Proceedings 7th Int. Baltic Conf. on Databases and Information Systems DB&IS 2006,
Vilnius, Lithuania, July 3-6, 2006. pp. 77-82. IEEE.

About softness for inductive querying on sequence databases

Ieva Mitasiunaite
INSA Lyon
LIRIS CNRS UMR 5205
69621 Villeurbanne cedex, France
Ieva.Mitasiunaite@insa-lyon.fr

Jean-François Boulicaut
INSA Lyon
LIRIS CNRS UMR 5205
69621 Villeurbanne cedex, France
Jean-Francois.Boulicaut@insa-lyon.fr

***Abstract*— In many application domains (e.g., WWW usage mining, telecommunication data analysis, molecular biology), large sequence databases are available and yet under-exploited. The inductive database framework assumes that both such databases and the various patterns holding within them might be queryable. In this setting, queries which return patterns are called inductive queries and solving them is one of the main topics in database mining research. Indeed, constraint-based mining techniques on sequence databases have been studied extensively the last few years and efficient algorithms enable to compute complete collections of patterns (e.g., sequences) which satisfy conjunctions of monotonic and/or anti-monotonic constraints in potentially large sequence databases (e.g., minimal and maximal frequency constraints). Studying new applications of these techniques, we consider that fault-tolerance and softness are extremely important issues for tackling real-life data analysis. In this paper, we address some of the open problems when computing soft occurrences of patterns within database sequences instead of the classical exact matching ones. Such an extension is not trivial since it prevents the clever use of monotonicity for pruning the search space. We describe our proposal and we provide an experimental validation on real-life clickstream data which confirms the added value of this approach.**

I. Introduction

Collecting huge volumes of sequential data (i.e., the data is a collection of sequences or strings in a given alphabet) has become far easier in many application domains (e.g., E-commerce, networking, life sciences). Our ability to discover actionable patterns from such datasets remains however limited.

This paper focuses on substring mining, i.e., the searched patterns are strings as well. Knowledge discovery processes based on substrings in string databases have been studied extensively. We study a database perspective on such processes, the so-called inductive database approach [1], [2], [3], [4]. The idea is that many steps in complex knowledge discovery processes might be considered as queries which returns selected data instances and/or patterns holding in the data. Designing query languages which would support such a querying activity remains a long-term goal, and only preliminary results have been obtained for some quite specific scenarios (e.g., rather simple processes based on association rule mining [5]).

We focus on the so-called inductive queries, i.e., queries which declaratively express the constraints that have to be satisfied by the substring solution patterns. Typical challenges are (a) to identify useful primitive constraints to specify the a priori interestingness of the substring patterns in the data, and (b) to be able to design efficient and (when possible) complete solvers for computing every pattern which satisfies a combination of primitive constraints. In other terms, the core technology for developing inductive querying is related to the quite active research area of constraint-based data mining.

The state-of-the-art is that efficient algorithms are available for solving specific conjunctions of primitive constraints. For instance, many solvers have been designed for frequent sub-string or sequential patterns possibly combined with some more or less restricted types of syntactic constraints (e.g., [6], [7], [8], [9], [10], [11]). A promising approach has been developed by De Raedt and colleagues which consider arbitrary Boolean combination of primitive constraints which are either monotonic or anti-monotonic [12], [13], [14]. Indeed, a key issue for designing efficient solvers is to consider constraint properties (like anti-monotonicity and its dual monotonicity property) and exploit them for clever search space pruning. Many useful primitive constraints are monotonic (e.g., maximal frequency in a data set, enforcing a given sub-string occurrence) or anti-monotonic (e.g., minimal frequency, avoiding a given sub-string occurrence).

Some useful constraints are however neither anti-monotonic nor monotonic. It is the case of regular expression constraints for which efficient ad-hoc optimization strategies have been developed [15], [16], [17]. It can be also useful to look for patterns which are similar to a reference pattern. For instance, it is interesting to look for sequences of actions on a WWW site which are frequent for a given group of users, infrequent for another group and which are similar enough to an expected pattern specified by the WWW site designer. Such a primitive similarity constraint generally lacks from any monotonicity property. In [18], we have considered a possible decomposition of such a constraint into a conjunction of an anti-monotonic one and a monotonic one. It has been implemented on top of FAVST [14] and it enables to combine such a similarity constraint with other user-defined constraints.

This paper follows this direction of research and it considers the intrinsic limitations of these previous approaches which are all based on exact matching of candidate patterns with data instances. We are indeed interested in sequence database analysis for various application domains (e.g., biological data analysis, WWW usage mining, seismic data analysis). Even

though our raw data is fundamentally sequential (spatially or temporally ordered), the sequences to be mined are generally preprocessed: the data can be fundamentally noisy due to technological issues w.r.t. measurement, alphabet design can be somehow exploratory, but also the phenomena we would like to observe can be fundamentally fuzzy and such that soft computing approaches are needed.

