Why Fuzzy Sequential Patterns can Help Data Summarization: an Application to the INPI Trademark Database

Céline Fiot, Anne Laurent, Maguelonne Teisseire, Benedicte Laurent

Why Fuzzy Sequential Patterns can Help Data Summarization: An Application to the INPI Trade Mark Database

Céline Fiot, Anne Laurent, Maguelonne Teisseire and Bénédicte Laurent

***Abstract*— Mining fuzzy rules is one of the best way to summarize large databases while keeping the information as clear and understandable as possible for the end-user. Several approaches have been proposed to mine such fuzzy rules, in particular to mine fuzzy association rules. However, we argue that it is important to mine rules that convey information about the order. For instance, it is very interesting to convey the idea of time running in rules, which is done in fuzzy sequential patterns. In this paper, we thus focus on fuzzy sequential patterns. We show that mining such rules require to manage many information and we propose algorithms to remain efficient in both memory and computation time. Our proposition is assessed by experiments. In particular, we apply our algorithms on the INPI database which stores almost 2 million trade marks.**

I. Introduction

Mining concise and understandable summaries is one of the main challenges for end-users who face large databases which they cannot exploit. In this framework, we argue that fuzzy logic is the key in order to keep the rules as understandable as possible.

Our interesting approach to discover such a knowledge consists in deriving linguistic summaries. This kind of summary has been extended to fuzzy summaries [1], [2]. Such summarization often requires a user-interaction in order to select interesting and useful knowledge, based on quality and validity measures. Few methods are indeed based on automatic generation of summary. Some of them are based on functional dependencies [3], [4], [5] or association rules [6] mining but few of them have been clearly detailed with implemented algorithms and experiments. An extension of association rules [7], [8] have thus been proposed to automatically generate fuzzy linguistic summaries of type "Most of people who eat *a lot of* candies purchase *few* potato chips", using support and confidence of those rules as validity criteria.

However, this kind of rules does not take the order into account, which leads to less informative rules. We thus aim at building fuzzy rules that describe the evolution of the data over one attribute (for instance the evolution over time). These rules are called fuzzy sequential patterns. Some works have been dealt with fuzzy sequential patterns. However, they did not study the efficient implementation for their algorithms. We focus here on the management of the database and rules in order to mine fuzzy sequential patterns as efficiently as possible, meaning that we try and keep memory and running time as low as possible.

In this paper we present one detailed algorithm, TOTALLY-FUZZY, for finding fuzzy sequential patterns. This algorithm can help in finding frequent sequences and so in generating fuzzy summaries taking into account the ordered aspect of data in summarization. Section II introduces a motivating examplewhich is the real data considered here as an application. Section III presents a short view on fuzzy association rules and fuzzy summarization. Then Section IV presents an introduction to sequential patterns and fuzzy sequential patterns. Section V next details our algorithm TOTALLY-FUZZY run to mine the trade mark database. Section VI presents this database, the implementation of the algorithm and the experiments. Finally, in Section VII, we conclude on the benefits of applying fuzzy sequential patterns to database summarization.

II. Motivating Example

We introduce here a short example that will be used in the next sections to illustrate the definitions and formalism we propose.

Choosing a trade mark is a decisive point for a company. Stakes are financially high and consequences of a mischosen trade mark can be disastrous. For this reason it could be interesting and useful to find out and understand how trade marks are build identifying recurrent patterns in the INPI trade mark database. This database contains around 2 millions of marks registered between 1961 and 2003 in France and statistical methods can only partly help studying trade marks and their building or evolution.

Let us consider a sample of the INPI database (table I), taken from the class 32 of trade marks, which registers marks linked to "beers, spring mineral water, soft drinks, fruit juice, ...". In this data set, each row records one registration with its date and the registered trade mark. The first row represents the first registration for the company C1. It means that it has registered the trade mark "AceCool©" at the date d1. A summarization of this database could be for instance that "Most company register at least one trade mark beginning with a *capital a*".

Using association rules to build fuzzy summaries in our context could help in finding out frequent correlations in letters or morpheme association such that "Most marks containing several *a* also contain few *b*" or "Most marks are componed of few *z* and few *x*". However some information are still hidden such as the ordering of letters in trade marks.

C´eline Fiot, Anne Laurent and Maguelonne Teisseire are with the LIRMM, University of Montpellier II - CNRS, 34392 Montpellier, France (email: {fiot, laurent, teisseire}@lirmm.fr).

B´en´edicte Laurent is with the PRAXILING ICAR Laboratory, University of Montpellier III - CNRS, 34392 Montpellier, France (email: benedicte.laurent@univ-montp3.fr).

