In: Databases and Information Systems IV Vol.155 Frontiers in Artificial Intelligence and Applications
O. Vasilecas, J. Eder, A. Caplinskas (Eds.), 2007, pp. 117-132. IOS Press

Introducing Softness into Inductive Queries on String Databases

Ieva MITASIUNAITE and Jean-François BOULICAUT
INSA Lyon, LIRIS CNRS UMR 5205, 69621 Villeurbanne cedex, France
{Ieva.Mitasiunaite,Jean-Francois.Boulicaut}@insa-lyon.fr

Abstract. In many application domains (e.g., WWW mining, molecular biology), large string datasets are available and yet under-exploited. The inductive database framework assumes that both such datasets 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 core research topics for data mining. Indeed, constraint-based mining techniques on string datasets have been studied extensively. Efficient algorithms enable to compute complete collections of patterns (e.g., substrings) which satisfy conjunctions of monotonic and/or anti-monotonic constraints in large datasets (e.g., conjunctions of minimal and maximal support constraints). We consider that fault-tolerance and softness are extremely important issues for tackling real-life data analysis. We address some of the open problems when evaluating soft-support constraint which implies the computations of pattern soft-occurrences instead of the classical exact matching ones. Solving efficiently soft-support constraints is challenging since it prevents from the clever use of monotonicity properties. We describe our proposal and we provide an experimental validation on real-life clickstream data which confirms the added value of this approach.

Keywords. Inductive databases, fault-tolerance, sequence mining

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 datasets 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]). As a needed step towards query languages for inductive databases,

it is interesting to investigate different pattern domains and this chapter is considering inductive queries on the string pattern domain. Such queries declaratively express the constraints that have to be satisfied by the solution patterns. Typical challenges are (a) to identify useful primitive constraints to specify the a priori interestingness of the 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.

The state-of-the-art is that efficient algorithms are available for solving specific conjunctions of primitive constraints on string patterns. For instance, many solvers have been designed for frequent substring 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 support in a data set, enforcing a given sub-string occurrence) or anti-monotonic (e.g., minimal support, avoiding a given sub-string occurrence).

Some useful constraints are however neither anti-monotonic nor monotonic. This is the case of regular expression constraints, i.e., rich syntactic constraints which enforce the solution patterns to belong to the language of a given regular expression. A typical application in WWW usage mining would be to look for the frequent sequences of clicks which are matching a given path through the WWW site (conjunction of a minimal support constraint with a regular expression constraint). Efficient ad-hoc optimization strategies have been developed for such a conjunction [15,16,17]. In many applications, it is also interesting to look for patterns which are similar enough to a reference pattern. For instance, it might be useful to look for sequences of clicks 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 studied a possible decomposition of such a constraint into a conjunction of an anti-monotonic one and a monotonic one. It was implemented on top of the FAVST algorithm [14] such that it is now possible to combine efficiently that kind of similarity constraint with other monotonic or anti-monotonic user-defined constraints.

This paper follows this direction of research and 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 string database analysis for various application domains (e.g., WWW usage mining, seismic or telecommunication data analysis, molecular biology). Even though our raw data is fundamentally sequential (spatially or temporally ordered), the strings 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.

In many application domains, the string 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 per-

formed something from a machine) and a meaningful browsing sequence will often be some kind of consensus between different occurrences of similar browsing sequences. Also, in many cases of sequential data mining, 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. Finally, the most famous case of degenerated data concerns molecular biology. We can consider that patterns on gene promoter sequences (e.g., substrings within DNA sequences) play a major role in gene regulation but 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 this paper, we address some of the open problems when computing soft-occurrences of patterns within string dataset. This is a significant revision and extension of the paper [19]. For instance, an original and detailed experimental validation has been carried out to replace the preliminary results described in [19]. In Section 1, we provide the needed definitions and the problem setting. Section 2 introduces our definition of soft-occurrences and our formalization of soft-support constraints. The proofs of properties are available and can be asked to the authors. They are omitted here because of space limitation. In Section 4, we provide an in-depth experimental validation on real-life clickstream data which confirms the added value of our approach. Finally, Section 5 is a short conclusion.

1. 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$\}$ [20]. In the general case, Q is a Boolean combination of the so-called primitive constraints.*

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

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 $\phi \sqsubseteq \psi$. Given a threshold value n, one can limit a maximal number of occurrences of ψ in ϕ and thus define $ContainsAtMost(\phi, n, \psi)$.