Let us assume the realistic context, e.g., in molecular biology, where we consider that patterns play a major role in the studied mechanisms (e.g., gene regulation) while it is well-known that evolution has lead to many variants of the "originally" useful patterns. As a result, when looking at the major scientific question of transcription factor binding site in DNA sequences, molecular biologists consider consensus regular expressions instead of exact matching information over the gene promoter sequences.

In some cases, the alphabet is specified a priori (e.g., {a,c,t,g} for gene promoter sequences). In many cases, the alphabet has to be designed and/or computed. For instance, in a WWW usage mining context, assume that the raw data concern facts about "`Users` who performed `Operations` from `Machines`". Depending of the analysis task, many event types and thus alphabets might be considered (e.g., an operation is performed from a machine, a user has performed something, a user has performed something from a machine) and a meaningful browsing sequence will often be again some kind of consensus between different occurrences of similar browsing sequences. Finally, many data are available as numerical time series that can be analyzed by means of substring pattern algorithms provided that the data is discretized and thus encoded as a sequence of events in a "computed" alphabet. These methods are not always robust enough and again, soft occurrences of patterns might appear much more meaningful.

In this paper, we address some of the open problems when computing soft occurrences of patterns within database sequences. Our choice is to study first the impact of soft frequency constraints. Such an extension is not trivial since it prevents the clever use of monotonicity for pruning the search space. In Section 2, we provide the needed definitions and the problem setting. Section 3 introduces our definition of soft occurrences while Section 4 is dedicated to soft frequency constraints. In Section 5, we provide an experimental validation on real-life clickstream data which confirms the added value of our approach which has been implemented on top of the FAVST algorithm. Finally, Section 5 is a short conclusion.

II. Problem setting

Definition 1 (Basic notions on strings): Let Σ be a finite alphabet, a string σ over Σ is a finite sequence of symbols from Σ, and Σ^* denotes the set of all strings over Σ. Σ^* is our language of patterns $\mathcal{L}$ and we consider that the mined data set denoted r is a multi-set[1] of strings built on Σ. $|\sigma|$ denotes the length of a string σ and ϵ denotes the empty string. We note σ_i the i^{th} symbol of a string σ, $1 \leq i \leq |\sigma|$, so that $\sigma = \sigma_1\sigma_2\ldots\sigma_{|\sigma|}$. A sub-string σ' of σ is a sequence of contiguous symbols in σ, and we note $\sigma' \sqsubseteq \sigma$. σ is thus a super-string of σ', and we note $\sigma \sqsupseteq \sigma'$. We assume that, given a pattern $\phi \in \mathcal{L}$, the supporting set of strings in r is denoted by $ext(\phi, r) = \{\sigma \in r \mid \phi \sqsubseteq \sigma\}$.

Example 1: Let $\Sigma = \{a, b, c, d\}$. *abbc*, *abdbc*, ϵ are examples of strings over Σ. Examples of sub-strings for $abdbc$ are a and dbc. $aabdbcd$ is an example of a super-string of $abdbc$. If r is $\{abccb, adccba, ccabd\}$, $ext(ccb, r) = \{abccb, adccba\}$.

Definition 2 (Inductive queries): A constraint is a predicate that defines a property of a pattern and evaluates either to $true$ or $false$. An inductive query on $\mathcal{L}$ and r with parameters p is fully specified by a constraint Q and its evaluation needs the computation of $\{\phi \in \mathcal{L} \mid Q(\phi, r, p)$ is true$\}$ [19]. In the general case, Q is a Boolean combination of the so-called primitive constraints.

Definition 3 (Generalisation/specialisation): A pattern ϕ is more general than a pattern ψ (denoted $\phi \succeq \psi$) iff $\forall r\ ext(\phi, r) \supseteq ext(\psi, r)$. We also say that ψ is more specific than ϕ (denoted $\psi \preceq \phi$). Two primitive constraints can be defined: $MoreGeneral(\phi, \psi)$ is true iff $\phi \succeq \psi$ and $MoreSpecific(\phi, \psi)$ is true iff $\phi \preceq \psi$.

For strings, constraint $SubString(\phi, \psi) \equiv \phi \sqsubseteq \psi$ (resp., $SuperString(\phi, \psi) \equiv \phi \sqsupseteq \psi$) are instances of $MoreGeneral(\phi, \psi)$ (resp., $MoreSpecific(\phi, \psi)$). In other terms, $\forall \phi, \psi \in \mathcal{L}$, $\phi \succeq \psi$ iff $\phi \sqsubseteq \psi$.