TABLE I

SAMPLE OF REGISTRATIONS IN THE INPI DATABASE

ID_comp	Date	Trade Mark
C1	1	AceCool©
	2	Youth Fountain Spring Water
	3	Spring Citron
	4	Equi Cola
C2	1	Asia-cola©
	2	Asiatitude
	3	Plant'Asia Grand Th
C3	1	Bzh' Citrus
	2	Bzh' Orang'
	3	Phare Ouest Beer
	4	Mad' n' Bzh
	5	A l'aise Breizh
	6	Le Finisterien Mad Breizh

For example, we cannot easily find using association rules or fuzzy association rules that "Most marks are componed of more than two syllables" or that "In most trade marks *a* is often followed by several consonants".

That is in what using of sequential patterns and more particularly fuzzy sequential patterns can help. In fact, sequential patterns are an extension of association rules enabling the taking into account of the temporal aspect of ordered data. In addition fuzzy sequential patterns can be used to mine additional knowledge so giving more information than sequential patterns. In the case of sequential patterns we consider the binary presence or absence for example "In most names *a* is often followed by *b*", whereas fuzzy sequential will precise the approximative number of *a* and *b*.

III. SUMMING UP DATABASES USING ASSOCIATION RULES

Fuzzy summaries have been studied for the 1980's. They are linguistically quantified propositions generally written as "Q Y are B", where Q is a linguistic quantifier, Y is a set of objects and B is a property. Such a summary could be for example "Most trade marks are long". They can be used to bring interpretable information from large and multidimensional data sets containing a large amount of valuable knowledge not directly accessible or exploitable for a user. Usually these propositions are partially automatically built from a combination of a priori known sets of quantifiers, objects and properties. They are then returned with a truth degree, computed by the systems for each combination.

On the contrary fuzzy summaries from association rules based approaches are completely automatically built. These methods only compute useful combinations of quantifiers, objects and properties and the truth degree is given by association rules characteristics such as *support*, *confidence* or *lift*.

Fuzzy association rules are an extension to the association rules approach proposed by [9]. They have been proposed to find frequently correlated numerical attributes in purchase databases. One of their aims was in fact to bring linguistical description to decision makers.

Let DB be a database and I the set of attributes or *items* in this database. Each of these attributes i_k will be associated with several fuzzy sets. A fuzzy association rule $(X, A) \rightarrow (Y, B)$ can be interpreted as a proposition in the following form "If X is A then Y is B", where X and Y are *itemsets* (sets of items) and A and B are fuzzy sets respectively associated to X and Y. The satisfiability of rules is determined thanks to two factors: the *support* which gives the significance of the rule and the *confidence* which gives the certainty of it.

Definition 1: The *support* or *significance* factor S of a rule $R : (X, A) \rightarrow (Y, B)$ reflects the number of records containing the itemset $(X, A) \cap (Y, B)$ in the database.

Definition 2: The *confidence* or *certainty* C factor of a rule $R : (X, A) \rightarrow (Y, B)$ measures the likelihood for a record in DB containing (X, A) to also contains (Y, B).

A rule is considered of a certain interest if it is frequent and it has sufficient certainty, that means its support and confidence are above user-defined minimum thresholds $minSupp$ and $minConf$.

From these interesting fuzzy association rules, one can automatically build fuzzy summaries. For example a fuzzy association rule "*word, lot & letter, lot* $\rightarrow$ *figure, few*" given with a support of 65% and a confidence of 90% can be interpretated as "Most of trade marks containing a lot of words and letters contain few figures". However some information is still hidden such as the form of the words. For example, we can not know if trade marks are componed first of a short word then of a long one or if letters are followed or preceeded by figures. That is why a method that mine ordered data could be useful.

IV. FROM SEQUENTIAL PATTERNS TO FUZZY SEQUENTIAL PATTERNS

By contrast to association rule based approaches, sequential pattern algorithms take the temporal aspect of data into account. They are thus well-adapted to ordered or historized data, such as monitoring data, texts or liguistic data. In this section we briefly describe the basic concepts of sequential patterns then fuzzy sequential patterns.

A. Sequential Patterns

Let DB be a set of customer transactions where each transaction T consists of three information: a customer-id, a transaction timestamp and a set of items.

Let $I = \{i_1, i_2, ..., i_m\}$ be a set of items. An *itemset* is a non-empty set of items, denoted by $(i_1 i_2 \dots i_k)$. It is a non-ordered representation. A *sequence* s is a non-empty ordered list of itemsets, denoted by $< s_1 s_2 ... s_p >$. A *n-sequence* is a sequence of n items (or of size n).

Example 1: Let us consider purchases of products $1, 2, 3, 4$, and 5 made by the customer Smith according to the sequence $s =< (1)\ (2\ 3)\ (4)\ (5) >$. It means that all the items of the sequence were bought separately except the

products 2 and 3 which were purchased at the same time. In this example, s is a 5-sequence.