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 $Supp(\phi, r)$ denotes the number of strings in r that are super-strings of ϕ, i.e., $|ext(\phi, r)|$. Given a threshold value f, $MinSupp(\phi, r, f) \equiv Supp(\phi, r) \geq f$ (resp. $MaxSupp(\phi, r, f) \equiv Supp(\phi, r) \leq f$) denotes a minimal (resp. maximal) support constraint in r.*

Example 2 *Assume $r = \{abd, abc, dc, c, dc\}$, we have $Supp(abd, r) = 1$, $Supp(dc, r) = 2$, $Supp(ad, r) = 0$, and $Supp(\epsilon, r) = 5$. $MinSupp(dc, r, 2)$, $MaxSupp(abd, r, 2)$, $MoreGeneral(c, dc)$, and $MinLen(abd, 3)$ are examples of satisfied constraints. $Q \equiv MinSupp(\phi, r, 2) \wedge MaxSupp(\phi, r, 4) \wedge MinLen(\phi, 2)$ 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 support is violated by a candidate string, none of its more specific strings (i.e., super-strings) can satisfy it and this 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 support, and can lead to dual pruning strategies. This has been studied in detail in many papers, e.g., [20,12].

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

Example 3 *$SuperString(\phi, \psi)$, $MinLen(\phi, v)$, and $MaxSupp(\phi, r, f)$ are monotonic constraints. $SubString(\phi, \psi)$, $ContainsAtMost(\phi, n, \psi)$, $MaxLen(\phi, v)$, and $MinSupp(\phi, r, f)$ are anti-monotonic ones.*

The evaluation of some constraints on a pattern ϕ does not require to scan r (e.g., $SuperString(\phi, \psi)$, $MaxLen(\phi, v)$), while to evaluate some others, one needs to find the occurrences of ϕ in r. For instance, we have defined $MinSupp(\phi, r, f)$ 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 support 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.

Problem setting. Our objective is to formalize the concept of soft-support constraints such that they can be processed efficiently, i.e., as combinations of monotonic and anti-monotonic constraints. This will enable to exploit efficient generic strategies for solving arbitrary combinations of soft-support constraints with other (anti)-monotonic constraints [12,13,14]. This is however challenging. Indeed, relevant similarity constraints are generally neither monotonic nor anti-monotonic [18] while our understanding of soft-occurrences relies on similarity constraints. As a result, preserving the (anti)-monotonicity of soft-support constraints can not be guaranteed. Looking for reasonable conditions under which such properties would be preserved is clearly our main technical issue.

2. Defining Soft-Occurrences and Soft-Support Constraints

The soft support of a pattern ψ is derived from a 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 $|w| = lcs(\phi, x) \geq lcs(\phi', x) = |w'|$.*

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

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 $\phi_3 = accadcdccccdddd$ 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 (Similarity constraint) *Given a reference pattern x and two 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 $C_{sim}(\phi, x, 2, 2)$.*

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

Let us now introduce our new support constraints.

Definition 12 (Soft-support) *If $sOcc(\phi, ins, dels)_1$, ..., $sOcc(\phi, ins, dels)_n$ are the soft-occurrences for ϕ in r, the soft-support of ϕ (denoted $SoftSupp(\phi, r, ins, dels)$) is $|ext(sOcc(\phi, ins, dels)_1, r) \cup \ldots \cup ext(sOcc(\phi, ins, dels)_n), r)|$.*

Definition 13 (Minimum/Maximum soft-support) *Given a user-defined threshold f, the Minimum Soft-support constraint is defined as $MinSoftSupp(\phi, r, f, ins, dels) \equiv SoftSupp(\phi, r, ins, dels) \geq f$. The Maximum Soft-support constraint is defined as $MaxSoftSupp(\phi, r, f, ins, dels) \equiv SoftSupp(\phi, r, ins, dels) \leq f$.*

Example 6 *Continuing Example 2, $SoftSupp(abd, r, 1, 1) = 2$ and $\{bd, abc, abd, ab\}$ are the soft-occurrences of abd on r. $SoftSupp(dc, r, 1, 1) = 5$ and $\{c, dc, d, bc, bd\}$ are the soft-occurrences of pattern dc. Examples of constraints which are satisfied are $MinSoftSupp(dc, r, 4, 1, 1)$ and $MaxSoftSupp(abd, r, 2, 1, 1)$.*