Definition 4 (Examples of constraints): Given a threshold value v, typical syntactic constraints are $MinLen(\phi, v) \equiv |\phi| \geq v$ and $MaxLen(\phi, v) \equiv |\phi| \leq v$. Assume that $Fr(\phi, r)$ denotes the number of strings in r that are super-strings of ϕ, i.e., $|ext(\phi, r)|$. Given a threshold value f, $MinFr(\phi, r, f) \equiv Fr(\phi, r) \geq f$ (resp. $MaxFr(\phi, r, f) \equiv Fr(\phi, r) \leq f$) denotes a minimal (resp. maximal) frequency constraint in r.

Example 2: Assume $r = \{abd, abc, dc, c, dc\}$, we have $Fr(abd, r) = 1$, $Fr(dc, r) = 2$, $Fr(ad, r) = 0$, and $Fr(\epsilon, r) = 5$. $MinFr(dc, r, 2)$, $MaxFr(abd, r, 2)$, $MoreGeneral(c, dc)$, and $MinLen(abd, 3)$ are examples of satisfied constraints. $Q \equiv MinFr(\phi, r, 2) \wedge MaxFr(\phi, r, 4) \wedge MinLen(\phi, 2)$ is an example of an inductive query whose solution set is $\{ab, dc\}$.

The concept of anti-monotonicity and its dual notion of monotonicity is central to our work. When an anti-monotonic constraint like the minimal frequency is violated by a candidate, no more specific string (i.e., super-string) can satisfy it and it gives rise to pruning in the search space. This has been the key property for the many efficient algorithms which mine frequent strings. Negations of anti-monotonic constraints are called monotonic, e.g., the maximal frequency, and can lead to dual pruning strategies. This has been studied in detail in many papers, e.g., [19], [12].

Definition 5 ((Anti-)monotonicity): Let r be a data set, $\mathcal{L}$ be the pattern language and p be parameters. A constraint Q is anti-monotonic iff $\forall r\ and\ \forall \phi, \psi \in \mathcal{L}$, $\phi \succeq \psi \Rightarrow Q(\psi, r, p) \rightarrow Q(\phi, r, p)$. Dually, a constraint Q' is monotonic iff $\phi \preceq \psi \Rightarrow Q'(\psi, r, p) \rightarrow Q'(\phi, r, p)$.

Notice that conjunctions and disjunctions of anti-monotonic

[1]Data may contain multiple occurrences of the same sequence.

(resp. monotonic) constraints are anti-monotonic (resp. monotonic).

Example 3: $SuperString$, $MinLen$, and $MaxFr$ are monotonic constraints. $SubString$, $MaxLen$, and $MinFr$ are anti-monotonic ones.

The evaluation of some constraints on a pattern ϕ does not require to scan r (e.g., $SuperString$, $MaxLen$), while to evaluate some others, one needs to find the occurrences of ϕ in r. For instance, we have defined $MinFr(\phi, r, f)$ based on a number of strings where ϕ occurs exactly (i.e., the cardinality of $\{\sigma \in r$ such that $\sigma \sqsupseteq \phi\}$). However, in many application domains, measures based on such exact occurrences may be misleading. We consider it is important to study a frequency constraint based on soft-occurrences. The idea is that a string $\sigma \in r$ supports ϕ if σ contains a sub-string σ' similar enough to ϕ. σ' is then called a soft-occurrence of ϕ.

Extensive studies of (anti)-monotonicity properties have given rise to efficient search space pruning strategies. It is far more complex and sometime impossible to consider generic algorithms[2] for constraints that do not have the monotonicity properties. An "enumerate and test" strategy is never possible in real-life problems (large alphabets and/or large input sequences and/or huge number of input sequences). A solution might be to heuristically compute part of the solution. We are however convinced that completeness has an invaluable added value, and we prefer to study smart relaxation or decomposition strategies to solve our inductive queries on strings.

Therefore, our objective is to tackle a minimum soft-frequency constraint. It means that we have to consider similarity constraints via the soft-occurrence concept. In most application domains, a relevant similarity constraint is fundamentally neither monotonic nor anti-monotonic [18]. A minimum soft-frequency constraint itself is neither guaranteed to preserve the anti-monotonicity of $MinFr(\phi, r, f)$. We however seek for (anti)-monotonic properties of these constraints since they enable to exploit efficient generic strategies for solving combinations of (anti)-monotonic constraints [12], [13] and its optimization for string mining called FAVST [14]. The added value of building up our proposal on top of the FAVST framework is that it enables an efficient evaluation for not only soft-frequency constraints but also for arbitrary conjunctions of these constraints with other (anti)-monotonic constraints.

III. Defining soft-occurrences

The soft frequency of a pattern ψ is derived from the number of its soft occurrences ϕ, i.e., patterns ϕ such that $sim(\phi, \psi)$ where sim returns true when the two patterns are similar. It enables to use the similarity approach from [18], slightly modifying the monotonic sub-constraint such that its parameters become less connected to $|\psi|$.