One sequence $< s_1\ s_2 ... s_p >$ is a *subsequence* of another one $<\ s'_1\ s'_2\ ...s'_m >$ if there exist integers $l_1 < l_2 < ... < l_p$ such that $s_1 \subseteq s'_{l_1},\ s_2 \subseteq s'_{l_2}, ...,\ s_p \subseteq s'_{l_p}$.

Example 2: The sequence s' = <(2) (5)> is a subsequence of s because (2) $\subseteq$ (2 3) and (5) $\subseteq$ (5). However, <(2) (3)> is not a subsequence of s.

All transactions from the same customer are grouped together and sorted in increasing order of their timestamp. They are called a data sequence. A customer *supports* a sequence s if it is included into the data sequence of this customer (s is a subsequence of the data sequence). The *support* of a sequence is defined as the percentage of customers supporting s. In order to decide whether a sequence is frequent or not, a minimum support value ($minSupp$) is specified by the user and the sequence is said to be frequent if the condition $supp(s) \geq minSupp$ holds. Given a database of customers transactions the problem of sequential patterns mining is to find all maximal sequences (frequent sequences not included into another frequent sequence) of which the support is greater than a specified threshold (minimum support) [10]. Each of these sequences represents a sequential pattern, also called a maximal frequent sequence.
Note that items are processed using a simple binary evaluation: present or not present. In our case, we know that names are componed of letters and we would like to know how many of them they are, so we have to mine quantitative attributes such as the number of figures or symbols. We could have divided these attributes into crisp intervals but the linguist expert knowledge have driven us to fuzzy intervals and so to fuzzy sequential patterns mining.

B. Fuzzy Sequential Patterns

In order to mine fuzzy sequential patterns, the universe of each quantitative item is partitioned into several fuzzy sets. The notions of item and itemset have been redefined compared to classical sequential patterns, as presented in [11].

Definition 3: A **fuzzy item** is the association of one item and one corresponding fuzzy set. It is denoted by $[x, a]$ where x is the item (also called attribute) and a is the associated fuzzy set.

Example 3: $[length, short]$ is a fuzzy item where $short$ is a fuzzy set defined by a membership function on the quantity universe of the possible values of the item $length$.

Definition 4: A **fuzzy itemset** is a set of fuzzy items. It can be denoted as a pair of sets (set of items, set of fuzzy sets associated to each item) or as a list of fuzzy items.
We use the following notation: $(X, A) = ([x_1, a_1], ..., [x_p, a_p])$ where X is a set of items and A a set of corresponding fuzzy sets and $[x_i, a_i]$ are fuzzy items.

Example 4: $(X, A) = ([length, short][letter, few])$ is a fuzzy itemset and can be also denoted by $((length, letter)(short, few))$.

One fuzzy itemset only contains one fuzzy item related to one single attribute. For example, the fuzzy itemset $([length, short][length, few])$ is not a valid fuzzy itemset, because it contains twice the attribute $length$.

Lastly we define a g-k-sequence.

Definition 5: A g-k-**sequence** $S =< s_1 \cdots s_g >$ is a sequence constituted by g fuzzy itemsets $s = (X, A)$ grouping together k fuzzy items $[x, a]$.

Example 5: The sequence $<([length, short][symbol, lot])$ $([letter, lot]) >$ groups together 3 fuzzy items into 2 itemsets. It is a fuzzy 2-3-sequence.

In the next sections of this article we use the following notations: let $\mathcal{C}$ represent the set of customers and $\mathcal{T}_c$ the set of transactions for one customer c. Let $\mathcal{I}$ be the set of attributes and $t[i]$ the value of attribute i in transaction t. In the example presented Section III, table I, $\mathcal{T}_c = \{$C1, C2, C3$\}$.

Let consider the database given table I. Each trade mark is parsed to get the number of words, signs, letters, figures and symbols. For example, Asia-cola© contains one word, 10 signs, 8 letters and 2 symbols. This translation is given by Table II. As no trade mark contains figures, this column is not represented.

TABLE II
RESULTS OF TABLE I PARSING

ID_comp	Reg_date	#word	#sign	#letter	#symbol
C1	1	1	8	7	1
	2	4	24	24	0
	3	2	12	12	0
	4	2	8	8	0
C2	1	1	10	8	2
	2	1	10	10	0
	3	3	18	16	2
C3	1	2	10	9	1
	2	2	10	8	2
	3	3	14	14	0
	4	3	9	7	2
	5	3	13	12	1
	6	4	22	22	0

Next step consists in dividing these quantitative attributes into fuzzy sets thanks to the linguist expert knowledge. The fuzzy sets, for each item, are given Fig. 1.

Finally from these membership functions we get the membership degrees for each attribute and fuzzy set. We obtain a membership database from which is extracted the registrations for C3, described Table III.

