*$(M, M_R) \geq$ *min-sim*. Thus, we can define a similarity constraint as follows: $\forall M \in L_M$, M satisfies $C_{sim-pot}(M_R)$ iff M is potentially similar to M_R. *sim-pot-comp* is still a prefix anti-monotone function, and thus $C_{sim-pot}(M_R)$, which is a relaxation of the initial constraint $C_{sim}(M_R)$, is convertible anti-monotone and is used in our complete and correct extraction algorithm.

Table 2. Files used for our experiments

Parameter	Description of the parameter	File F1	File F2	File F3
$\|D\|$	Number of event sequences	25k	50k	100k
$\|C\|$	Average number of elements per event sequence	10	10	10
$\|T\|$	Average element size	10	10	10
$\|S\|$	Average size of maximal sequential patterns	10	10	10
$\|N_S\|$	Number of maximal sequential patterns	5k	5k	5k
$\|N\|$	Number of items	100	100	2k

4 Experiments

Our prototype has been implemented in JAVA and experiments have been run on a Pentium III 800 Mhz with 512Mb of memory. We used 3 synthetic data sets (Table 2) obtained with the Quest generator [1]. Experiments aim at showing the relevancy of the active use of the conjunction of similarity and frequency constraints, compared to its use during post-processing. To simplify, we consider a fixed frequency threshold and only inclusions and deletions operations with uniform costs. Thus, the similarity threshold corresponds to a maximum number of operations allowed to align a pattern on the reference one. We are mainly interested in execution time, number of constrained and generated candidates, and respective selectivities of C_{freq} and $C_{sim}(M_R)$ constraints. Results are depicted on Figure 1. When the similarity threshold is 0, it means that the similarity constraint is taken into account in a post-processing phase. When it is equal to 1, it means that no editing operations are allowed. First, we can remark that the higher the similarity threshold is, the better the performances are. Moreover, we can see that the number of generated candidates and the execution time globally follow the same trends. Notice an exception for the file F3 with a similarity threshold of 1. Indeed, performances increase whereas the number of generated candidates remains the same. We can explain that by the decrease of the selectivity of support-based pruning and the increase of the selectivity of similarity-based pruning, whose validity is not difficult to check. Finally, we observe that the selectivity of the similarity-based constraint is inversely proportional to the selectivity of the support-based constraint.

5 Conclusion

We studied how to push a similarity constraint into a frequent sequential patterns mining algorithm. Experimental results confirm the relevancy of the approach. There are many possible extensions of this work: first, it worths to study the impact of more complex alignments methods (such as the `Viterbi` algorithm). Moreover, we have to use the prototype against real data sets and study the influence of the choice of editing operations costs.

[1] `http://www.almaden.ibm.com/cs/quest/index.html`

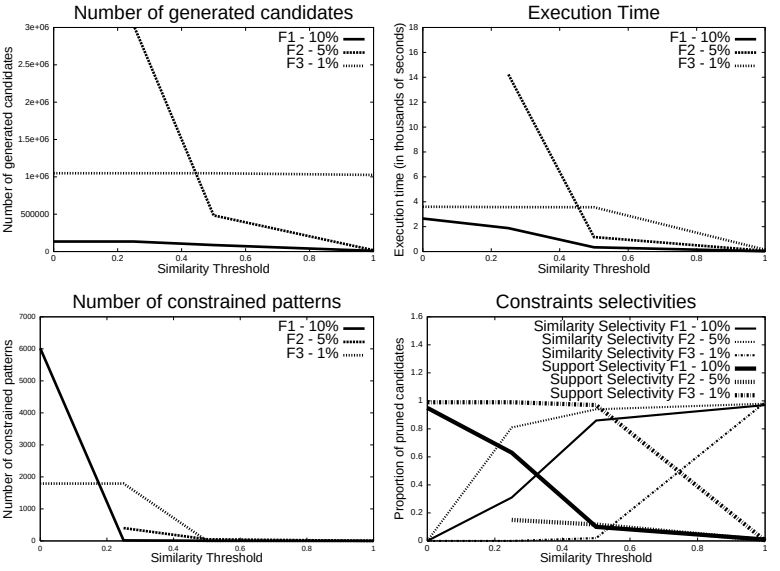

Fig. 1. Extractions results on the three files

References

1. R. Agrawal and R. Srikant. Mining sequential patterns. In *Proc. ICDE'95*, pages 3–14. IEEE Press, March 1995.
2. Matthieu Capelle. Extraction de motifs séquentiels sous contraintes (in french). Master's thesis, DEA ECD, INSA Lyon, Villeurbanne, France, September 2001.
3. M. N. Garofalakis, R. Rastogi, and K. Shim. SPIRIT: Sequential Pattern Mining with Regular Expression Constraints. In *Proc. VLDB'99*, pages 223–234. Morgan Kaufmann, September 1999.
4. Levenshtein. Binary codes capable of corecting deletions, insertions, and reversals, 1966.
5. J. Liu, Kelvin Chi Kuen Wong, and Ka Keung Hui. Discovering user behavior patterns in personalized interface agents. In *Proc. IDEAL 2000*, pages 398–403. Springer Verlag LNCS 1983, December 2000.
6. H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.
7. P. Moen. *Attribute, Event Sequence, and Event Type Simarity Notions for Data Mining*. PhD thesis, Dept. of Computer Science, University of Helsinki, Finland, February 2000.
8. R. T. Ng, L. V.S. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained associations rules. In *Proc. SIGMOD'98*, pages 13–24. ACM Press, June 1998.
9. J. Pei, J. Han, and L. V.S. Lakshmanan. Mining frequent itemsets with convertible constraints. In *Proc. ICDE'01*, pages 433–442. IEEE Computer Press, April 2001.
10. M. J. Zaki. Sequence mining in categorical domains: Incorporating constraints. In *Proc. CIKM'00*, pages 422–429. ACM Press, November 2000.