Definition 6 (Longest Common Subsequence): Let x be a pattern from $\mathcal{L}$. A subsequence of x is any string w that can be obtained from x by deleting zero or more (not necessarily consecutive) symbols. More formally, w is a subsequence of x if there exists integers $i_1 < i_2 < \ldots < i_n$ s.t. $w_1 = x_{i_1}, w_2 = x_{i_2}, \ldots, w_n = x_{i_n}$. w is a Longest Common Subsequence (LCS) of x and ϕ if it is a subsequence of x, a subsequence of ϕ, and its length is maximal.

Notice that $|w| = lcs(\phi, x)$ and, in general, w is not unique.

Definition 7 (Insertions, Deletions): Let x be the reference pattern, ϕ be a candidate pattern from $\mathcal{L}$. Let fix any LCS of ϕ and x, and denote the symbols of ϕ (resp. x) that do not belong to a LCS as deletions (resp. insertion). The number of deletions (resp. insertions) is $Dels(\phi, x) = |\phi| - lcs(\phi, x)$ (resp. $Ins(\phi, x) = |x| - lcs(\phi, x)$). Notice that x can be produced from ϕ by deleting from ϕ the deletions and inserting into ϕ the insertions.

Lemma 1: Assume $x, \phi \in \mathcal{L}$, $\phi' \sqsubseteq \phi$, w one LCS of ϕ and x, and w' one LCS of ϕ' and x. We have $|w| = lcs(\phi, x) \geq lcs(\phi', x) = |w'|$.

The formal proofs of this lemma and the other propositions or properties are available and can be asked to the authors.

Definition 8 (Max Insertions constraint): Let x be the reference pattern, ϕ be a candidate pattern from $\mathcal{L}$, and ins a threshold value. The Maximum Insertions constraint is defined as $MaxIns(\phi, x, ins) \equiv Ins(\phi, x) \leq ins$.

Proposition 1: $MaxIns(\phi, x, ins)$ is monotonic.

Example 4: Assume $x = cbcddda$. Patterns $\phi_1 = dbddda$ and $\phi_2 = bcddada$ satisfy $MaxIns(\phi, x, 2)$: $Ins(\phi_1, x) = |x| - |bddda| = 2$ and $Ins(\phi_2, x) = |x| - |bcddda| = 1$. Pattern $\phi_3 = accadcdccccddddd$ also satisfies it: $Ins(\phi_3, x) = |x| - |ccddd| = 2$.

Constraint $MaxIns(\phi, x, ins)$ enables to specify a degree of similarity (i.e., a maximum number of non matching symbols on reference), and thus to capture patterns which are similar to the reference one. Note however that $MinLCS(\phi, x, l)$ does not restrict the dissimilarity of a candidate. Thus, we need for a second constraint that would bound the number of "errors" within a candidate.

Definition 9 (Max Deletions constraint): Let x be the reference pattern, ϕ be a candidate pattern from $\mathcal{L}$, and $dels$ a threshold value. The Maximum Deletions constraint is defined as $MaxDels(\phi, x, dels) \equiv Dels(\phi, x) \leq dels$.

Proposition 2: $MaxDels(\phi, x, d)$ is anti-monotonic.

Definition 10 (A Similarity constraint): Given a reference pattern x and thresholds ins and $dels$, our similarity constraint for a pattern ϕ w.r.t. x is defined as $C_{sim}(\phi, x, ins, dels) \equiv MaxIns(\phi, x, ins) \wedge MaxDels(\phi, x, dels)$.

Example 5: Continuing Example 4, patterns ϕ_1 and ϕ_2 satisfy $C_{sim}(\phi, x, 2, 1)$. Pattern $\phi_4 = dbdddca$ satisfies $C_{sim}(\phi, x, 2, 2)$ since $lcs(\phi_4, x) = |x| - |bddda| = 2$. Pattern ϕ_3 does not satisfy neither $C_{sim}(\phi, x, 2, 1)$ nor $C_{sim}(\phi, x, 2, 2)$.

Remark 1: Note that $|x| - ins \leq |\phi| \leq |\phi| + dels$ is an essential condition for $C_{sim}(\phi, x, ins, dels)$ to be satisfied.

Definition 11 (Soft-occurrence): If a string $\sigma \in r$ contains ϕ such that $C_{sim}(\phi, \psi, ins, dels)$ is satisfied, we say that ϕ is a soft-occurrence of ψ denoted as $sOcc(\psi, ins, dels)$.

[2] Algorithms not dedicated to a specific combination of primitive constraints

IV. Definition of soft-frequency

Definition 12 (Soft-frequency): If $sOcc(\phi, ins, dels)_1$, $sOcc(\phi, ins, dels)_2$, ..., $sOcc(\phi, ins, dels)_n$ are the soft-occurrences for ϕ in r, the soft-frequency of ϕ $SoftFr(\phi, r, ins, dels)$ is $|ext(sOcc(\phi, ins, dels)_1, r) \cup \ldots \cup ext(sOcc(\phi, ins, dels)_n), r)|$.