Table III is used below to illustrate the upcoming definitions and algorithms for the sequence mS =<*([letter, very few]) ([word, few])*>, "Registrations first contain names with very few letters then trade names are made of few words".

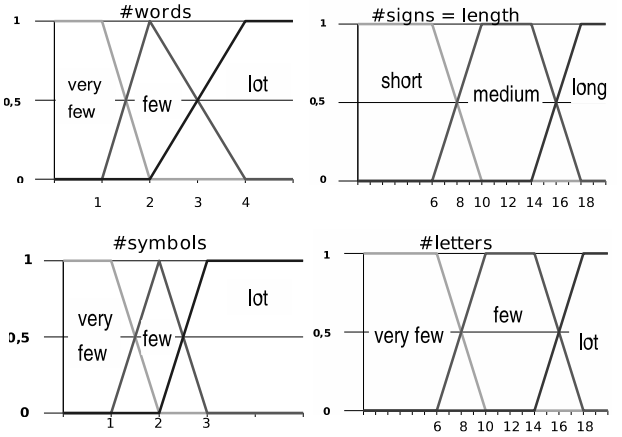

Fig. 1. Fuzzy sets

TABLE III

MEMBERSHIP DEGREES FOR C3

	Fuzzy items									
	#word		length			#letter			#symb	
D.	f.	L.	sh.	m.	L.	v.f.	f.	L.	v.f.	f.
d1	1			1		0.25	0.75		1	
d2	1			1		**0.5**	0.5			1
d3	0.5	**0.5**			1		1			
d4	0.5	0.5	0.25	0.75		0.75	0.25			1
d5	0.5	0.5		1			1		1	
d6		1			1			1		

C. Three Ways for Support Computation

T.-P. Hong and al [12] and then Y.-C. Hu and al [13] have presented two proposals of fuzzy sequential patterns mining approaches. We have extended their initial definitions of fuzzy support to give the user three levels of fuzzification, which could be used to cope with different problems.

The *support*[1] *of a fuzzy itemset* is computed as the proportion of customers supporting it. However the cardinality of a fuzzy set depends on the counting method. We transpose here three of those technics in the framework of fuzzy sequential patterns and we propose three definitions for the fuzzy support:

• SPEEDYFUZZY is based on the count "supports / does not support". Computing the support of a fuzzy itemset consists in counting all the elements for which the membership degree is not null:

$$S_{SF}(c,(X,A)) = \begin{cases} 1 & \text{if } \exists t \in \mathcal{T}_c | \forall [x,a] \in (X,A),\ \mu_a\big(t[x]\big) > 0 \\ 0 & \text{else} \end{cases} \quad (1)$$

Example 6: With SPEEDYFUZZY, customer 1 supports the sequence mS since transactions are found containing the regarded itemsets with a membership degree greater than zero, underlined fuzzy items in Table III.

• MINIFUZZY is based on a thresholded count, so it only keeps the elements for which the membership degree is greater than a given threshold. This method increments the number of customers supporting the fuzzy itemset only when each item of the candidate sequence has a membership degree greater than a specified threshold in the data sequence of the customer:

$$S_{MF}(c,(X,A)) = \begin{cases} 1 & \text{if } \exists t \in \mathcal{T}_c | \forall [x,a] \in (X,A),\ \mu_a\big(t[x]\big) > \omega \\ 0 & \text{else} \end{cases} \quad (2)$$

Example 7: With MINIFUZZY, customer 1 supports the sequence mS since a succession of transactions is found containing the items with a membership degree greater than the threshold (ω=0.49), boldfaced in Table III.

• TOTALLYFUZZY carries out a thresholded Σ-count. The support is computed by a weighted sum of the membership degrees greater than a given threshold. In this approach the importance of each fuzzy itemset in the data sequence is taken into account in the support computation. To do so the threshold membership function α is defined as:

$$\alpha_a\big(t[x]\big) = \begin{cases} \mu_a\big(t[x]\big) & \text{if } \mu_a\big(t[x]\big) > \omega \\ 0 & \text{else} \end{cases} \quad (3)$$

The support counting formula becomes:

$$S_{TF}(c,(X,A)) = \underline{\perp}_{j=1}^{\theta_c} \overline{\top}_{[x,a]\in(X,A)} \Big[\alpha_a\big(t_j[x]\big)\Big] \quad (4)$$

where $\overline{\top}$ and $\underline{\perp}$ are the generalized t-norm and t-conorm opertators.
Note that the Σ-count is a thresholded Σ-count, with a threshold ω=0.

Example 8: With TOTALLYFUZZY, customer 1 supports the sequence mS if a following of transactions is found containing the fuzzy items of the sequence, with a membership degree greater than the threshold ω (0.49). The best value for the sequence is kept, items twice underlined in Table III.