Table 1. Support and soft-support

	$Supp$	$\frac{Supp \times 100\%}{SoftS(1,1)}$	$\frac{Supp \times 100\%}{SoftS(1,2)}$	$\frac{Supp \times 100\%}{SoftS(2,1)}$	$\frac{SoftS(1,1)}{SoftS(1,2)}$	$\frac{SoftS(1,1)}{SoftS(2,1)}$
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
Min val	23	1.53	1.14	0.54	0.45	0.06
Max val	843	100	100	97.6	1	0.99

Proposition 3 *Constraint* $MinSoftSupp(\phi, r, ins, dels)$ *is anti-monotonic (dually, constraint* $MaxSoftSupp(\phi, r, ins, dels)$ *is monotonic) when* $dels \geq ins$.

3. Experimental Validation

We have performed a number of experiments to empirically evaluate the properties of the $MinSoftSupp(\phi, r, ins, dels)$ constraint. To the best of our knowledge, FAVST [14] algorithm is among the best algorithms for mining strings that satisfy arbitrary conjunctions of (anti)-monotonic constraints. As a result, the FAVST framework enables to push constraints $C_{sim}(\phi, x, ins, dels)$, $MinSoftSupp(\phi, r, ins, dels)$ and $MaxSoftSupp(\phi, r, ins, dels)$ (when $dels \geq ins$), and their arbitrary conjunctions with other anti-(monotonic) constraints. We have developed in C our own implementation of the FAVST algorithm. Our experimental validation has been carried out on the KDD Cup 2000 real-world clickstream datasets [21], using a Intel(R) Pentium(R) M 1.69GHz processor (1GB main memory).

To produce time ordered sequences of templates requested for each session, we have extracted attributes "Session ID", "Request Sequence", "Request Template". There are 137 different request templates, i.e., these sequences are strings over an alphabet of 137 symbols. The produced dataset, referred S, contains $234,954$ strings. The shortest string is of length 1 while the largest one is of length $5,487$. We have also extracted the attributes "Session First Request Day Of Week" and "Session First Request Hour Of Day" to split the dataset S into four datasets: SWE for sessions requested on Saturday or Sunday ($47,229$ strings), SWD for sessions requested on workdays ($187,725$ strings), SD for sessions requested from 8 am till 7 pm ($137,593$ strings), and SN for the sessions requested from 7 pm to 8 am ($97,361$ strings).

3.1. Comparative Study of Support, Soft-Support and Degrees of Softness

Solving $MinSoftSupp(\phi, r, ins, dels)$ for ϕ means to solve $C_{sim}(\psi, \phi, ins, dels)$ to find all patterns ψ that are soft-occurrences of ϕ given parameters ins and $dels$. We performed experiments to assess the soft-support w.r.t. support, and the impact of different combinations of parameters ins and $dels$ on resulting "softness". We have computed $SoftSupp(\psi, \mathtt{S}, 1, 1)$, $SoftSupp(\psi, \mathtt{S}, 1, 2)$, and $SoftSupp(\psi, \mathtt{S}, 2, 1)$ for 796 patterns that are the solutions to $IQ_1 \equiv MinSupp(\psi, \mathtt{S}, 0.01\%) \wedge MinLen(\psi, 7) \wedge MaxLen(\psi, 7)$. We took the patterns of the same length so that soft support would not be influenced by variable length but only by ins and $dels$ values. We got 796 solution patterns. Table 1 provides a statistical summary.

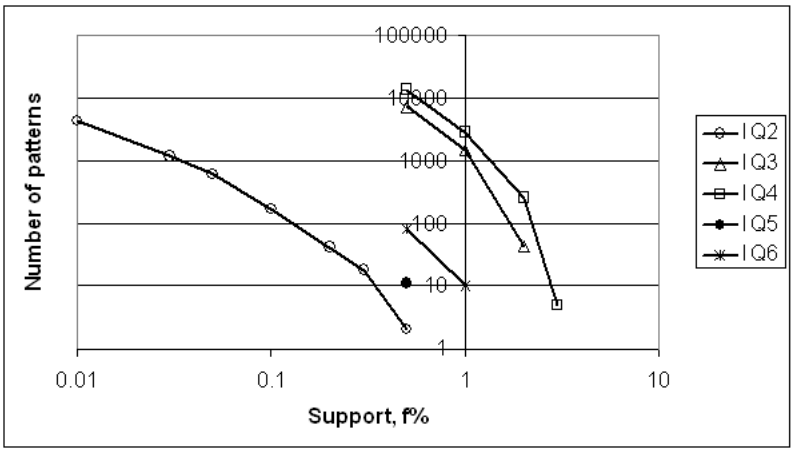

Figure 1. Selectivity of $MinSupp(\phi, r, f)$ and $MinSoftSupp(\phi, r, sf, ins, dels)$