Definition 13 (Minimum Soft-Frequency): Given a user-defined threshold f, the Minimum Soft-Frequency constraint is defined as $MinSoftFr(\phi, r, f, ins, dels) \equiv SoftFr(\phi, r, ins, dels) \geq f$.

Example 6: Continuing from Example 2. $SoftFr(abd, r, 1, 1) = 2$ and $\{bd, abc, abd, ab\}$ are the soft-occurrences of abd on r. $SoftFr(dc, r, 1, 1) = 5$ and $\{c, dc, d, bc, bd\}$ are the soft-occurrences of dc on r. $MinSoftFr(dc, r, 4, 1, 1)$ is an example of satisfied constraint.

Proposition 3: Constraint $MinSoftFr(\phi, r, ins, dels)$ is anti-monotonic when $dels \geq ins$.

Remark 2: The relation $sim(\phi, x)$ induced by $C_{sim}(\phi, x, ins, dels)$ is symmetric (i.e., if $sim(\phi, x)$ then $sim(x, \phi)) \iff ins = dels$.

Remark 3: In most of the application domains, it makes sense to use a similarity relation which is symmetric. With symmetric $C_{sim}(\phi, x, ins, dels)$, $MinSoftFr(\phi, r, f, ins, dels)$ has the desired anti-monotonicity property.

V. Experiments

This section concerns our empirical evaluation of the $MinSoftFr(\phi, r, ins, dels)$ constraint solving.

To the best of our knowledge, FAVST algorithm [14] is among the best algorithms for mining strings that satisfy conjunctions of anti-monotonic and monotonic constraints. It uses a Version Space Tree (VST) [12] designed to index a version space of strings. The FAVST framework enables to push constraints $C_{sim}(\phi, x, ins, dels)$, $MinSoftFr(\phi, r, ins, dels)$ (when $dels \geq ins$), and their arbitrary conjunctions with other anti-(monotonic) constraints deeply into the extraction phase.

We have implemented FAVST in C and we decided to process KDD Cup 2000 real-world clickstream datasets [20] on a Intel(R) Pentium(R) M 1.69GHz processor (1GB main memory).

We have extracted attributes "Session ID", "Request Sequence", "Request Template" and we produced time ordered sequences of templates requested for each session. We had $234,954$ sessions and 137 different request templates. As a result, the used sequence database was containing $234,954$ strings over an alphabet of 137 symbols. The shortest input string had a length of 1 while the largest one had a length of $5,487$.

A. $MinSoftFr(\phi, r, ins, dels)$ and $C_{sim}(\phi, x, ins, dels)$

Solving $MinSoftFr(\phi, r, ins, dels)$ for ϕ means to solve $C_{sim}(\psi, \phi, ins, dels)$ to find all patterns ψ that are soft-occurrences of ϕ for parameters ins and $dels$.

We considered two strategies to push $C_{sim}(\psi, \phi, ins, dels)$.

The first one denoted "ExploitDelsAndIns" explores the search space by exploiting both $MaxDels(\psi, \phi, dels)$ and $MaxIns(\psi, \phi, ins)$.

The second one, denoted "ExploitIns" explores the search space by exploiting $MaxLen(\psi, |\psi| + dels)$ (see Remark 1) and $MaxIns(\psi, \phi, ins)$. It exploits $MaxDels(\psi, \phi, dels)$ only to ensure that $C_{sim}(\psi, \phi, ins, dels)$ is satisfied.

The idea underlying such strategies is to get a trade-of between the size of pruned search space and the cost of constraint evaluation which gives rise to such a pruning.

When $dels < ins$, $MinSoftFr(\psi, r, ins, dels)$ can not be exploited efficiently, i.e., it has to be evaluated for every ψ on r. Obviously, extraction becomes intractable. A solution might be firstly prune the search space by pushing other (anti)-monotonic constraints (e.g., $MinLen$, $MaxLen$, $MinFr$, $MaxFr$) and then post-process $MinSoftFr(\psi, r, ins, dels)$ on such a restricted ψ space.

B. Studying $C_{sim}(\phi, x, ins, dels)$ pruning

Our definition of $C_{sim}(\phi, x, ins, dels)$ has given rise to (anti)-monotonic constituents which can be used for efficient pruning. In case we would like to evaluate a similarity constraint which no monotonicity properties, it would lead to a constraint checking for every candidate pattern ϕ. Nevertheless, many known similarity measures on strings (e.g., [21], [22] or edit distances [23]) enable to infer the minimal and maximal length of the patterns ϕ which might be similar to ψ. It means that we can exploit $MinLen(\phi, v)$ and $MaxLen(\phi, v)$ to prune the search space for a number of similarity constraints.