The **support of a fuzzy sequence** is computed as the ratio of the number of customers supporting this fuzzy sequence compared to the total number of customers in the database:

$$FSupp_{(X,A)} = \frac{\sum_{c\in\mathcal{C}} \Big[S(c,gS)\Big]\Big]}{|\mathcal{C}|} \quad (5)$$

where the support degree $S(c,gS)$ indicates if the customer c supports the fuzzy sequence gS. This support degree is computed using the algorithms SPEEDYFUZZY, MINIFUZZY and TOTALLYFUZZY. In this paper we only describe TOTALLYFUZZY which has been used for experiments on the INPI Trade Mark Database presented Section VI.

V. TOTALLYFUZZY: AN ALGORITHM TO MINE FUZZY SEQUENTIAL PATTERNS

TOTALLYFUZZY implements the fuzzy support computation using a thresholded Σ-count to calculate the number of customers supporting a sequence. In this section, we present the support calculation carried out by this algorithm and the overall algorithm which extracts the fuzzy sequential patterns.

A. A Trade-Off between Space and Computational Complexity

When considering classical sequential patterns, a customer supports or not a sequence. So as soon as the sequence is found within a customer's transactions, the scan of the database can stop. On the contrary, as TOTALLYFUZZY computes a thresholded Σ-count, for each customer and each

[1]Note that the support of a fuzzy item, itemset or sequence is not the support a fuzzy set (cardinality of the crisp subset of elements having a nonzero membership grades [14]), but rather the frequency of this fuzzy item, itemset or sequence in the database.

sequence, the best membership degree must be considered. This degree is computed as the aggregation of the itemset supports. The order of the fuzzy items must also be taken into account. That leads to an exhaustive scanning of the transaction set, as performed for association rule mining. Let us consider for instance the table III. If we look for the same sequence as above, *<([letter, very few]) ([word, few])>*, the first occurrence of this sequence is the one underlined, supported by transaction d1 then d3. However the best occurrence of this sequence is given by transaction d4 followed by d6, underlined twice.

A naive approach could be that for each candidate k-sequence (likely frequent sequence) we scan all the database to find its support, it means at most n^k scans of the database if n is the number of frequent items. The only structure kept in memory would so be the list of candidate sequence. However this would be very unefficient, as the computational time would explode.

To reduce this number of scans, we have defined a data structure enabling us to find all the representation of all the k-sequences in only k scans of the database. The computational time is so lower but the used memory space is increased. However some optimizations have been implemented to bound this spatial complexity.

So we present here an efficient implementation based on the notion of *path*. One path corresponds to a possible instantiation of the candidate sequence itemsets into the customer's transaction set. Several paths may be initialized for one customer (here one registrating company). For the global support computation we only keep the complete one having the best degree.

Definition 6: One *path* is a triplet containing the already found sequence seq, the currently searched itemset $curIS$ (coming next in the candidate sequence) and the current membership degree $curDeg$.

The next subsection presents an illustration of TOTALLYFUZZY working, then Subsection V-C details the functions for the support calculation. Finally Subsection V-D presents the overall algorithm to extract the fuzzy sequential patterns.

B. Computing a sequence support: an Illustration

As an illustration we run TOTALLYFUZZY to compute the support of the candidate sequence *<([letters, very few]) ([words, few])>* for customer 3 from table III, with a threshold ω= 0.2. It is summarized in table IV.

First the process is initialized by creating one first empty path $pth1 = (\varnothing, ([letter, veryfew]), 0)$ (Table IV, row 1). Then it begins scanning the first record of customer 3. The currently searched itemset $curIS$ is found with a degree *d1[letter, very few]=0.25*$geq\omega$. The path $pth1$ is so updated with $pth1.seq \leftarrow < ([letter, veryfew]) >$, $pth1.curIS \leftarrow ([words, few])$ and $pth1.seq \leftarrow 0.25$ (Table IV, row 2).

Then transaction d2 contains the first sequence of the candidate sequence g-S, $([letter, veryfew])$. So a new path is created, $pth2 \leftarrow (< ([letter, veryfew]) >, ([words, few]), 0.5)$. As this transaction does not contain the next itemset for $pth1$, the scan of the dataset should continue. Before that, an optimization is carried out to avoid using to much space in memory: for two paths at the same step, only the one with the best $curDeg$ value is kept. Here we have $pth1.curDeg < pth2.curDeg$ so $pth1$ is deleted and TOTALLYFUZZY keeps on scanning the next transaction with $pth2$ (Table IV, row 3).

Transaction d3 is then checked. It contains $pth2.curIS = ([words, few])$. It is so modified to $pth2 \leftarrow (< ([letter, veryfew]) ([words, few]) >, \varnothing, 0.5)$. This path is closed since it contains all the itemsets of g-S. However we keep the possibility to improve this solution. So, before having modified $pth2$, it is copied into $pth3$. At this step, we have two paths: $pth2$, closed with a support degree of 0.5 and $pth3 = (< ([letter, veryfew]) >, ([words, few]), 0.5)$ (Table IV, row 4).