We observe that, in most cases, the support of a pattern is quite small w.r.t. its soft-support. Also, $SoftSupp(\psi, \mathtt{S}, 1, 1)$ tends to be smaller than $SoftSupp(\phi, \mathtt{S}, 1, 2)$ and $SoftSupp(\phi, \mathtt{S}, 2, 1)$. Finally, $SoftSupp(\phi, \mathtt{S}, 1, 2)$ tends to be smaller than $SoftSupp(\phi, \mathtt{S}, 2, 1)$.

3.2. Selectivity of Minimal (Soft)-Support Constraints

To compare the selectivity of $MinSoftSupp(\phi, r, f, ins, dels)$ and $MinSupp(\phi, r, f)$ constraints we computed solutions to

$IQ_2 \equiv MinSupp(\phi, \mathtt{S}, f) \wedge MinLen(\phi, 5) \wedge MaxLen(\phi, 10)$,
$IQ_3 \equiv MinSoftSupp(\phi, \mathtt{S}, f, 1, 1) \wedge MinLen(\phi, 5) \wedge MaxLen(\phi, 10)$
$IQ_4 \equiv MinSoftSupp(\phi, \mathtt{S}, f, 1, 2) \wedge MinLen(\phi, 5) \wedge MaxLen(\phi, 10)$
$IQ_5 \equiv MinSoftSupp(\phi, \mathtt{S}, f, 1, 0) \wedge MinLen(\phi, 5) \wedge MaxLen(\phi, 10)$
$IQ_6 \equiv MinSoftSupp(\phi, \mathtt{S}, f, 1, 2) \wedge MinLen(\phi, 4) \wedge MaxLen(\phi, 10)$

The size of the corresponding solutions is plotted against different f thresholds in the graph given in Figure 1

For $MinSupp(\phi, \mathtt{S}, f)$, we started at $f = 0.01\%$. This is a pretty small value and it appears fair to consider that patterns which do not satisfy this constraint are not interesting. For $MinSoftSupp(\phi, \mathtt{S}, sf, ins, dels)$, we started at $f = 0.5\%$ because of consumed time restrictions (see Section 3.4). For both constraints, we increased f value until the corresponding solution set became empty. In all, there are $767,238$ patterns satisfying $MinLen(\phi, 4) \wedge MaxLen(\phi, 10)$, and $727,873$ patterns satisfying $MinLen(\phi, 5) \wedge MaxLen(\phi, 10)$.

Observe that $MinSupp(\phi, \mathtt{S}, f)$ with even very small support thresholds drastically prunes, while the same support values for $MinSoftSupp(\phi, \mathtt{S}, f, ins, dels)$ are not selective at all. It emphasizes the added value for $MinSoftSupp(\phi, \mathtt{S}, f, ins, dels)$: one might assume that at least 1% of the sessions share common requested templates, and $MinSoftSupp(\phi, \mathtt{S}, 1\%, ins, dels)$ enables to extract these regularities while $MinSupp(\phi, \mathtt{S}, 1\%)$ leads to an empty collection.