To empirically evaluate the impact of $C_{sim}(\phi, \psi, ins, dels)$ evaluation when checking $MinSoftFr(\psi, r, f, ins, dels)$ and thus looking for $sOcc(\psi, ins, dels)$, we have computed $SoftFr(\psi, r, 1, 1)$ for all ψ satisfying $IQ \equiv MinFr(\psi, 0.01\%) \wedge MinLen(\psi, 4) \wedge MaxLen(\psi, 32)$. Three strategies have been used: "ExploitDelsAndIns", "ExploitIns", and pushing only $MinLen(\phi, |\psi| - ins)$ and $MaxLen(\phi, |\phi| + dels)$ (see Remark 1), denoted "DoNotExploit". There are $6,945$ patterns satisfying IQ, $5,013$ of them are of length ≤ 6. The graph given in Fig. 1 plots a mean value of number of tests of $C_{sim}(\phi, \psi, 1, 1)$ for each length of ψ.

First, let us observe that "ExploitIns" always performs better than "DoNotExploit". For $4 \leq |\psi| \leq 7$, "ExploitIns" over-performs "DoNotExploit" by approximately 1000 (i.e., "DoNotExploit" evaluated the similarity constraint for approximately 1000 more candidates than "ExploitIns"), for $8 \leq |\psi| \leq 11$ by approximately 100, from $\psi \geq 20$ they start to converge. "ExploitDelsAndIns" over-performs "DoNotExploit" approximately 4 times for $4 \leq |\psi| \leq 6$. The behaviors coincides at $|\psi| = 7$, $|\psi| = 8$, and thereinafter "ExploitDelsAndIns" performs poorer than both "ExploitIns" and "DoNotExploit". This suggests to use an adaptive strategy that would dynamically enforce either "ExploitDelsAndIns" or "ExploitIns".

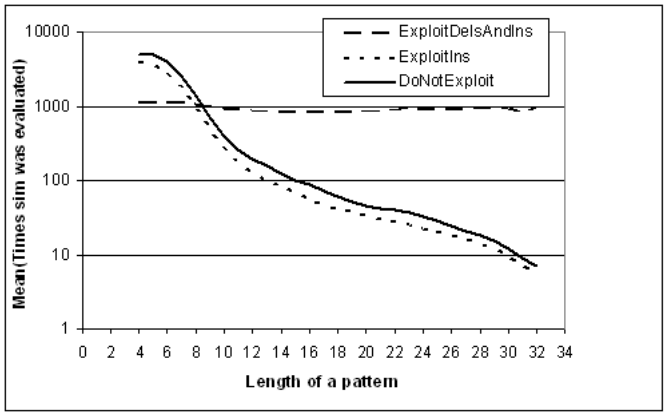

Fig. 1. Strategies to compute $C_{sim}(\phi, \psi, ins, dels)$

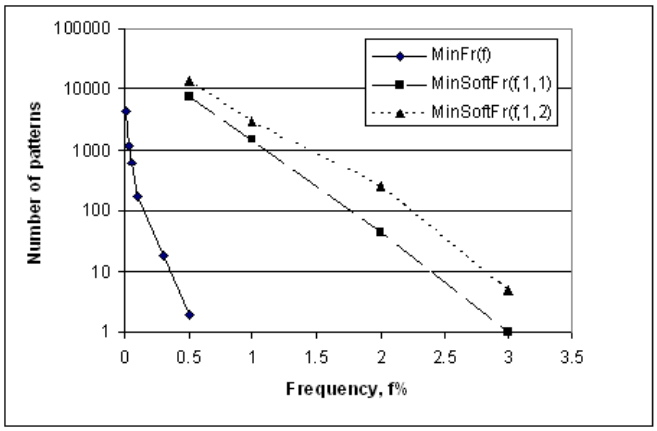

Fig. 2. $MinFr(\phi, r, f)$ and $MinSoftFr(\phi, r, sf, ins, dels)$ pruning

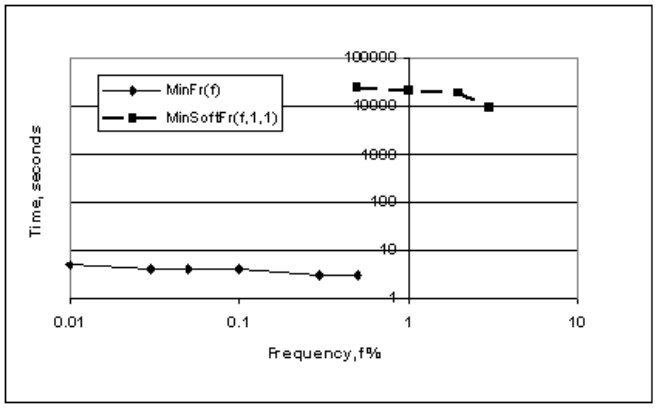

Fig. 3. Run time $MinFr(\phi, r, f)$ and $MinSoftFr(\phi, r, sf, 1, 1)$

C. Comparative study of frequency, soft-frequency, and degrees of softness

We have performed experiments to assess the soft-frequency w.r.t. frequency, and the impact of different combinations of parameters ins and $dels$ on resulting "softness". We have computed $SoftFr(\psi, 1, 1)$, $SoftFr(\psi, 1, 2)$, and $SoftFr(\psi, 2, 1)$ for all ψ satisfying $IQ \equiv MinFr(\psi, 0.01\%) \wedge MinLen(\psi, 7) \wedge MaxLen(\psi, 7)$. We got 796 solution patterns. Table I provides some statistical summary.