Transaction d4 is then checked. $pth3.curIS$ is found so it is copied, modified and then closed with a degree of 0.5. As d4 also contains the first itemset of g-S, a new path is created. At this point we have four paths (Table IV, row 5). The $Optimize$ function deletes for each step in the sequence the path with the lower degree. It means $pth3$ and $pth4$. Scanning then continues. At d5, $pth5$ is copied into $pth6$, modified and closed with a support degree of 0.63, so $pth2$ is deleted. Finally, at d6, $pth6$ is updated and closed with a degree of 0.87. This is the path kept for customer 3.

TABLE IV
ILLUSTRATION OF TOTALLYFUZZY ON C3

	$pth1 : (\varnothing, ([letter, veryfew]), 0)$
After d1	$pth1 : (< ([letter, veryfew]) >, ([words, few]), 0.25)$
After d2	$pth1 : (< ([letter, veryfew]) >, ([words, few]), 0.25)$ $pth2 : (< ([letter, veryfew]) >, ([words, few]), 0.5)$
Opt.	$pth1$ is deleted
After d3	$pth2 : (< ([letter, veryfew]) ([words, few]) >, \varnothing, 0.5)$, closed $pth3 : (< ([letter, veryfew]) >, ([words, few]), 0.5)$
After d4	$pth2 : (< ([letter, veryfew]) ([words, few]) >, \varnothing, 0.5)$, closed $pth3 : (< ([letter, veryfew]) ([words, few]) >, \varnothing, 0.5)$, closed $pth4 : (< ([letter, veryfew]) >, ([words, few]), 0.5)$ $pth5 : (< ([letter, veryfew]) >, ([words, few]), 0.75)$
Opt.	$pth3$ and $pth4$ are deleted
After d5	$pth2 : (< ([letter, veryfew]) ([words, few]) >, \varnothing, 0.5)$, closed $pth5 : (< ([letter, veryfew]) > ([words, few]) >, \varnothing, 0.63)$, closed $pth6 : (< ([letter, veryfew]) >, ([words, few]), 0.75)$
Opt.	$pth2$ is deleted
After d6	$pth5 : (< ([letter, veryfew]) > ([words, few]) >, \varnothing, 0.63)$, closed $pth6 : (< ([letter, veryfew]) > ([words, few]) >, \varnothing, 0.87)$, closed
Opt.	$pth5$ is deleted
C3 deg.	**0.87**

C. Computing a sequence support: the Algorithms

The algorithm TOTALLYFUZZY uses the function *FindTotallySeq* to carry out an ordered scanning in one customer's transaction set.

When the first itemset of the sequence is found, one path is created with the itemset support. The next transactions are checked to find either the following part of the sequence or once again the beginning of the sequence or an improvement of the paths, already created. All the possible paths are thus completed step-by-step at each transaction. The support degree of the best path for the whole sequence is then returned. The $Update$ function allows the update of each path and closes the complete ones. The $Optimize$ algorithm, not

presented in this paper, enables to delete unnecessary paths. The function $CalcTotallySupport$ computes the support for one candidate sequence by adding for each customer the aggregation value of the optimal path for this sequence.

ALGORITHM **1** - CalcTotallySupport

Input: gS, candidate g-k-sequence ;
Ouput: $FSupp$ fuzzy support for the sequence gS ;

$FSupp, nbSupp, m \leftarrow 0$;
For each customer client $c \in \mathcal{C}$ **do**
  $m \leftarrow$ FindTotallySeq(g-S, $\mathcal{T}_c$);
  $nbSupp$ += m ;
End For
$FSupp \leftarrow nbSupp/\Gamma$;
return $FSupp$;

ALGORITHM **2** - Update

Input: pth, path to update

pth.seq $\leftarrow$ pth.seq $\cup pth$.curIS ;
If (pth.curIS.next $\neq \varnothing$) **Then**
  pth.curIS $\leftarrow$ pth.curIS.next ;
Else
  Close(pth);
End If
Optimize(pth) ;

ALGORITHM **3** - FindTotallySeq

Input: g-S, candidate g-k-sequence;
T, transaction set to run
Ouput: m, support degree of the best g-S representation instanciated in the transaction set T