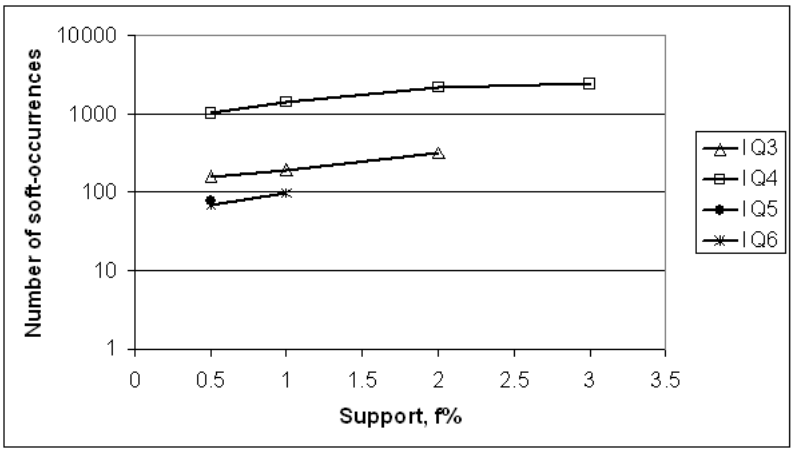

Figure 2. Number of soft-occurrences

Figure 2 plots the mean values of the number of soft-occurrences for patterns that are solutions to IQ_3, IQ_4, IQ_5 and IQ_6. It reveals that, in general, the greater pattern soft-support the more soft-occurrences, or similar patterns, it has. This is however not a theorem.

3.3. Empirical Assessment of Soft-Support Constraint

We strongly believe that softness is needed to find valid regularities or patterns when data or the phenomenon we would like to capture is somehow noisy. We performed experiments to assess this expected added value. In the examples of extracted patterns, i.e., sequences of requested templates, we denote a template by a latin alphabet majuscule letter (see Table 2).

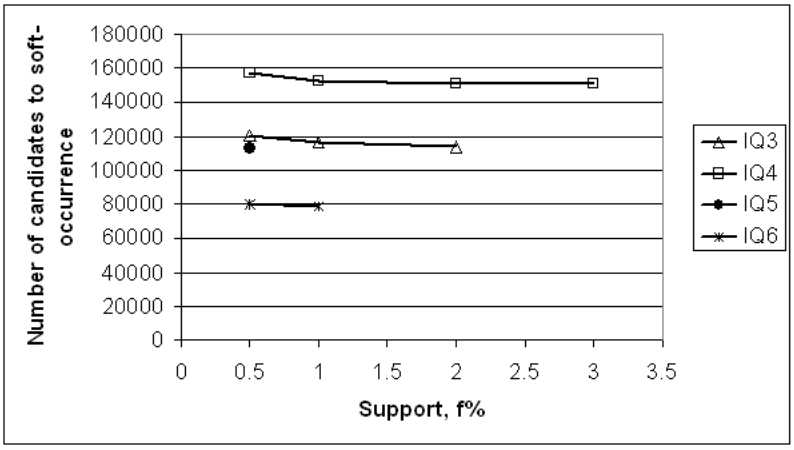

Figure 3. Number of candidates to soft-occurrences

Table 2. Identificators of templates

ID	Template
A	main/home\.jhtm
B	main/departments\.jhtml
C	main/search_results\.jhtml
D	products/productDetailLegwear\.jhtml
E	main/shopping_cart\.jhtml
F	main/login2\.jhtml
G	main/registration\.jhtml
H	main/welcome\.jhtml
I	checkout/expressCheckout\.jhtml
J	main/boutique\.jhtml
K	main/assortment\.jhtml
L	main/vendor\.jhtml
M	main/leg_news\.jhtml
N	products/productDetailLegcare\.jhtml

We have already mentioned that soft-support constraint helps to identify regularities whose exact-support does not appear discriminant (i.e., using exact support, the relevant pattern is blurred among many other other ones). For instance, solving

$IQ_7 \equiv MinSoftSupp(\phi, \mathtt{S}, 2\%, 1, 1) \wedge MinLen(\phi, 5) \wedge MaxLen(\phi, 10)$,

we retrieve the pattern ABCCD having the highest soft-support (2.9%) among patterns whose length is at least 5. Its exact-support is only 0.18%. As a second example, solving

$IQ_8 \equiv MinSoftSupp(\phi, \mathtt{S}, 0.5\%, 1, 1) \wedge MinLen(\phi, 7) \wedge MaxLen(\phi, 10)$,

we find the pattern DCDDDDD of maximal soft support (0.8%) among the patterns whose length is at least 7. Its exact-support is 0.06% only.

Finally, among the solution patterns to

$IQ_9 \equiv MinSoftSupp(\phi, \mathtt{S}, 0.5\%, 1, 1) \wedge ContainsAtMost(\phi, 3, \mathtt{D}) \wedge MinLen(\phi, 7) \wedge MaxLen(\phi, 10)$,

the maximal soft-support is 0.6% for the pattern CDEFGHI while its exact-support is 0.09%.