Let us notice that, in most cases, the frequency of a pattern is quite small w.r.t. its soft-frequency. Also, $SoftFr(1, 1)$ tends to be smaller than $SoftFr(1, 2)$ and $SoftFr(2, 1)$. Finally, $SoftFr(1, 2)$ tends to be smaller than $SoftFr(2, 1)$.

D. Studying pruning thanks to minimal (soft-)frequency

We performed experiments to compare the selectivity of $MinSoftFr(\phi, r, f, ins, dels)$ and $MinFr(\phi, r, f)$ constraints. For this purpose, we have computed solutions to $IQ_1 \equiv MinFr(\phi, r, f) \wedge MinLen(\phi, 5) \wedge MaxLen(\phi, 8)$, $IQ_2 \equiv MinSoftFr(\phi, r, f, 1, 1) \wedge MinLen(\phi, 5) \wedge MaxLen(\phi, 8)$, and $IQ_3 \equiv MinSoftFr(\phi, r, f, 1, 2) \wedge MinLen(\phi, 5) \wedge MaxLen(\phi, 8)$. The size of the corresponding solutions is plotted against different f thresholds on the graph given in Fig. 2.

$MinFr(\phi, r, f)$ with even very small frequency thresholds drastically prunes, while the same frequency values for $MinSoftFr(\phi, r, f, ins, dels)$ are not selective at all. These extractions emphasize the added value for $MinSoftFr(\phi, r, f, ins, dels)$: one might assume that at least 1% of the sessions share something, and $MinSoftFr(\phi, r, 1\%, ins, dels)$ enables to extract these common regularities while $MinFr(\phi, r, 1\%)$ leads to an empty collection.

Fig. 3 plots the run time to solve IQ_1 and IQ_2 with different f values. A rather poor time efficiency to process $MinSoftFr(\phi, r, f, ins, dels)$ is not surprising. First, we do not use the well-known $MinFr(\phi, r, f)$ efficient pruning, and we evaluate $MinSoftFr(\phi, r, 1\%, ins, dels)$ for all candidates occurring at least once in the data. Then, $MinSoftFr(\phi, r, f, ins, dels)$ is far less selective than $MinFr(\phi, r, f)$. Finally, its evaluation is expensive. When $ins = dels$, one can exploit the symmetric property of the underlying similarity relation. An adaptive computation strategy, dynamic choice between $ExploitInsAndDels$ and $ExploitIns$ (see Section V-B), can be considered.

It is clear that further experiments are needed for a deeper empirical evaluation of the minimum soft-frequency constraint. Basic ideas for optimization (e.g., adaptive strategies) have been however identified.

VI. Conclusion

The promising vision of the inductive database framework is that expert data owners might be able to query both the data and the patterns holding in the data. In this paper, we have considered the so-called inductive querying problem on string databases, i.e., the evaluation of constraints which specify declaratively the desired properties for string patterns. Solving by means of generic algorithms, arbitrary combinations of useful primitive constraints is challenging. In this paper, we have started to revisit constraint-based mining of substring patterns by introducing soft-frequency constraints. It might be

TABLE I

FREQUENCY AND SOFT-FREQUENCY

	Fr	$\frac{Fr \times 100\%}{SoftFr(1,1)}$	$\frac{Fr \times 100\%}{SoftFr(1,2)}$	$\frac{Fr \times 100\%}{SoftFr(2,1)}$	$\frac{SoftFr(1,1)}{SoftFr(1,2)}$	$\frac{SoftFr(1,1)}{SoftFr(2,1)}$
Min val	23	1.53	1.14	0.54	0.45	0.06
Max val	843	100	100	97.6	1	0.99
Mean val	57.89	14.61	12.37	6.9	0.76	0.37
Stand Dev	70.63	21.26	20.6	14.72	0.09	0.18

quite useful when dealing with intrinsically noisy data sets. We formalized an approach to soft-frequency constraint checking which can take the most from efficient strategies for solving conjunctions of monotonic and anti-monotonicity constraints. As a result, the analysts can combine our soft-frequency constraints with other many other user-defined constraints of interest. Preliminary applications of these ideas on gene promoter sequence database analysis are ongoing.