$Paths$: **list of paths** $\rightarrow$ **(seq, curIS, curDeg)**
$Paths \leftarrow$Path($\varnothing$, gS.first, 0)
For each transaction $t \in T$ **do**
  For each path $pth \in Paths$, not updated at t **do**
    If (pth not closed) **Then**
      If (pth.curIS $\in t$) **Then**
        pth.curDeg $\leftarrow$ pth.curDeg | $\overline{\top}_{[x,a] \in pth.\text{curIS}} \alpha_a(t[x])$;
        Update(pth);
      End If
    End If
    For j **from** 2 **to** pth.curIS -1 **do**
      [Search for an improvement of the current path]
      If ((gS.get(j) $\in t$) & ($\overline{\top}_{[x,a] \in gS.get(j)} \alpha_a(t[x]) > pth$.curDeg[j])) **Then**
        nCurIS $\leftarrow$ gS.get(j) ;
        For i **from** 1 **to** j-1 **do**
          nSeq $\leftarrow$ nSeq | gS.get(i) ;
          nCurDeg $\leftarrow$ nCurDeg | pth.curDeg[i];
        End For
        nCurDeg $\leftarrow$ nCurDeg | $\overline{\top}_{[x,a] \in gS.get(j)} \alpha_a(t[x])$;
        $Paths \leftarrow Paths \cup$ Update((nSeq, nCurIS, nCurDeg)) ;
      End If
    End For
  End For
  If ((gS.first $\in t$) & (not(FirstPass))) **Then**
    $pth \leftarrow$Path($\varnothing$, gS.first, $\overline{\top}_{[x,a] \in gS.first} \alpha_a(t[x])$);
    $Paths \leftarrow Paths \cup pth$;
    Update(pth) ;
  End If
  Paths.Optimize() ; *[deletion of less pertinent paths]*
End For
For each path $pth \in Paths$ **do**
  If (pth not closed) **Then**
    Cut(pth); *[deletion of paths not containing the whole sequence]*
  End If
End For
$pth \leftarrow Paths.first$; *[$Paths$ only contains the best complete path]*
$m \leftarrow \odot(pth.\text{curDeg})$; *[Aggregation to return the support degree]*
return m;

D. Mining Frequent Sequences

Our approach extends the level-wise approach generate-prune within the context of sequential patterns and more particularly uses the prefix-tree structure described in [15], in order to improve the support computation process. This structure is used to store both the candidate sequences and the frequent ones. The tree on figure 2 represents the sequences *<([letter, very few][figure, few])([figure, few])>*, *<([letter, very few])([length, medium])>* and also the subsequences *<([length, medium])>* and *<([figure, few])([figure, few])>*.

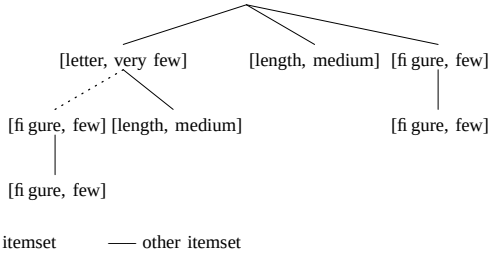

Fig. 2. Storage of sequences as a prefix-tree

The overall algorithm PSP-Totally is presented by the Algorithm 4. First the fuzzy support of all fuzzy items is computed and only the items with a support greater than $minSupp$ are stored as frequent ones of size 1. Then, while a candidate generation still is possible, the $generate$ function builds the candidate sequences of size k from the frequent sequences of size k-*1*. Then a scan over the database is run and supports are computed thanks to $CalcTotallySupp$. Finally all infrequent sequences of size k are pruned and the process continues. At the end of the process we obtain a Prefix-tree containing all the frequent sequences of the database with their support at each leaf of the tree.

ALGORITHM **4** - PSP-Totally

Input: DB, a database, $minSupp$
Ouput: PT, the prefix-tree of frequent sequences

find-1-Frequent() ;
While (#candidate < 0) **do**
  $generate(PT$.depth+1) ;
  For each sequence s in PT **do**
    $CalcTotallySupp(s)$;
  End For
  $prune(PT)$;
End While
return (PT) ;

As for PSP, the overall computational complexity of PSP-Totally only depends on the length of the candidate sequences, and so of the support calculation. The computation of the $generate$ and $prune$ functions are indeed insignificant in comparison with the one of $CalcTotallySupp$.

VI. EXPERIMENTS

In this paper we present one of our mining experiments, looking for sequences of words in registered trade marks (intra-name patterns). Further experiments, detailed here but

not finished yet, will consist in looking for evolution in the registration by registering company (inter-name patterns).

A. The INPI Trade Mark Database

The INPI is the French *National Institute for the Industrial Property*. It aims at managing and registering industrial title deeds, communicating information on industrial property and devising and adapting french industrial property right.
The INPI trade mark database registers around 2 millions of names registered between 1961 and 2004. These trade marks have been registered in one or several categories corresponding to industrial fields such as “toys, games” or “clothing, shoes, hat industry”. Each record is componed of several attributes, for instance: id_number, trade mark, registration date, registrating company, registrating class.