To get some empirical feedback on both the soft-support and its corresponding exact-support ratio, we evaluated on the query

$IQ_{10} \equiv MinSoftSupp(\phi, \mathtt{S}, 0.5\%, 1, 1) \wedge MinLen(\phi, 5) \wedge MaxLen(\phi, 10)$

Table 3 gives the number of patterns for the intervals of the ratio exact-support/soft-support. We observe that the value of exact-support is not discriminant for the major part of the solutions to IQ_{10}.

Next, we have been looking for sequences of templates that might be specific to workdays (resp. daytime), i.e., frequent among the sessions requested during workdays (resp. daytime) and not frequent among the sessions requested during weekends (resp. nighttime), and vice versa. There were no such sequences of length at least 5 when searching for soft-occurrences with parameters $ins = 1$ and $dels = 1$. We thus relaxed the $MinLen(\phi, 5)$ constraint and restricted the similarity parameters accordingly. It has given the query

Table 3. Number of patterns for exact-support/soft-support intervals

exact-support/soft-support, r	Nb of patterns
$0.0002 \leq r < 0.001$	2131
$0.001 \leq r < 0.01$	3159
$0.01 \leq r < 0.1$	1720
$0.1 \leq r < 0.5$	419
$r \geq 0.5$	5

$$IQ_{11} \equiv MinSoftSupp(\phi, \texttt{SWD}, 0.6\%, 0, 1) \wedge MaxSoftSupp(\phi, \texttt{SWE}, 0.3\%, 0, 1) \wedge MinLen(\phi, 4) \wedge MaxLen(\phi, 10).$$

We found 3 patterns in the solution set for IQ_{11}: JBCC, KBCC, and KKKJ. Similarly, solving

$$IQ_{12} \equiv MinSoftSupp(\phi, \texttt{SD}, 0.6\%, 0, 1) \wedge MaxSoftSupp(\phi, \texttt{SN}, 0.3\%, 0, 1) \wedge MinLen(\phi, 4) \wedge MaxLen(\phi, 10)$$

has given one solution pattern only: DEDE.

In a number of cases, the exact-support and the soft-support of patterns do not coincide. When looking for dataset-characteristic patterns, exact-support can differ significantly, while soft-supports are similar. We extracted patterns satisfying $MinSupp(\phi, r_1, f_1) \wedge MaxSupp(\phi, r_2, f_2)$ constraints and we evaluated their soft-support (with parameters $ins = 1$, $dels = 1$ when $|\phi| \geq 5$, and $ins = 0$, $dels = 1$ otherwise). For instance, the pattern ADDD is a solution pattern to

$$IQ_{13} \equiv MinSupp(\phi, \texttt{SWE}, 0.01\%) \wedge MaxSupp(\phi, \texttt{SWD}, 0.005\%) \wedge MinLen(\phi, 4) \wedge MaxLen(\phi, 10),$$

but its soft-support in SWE is 0.5% and in SWD is 0.4%. Similarly, the pattern AMALA belongs to the solution set of

$$IQ_{14} \equiv MinSupp(\phi, \texttt{SN}, 0.1\%) \wedge MaxSupp(\phi, \texttt{SD}, 0.05\%) \wedge MinLen(\phi, 4) \wedge MaxLen(\phi, 10),$$

but its soft-support in SN is 0.5% and in SD is 0.4%.

Exact-support and soft-support values can be even contradictory. For instance, the patterns DBDBN and ABDBDB belong to the solution set of

$$IQ_{15} \equiv MinSupp(\phi, \texttt{SN}, 0.1\%) \wedge MaxSupp(\phi, \texttt{SD}, 0.07\%) \wedge MinLen(\phi, 4) \wedge MaxLen(\phi, 10),$$

but their soft-supports in SN are 0.3% while they are 0.8% in SD.

These examples emphasize that soft-support constraint are needed to avoid misleading hypothesis on dataset characterization.

3.4. Time Efficiency

The runtime to solve IQ_2, IQ_3, IQ_4, IQ_5 and IQ_6 is given in Figure 4. The fact we get a rather poor time efficiency to compute $MinSoftSupp(\phi, \texttt{S}, f, ins, dels)$ is not surprising. Firstly, $MinSoftSupp(\phi, \texttt{S}, f, ins, dels)$ is far less selective than $MinSupp(\phi, \texttt{S}, f)$ (see Figure 1). Then, the evaluation of $SoftSupp(\phi, r, ins, dels)$ is much more expensive then exact-support counting: even if we push deeply into the extraction phase the (anti)-monotonic conjuncts $MaxDels(\psi, \phi, dels)$ and $MaxIns(\psi, \phi, ins)$ (parts of similarity constraint $C_{sim}(\psi, \phi, ins, dels)$), the number of candidates for

which $C_{sim}(\psi, \phi, ins, dels)$ still has to be evaluated can be huge (e.g., hundreds of thousands, see Figure 3).