ACKNOWLEDGMENT

We wish to thank Blue Martini Software for contributing the KDD Cup 2000 data. This research is partly funded by ACI CNRS MD 46 Bingo and by EU contract IST-FET IQ FP6-516169 (FET arm of the IST programme).

REFERENCES

[1] T. Imielinski and H. Mannila, "A database perspective on knowledge discovery," *CACM*, vol. 39(11), pp. 58-64, 1996.

[2] L. De Raedt, "A perspective on inductive databases," *SIGKDD Explorations*, vol. 4(2), pp. 69-77, 2003.

[3] J-F. Boulicaut, "Inductive databases and multiple uses of frequent itemsets: the cInQ approach," In *Database Technologies for Data Mining - Discovering Knowledge with Inductive Queries*, Springer-Verlag LNCS 2682, 2004, pp. 1-23.

[4] J-F. Boulicaut, L. De Raedt, and H. Mannila (Editors), *Constraint-based mining and inductive databases*, Springer-Verlag LNAI 3848, 2006, 405 pages.

[5] M. Botta, J-F. Boulicaut, C. Masson, and R. Meo, "Query languages supporting descriptive rule mining: a comparative study," In *Database Technologies for Data Mining - Discovering Knowledge with Inductive Queries*, Springer-Verlag LNCS 2682, 2004, pp. 27-56.

[6] R. Srikant and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements," In *Proceedings EDBT*, 1996, Springer-Verlag, pp. 3-17.

[7] F. Masseglia, F. Cathala, and P. Poncelet, "The PSP approach for mining sequential patterns," In *Proceedings PKDD*, 1998, Springer-Verlag, pp. 176-184.

[8] M.J. Zaki, "Spade: An efficient algorithm for mining frequent sequences," *Machine Learning*, vol. 42(1-2), pp. 31-60, 2001.

[9] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M-C. Hsu, "Freespan: frequent pattern-projected sequential pattern mining," In *Proceedings ACM SIGKDD*, 2000, pp. 355-359.

[10] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M-C. Hsu, "Prefixspan: Mining sequential patterns by prefix-projected growth," In *Proceedings IEEE ICDE*, 2001, pp. 215-224.

[11] M. Leleu, C. Rigotti, J-F. Boulicaut, and G. Euvrard, "Constraint-based mining of sequential patterns over datasets with consecutive repetitions," In *Proceedings PKDD*, 2003, Springer-Verlag, pp. 303–314.

[12] L. De Raedt, M. Jaeger, S. Dan Lee, and H. Mannila, "A theory of inductive query answering," In *Proceedings IEEE ICDM*, 2002, pp. 123-130.

[13] S. Dan Lee and L. De Raedt, "An algebra for inductive query evaluation," In *Proceedings IEEE ICDM*, 2003, pp. 147-154.

[14] S. Dan Lee and L. De Raedt, "An efficient algorithm for mining string databases under constraints," In *Proceedings KDID*, 2004, Springer-Verlag, pp. 108-129.

[15] M. N. Garofalakis, R. Rastogi, and K. Shim, "Spirit: Sequential pattern mining with regular expression constraints," In *Proceedings VLDB*, 1999, Morgan Kaufmann Publishers Inc., pp. 223-234.

[16] J. Pei, J. Han, and W. Wang, "Mining sequential patterns with constraints in large databases," In *Proceedings ACM CIKM*, 2002, pp. 18-25.

[17] H. Albert-Lorincz and J.-F. Boulicaut, "Mining frequent sequential patterns under regular expressions: a highly adaptive strategy for pushing constraints," In *Proceedings SIAM DM*, 2003, pp. 316-320.

[18] I. Mitasiunaite and Jean-François Boulicaut, "Looking for monotonicity properties of a similarity constraint on sequences," In *Proceedings of ACM Symposium of Applied Computing SAC 2006, Special Track on Data Mining*, ACM Press, pp. 546-552.

[19] H. Mannila and H. Toivonen, "Levelwise search and borders of theories in knowledge discovery," *Data Mining and Knowledge Discovery*, vol. 1(3), pp. 241-258, 1997.

[20] R. Kohavi, C. Brodley, B. Frasca, L. Mason, and Z. Zheng, "KDD-Cup 2000 organizers' report: Peeling the onion," *SIGKDD Explorations*, vol. 2(2), pp. 86-98, 2000.

[21] M.-F. Sagot, V. Escalier, A. Viari, and H. Soldano, "Searching for repeated words in a text allowing for mismatches and gaps," In *Proceedings 2nd South American Workshop on String Processing*, 1995, pp. 87-100.

[22] D. Sankoff and J. Kruskal, *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, Reading, Mass. Addison-Wesley, 1983.

[23] V. Levenshtein, "Binary codes capable of correcting spurious insertions and deletions of ones," *Probl. Inf. Transmission*, vol. 1, pp. 8-17, 1965.