The evaluation of $C_{sim}(\psi, \phi, ins, dels)$ is expensive as well: to compute the Longest Common Subsequence we have employed a classical dynamic programming approach of time complexity O(nm) [22]. There is clearly a room for improvements here (see, e.g., [23] for a survey). Also, when $ins = dels$, one can exploit the symmetric property of the underlying similarity relation. In addition, computations can be tuned by choosing dynamically the order of constraints to push.

Let us notice however that, even though soft-support counting may take hours (as we see in Figure 4), it does not prevent from optimizing sequences of inductive queries involving soft-support constraints. Indeed, it is possible to evaluate once the soft-support for patterns in a dataset for some minimal value α (resp. maximal value β) before solving queries which involve $MinSoftSupp(\phi, r, f, ins, dels)$ for $f \geq \alpha$ (resp. $MaxSoftSupp(\phi, r, f, ins, dels)$ for $f \leq \beta$). Also, when computing a conjunction $MinSoftSupp(\phi, r_1, f_1, ins_1, dels_1) \wedge MaxSoftSupp(\phi, r_2, f_2, ins_2, dels_2)$, the evaluation of $MaxSoftSupp(\phi, r_2, f_2, ins_2, dels_2)$ is far less expensive since we prune its search space by the anti-monotonic constraint $MinSoftSupp(\phi, r_1, f_1, ins_1, dels_1)$ and compute $MaxSoftSupp(\phi, r_2, f_2, ins_2, dels_2)$ by starting at the S border for the anti-monotonic constraint $MinSoftSupp(\phi, r_1, f_1, ins_1, dels_1)$ (see, e.g., [24]), ascending towards more general patterns. Figure 5 provides the needed time for solving $MaxSoftSupp(\phi, \texttt{SN}, \beta, ins, dels)$ when $1\% \leq \beta \leq 5\%$ in a conjunction $MinSoftSupp(\phi, \texttt{SD}, \alpha = 0.5\%, ins, dels) \wedge MaxSoftSupp(\phi, \texttt{SN}, \beta, ins, dels)$ for pairs of parameters $ins = 0$, $dels = 1$ and $ins = 1$, $dels = 1$. Notice also that we can store the computed patterns and their soft-supports such that the future extractions can be drastically optimized.

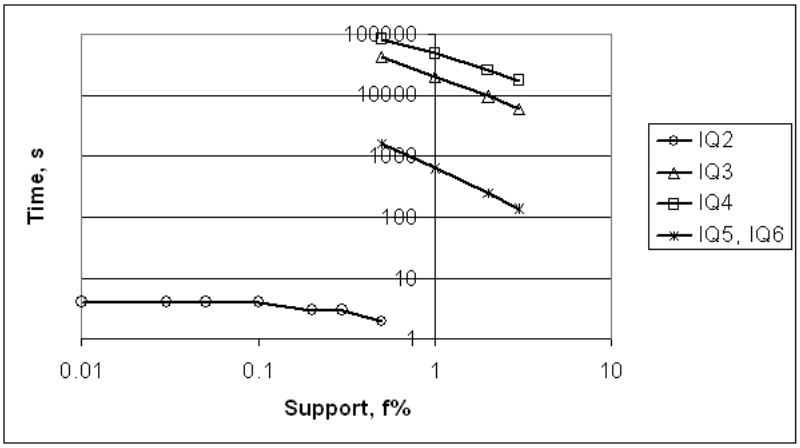

Figure 4. Time efficiency

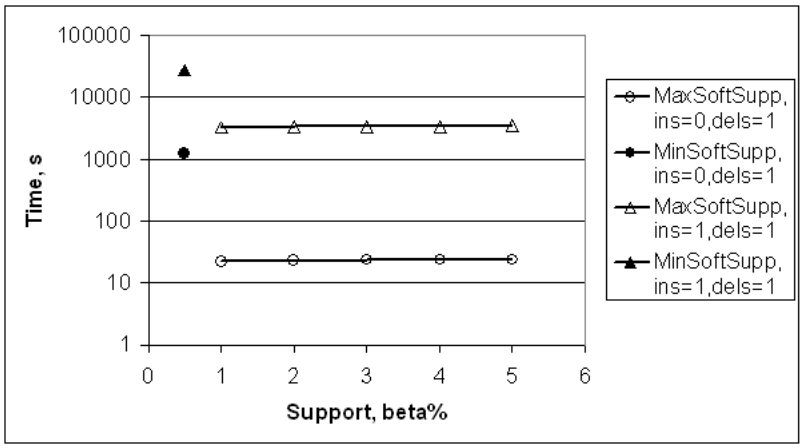

Figure 5. Time efficiency for $MaxSoftSupp(\phi, \texttt{SN}, \beta, ins, dels)$

4. Conclusion

The 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 datasets, i.e., the evaluation of constraints which specify declaratively the desired properties for string patterns. Solving arbitrary combinations of useful primitive constraints by means of generic algorithms is challenging. In this paper, we revisited the popular support constraints when introducing soft occurrences. It might be quite useful when dealing with intrinsically noisy data sets. We formalized an approach to soft-support 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-support constraints with many other user-defined constraints of interest.

Acknowledgements

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

References

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

[2] L. De Raedt. A perspective on inductive databases. *SIGKDD Explorations*, 4(2):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*, pages 1–23. Springer-Verlag LNCS 2682, 2004.

[4] J-F. Boulicaut, L. De Raedt, and H. Mannila (Eds.). *Constraint-based Mining and Inductive Databases*. Springer-Verlag LNAI 3848, 2006.

[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*, pages 27–56. Springer-Verlag LNCS 2682, 2004.

[6] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Proceedings EDBT '96*, pages 3–17. Springer-Verlag, 1996.

[7] F. Masseglia, F. Cathala, and P. Poncelet. The PSP approach for mining sequential patterns. In *Proceedings PKDD'98*, pages 176–184. Springer-Verlag, 1998.

[8] M.J. Zaki. Spade: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1-2):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'00*, pages 355–359. ACM Press, 2000.

[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'01*, pages 215–224. IEEE Computer Society, 2001.

[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*, pages 303–314. Springer-Verlag, 2003.

[12] L. De Raedt, M. Jaeger, S. Dan Lee, and H. Mannila. A theory of inductive query answering. In *Proceedings IEEE ICDM'02*, pages 123–130. IEEE Computer Society, 2002.

[13] S. Dan Lee and L. De Raedt. An algebra for inductive query evaluation. In *Proceedings IEEE ICDM'03*, pages 147–154. IEEE Computer Society, 2003.

[14] S. Dan Lee and L. De Raedt. An efficient algorithm for mining string databases under constraints. In *Proceedings KDID'04*, pages 108–129. Springer-Verlag, 2004.

[15] M. N. Garofalakis, R. Rastogi, and K. Shim. Spirit: Sequential pattern mining with regular expression constraints. In *Proceedings VLDB '99*, pages 223–234. Morgan Kaufmann Publishers Inc., 1999.

[16] J. Pei, J. Han, and W. Wang. Mining sequential patterns with constraints in large databases. In *Proceedings ACM CIKM'02*, pages 18–25. ACM Press, 2002.

[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*, pages 316–320. SIAM, 2003.

[18] I. Mitasiunaite and J.-F. Boulicaut. Looking for monotonicity properties of a similarity constraint on sequences. In *Proceedings of ACM SAC'06, Special Track on Data Mining*, pages 546–552. ACM Press, 2006.

[19] I. Mitasiunaite and J.-F. Boulicaut. About softness for inductive querying on sequence databases. In *Proceedings of DB&IS'06*, pages 77–82. IEEE, 2006.

[20] H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.

[21] R. Kohavi, C. Brodley, B. Frasca, L. Mason, and Z. Zheng. KDD-Cup 2000 organizers' report: Peeling the onion. *SIGKDD Explorations*, 2(2):86–98, 2000. http://www.ecn.purdue.edu/KDDCUP.

[22] D. S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *CACM*, 18(6):341–343, 1975.

[23] A. Apostolico. String editing and longest common subsequences. In *Handbook of Formal Languages*, volume 2 Linear Modeling: Background and Application, pages 361–398. Springer-Verlag, 1997.

[24] T. M. Mitchell. Generalization as search. *Artificial Inteligence*, (18):203–226, 1981.