The global goal of analysing such a database is, for the linguist expert, to understand how trade marks act on consumers. One step towards this aim is to understand building mecanisms of those marks from a linguistic point of view, through different axes: morpheme analysis (e.g. number of letters, ...), phonetic, graphic (presence of numbers or special characters (e.g. !, @, ...). Some of these analysis can be made thanks to statistical methods such as plotting histogramms or graphs to look at evolution or distribution of registration number, but it is hard to retrieve relevant information in such a huge database when facing so many indicators.

Therefore data mining tools can be really useful to sum this trade mark database up. Then this data set contains many quantitative information such as letter number or word number for a same trade mark. So it requires to use Fuzzy Sets Theory based mining approaches in order to help the linguist expert, without choosing crisp thresholds or intervals. For example, how can we decide that a word is long? Must it be longer than 10 letters? or 12? Moreover we require to analyse series and connections in those names and so a method allowing sequence mining.
For all these reasons we have chosen to mine fuzzy sequential patterns in the INPI trade mark database.

B. Data Translation for Intra-name Pattern Mining

These experiments aim at discovering word form in registered trade names. Summaries would be described for example as “Almost one third of registered names are first componed of one part containing a lot of letters then of a part with few figures and then of a part with few punctuation signs and few letters”. These proposition would be built from a sequence *<([letter, lot])([figure, few])([punct_symb, few],[letter, few])>*.
In this case, we can both consider trade marks made of one or several words. In fact a name only componed of one word will appear as a fuzzy itemset instead of a sequence for a name made of several words. In order to mine this kind of patterns, the INPI database is converted into the format [ID_TM, #WORD, FUZZY ITEM, DEGREE], where fuzzy items are described on figure 3.

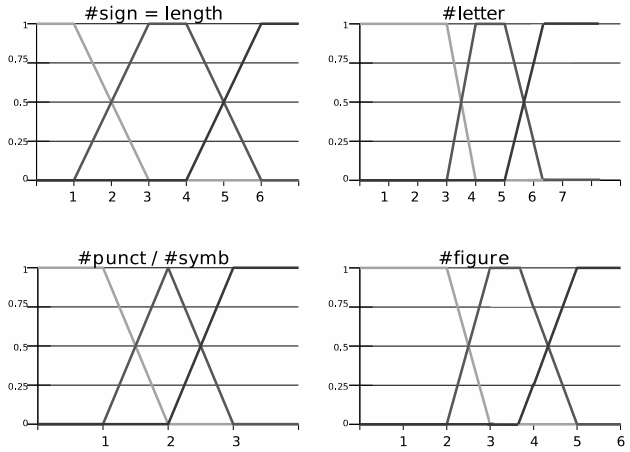

Fig. 3. Fuzzy sets for fuzzy items for intra-name pattern mining

TABLE V
DATABASE TRANSLATION FOR INTRA-NAME PATTERN MINING

Mining formalism		Trade mark database
customer	↔	a trade mark
date	↔	number of the word in the trade mark
fuzzy item	↔	# of signs, letters, figures, punctuations or symbols in this word

C. From Fuzzy Sequential Patterns to Fuzzy Summarization

First we have chosen to begin with the mining into classes that seemed to have an atypical behavior through statistical analysis. The class number 38, registering trade marks linked to the field of “Telecommunication”, was given to contain more symbols and punctuations than the other trade marks. Our summarization shows that even if these classes contains a greater proportion of symbols, it is not a characteristic point. We detail here the results obtained by mining this dataset of more than 480,000 trade marks componed of one or several words.

First we have mined for a general description of this class. For a minimum support 70% ($minSupp$ = 0.7), we obtain the sequential pattern <([#sign, lot][#letter, lot])> (76.8%), which has been translated into the statement “Most of trade marks contain at least one word which contain a lot of signs and a lot of letters”. Then, decreasing the $minSupp$ value, we have find out that “Almost one quarter of trade marks contain a least two words, both having a lot of signs and a lot of letters” (<([#sign, lot][#letter, lot]) ([#sign, lot][#letter, lot])> (24.7%)).

During a second step, we have divided the class 38 into four databases, according to the number of words by trade marks: one word (74,977 names), exactly two words (101,440 names), at least two words (303,742 names) and at least three words (228,940 names).

This part of mining has aimed at discovering more specific knowledge about trade marks in the class 38. What we can give as a summarization for this category of trade marks is that: “Most trade marks componed of one word

contains a lot of signs and a lot of letters (<([#sign, lot][#letter, lot])> (74.1%))" and "Only few trade marks made of one word contains a medium amount of signs or letters (<([#sign, medium])> (14.6%), <([#letter, medium])> (15.1%))". Those two summaries could have been found using summarization based on fuzzy association rules mining, whereas the following ones expressing sequences between words in trade marks can only be obtained thanks to fuzzy sequential pattern summarization.

Mining in trade marks componed of two words reveals that there composition is quite various. We have indeed discovered that even "Most of them contain at least one word with a lot of signs" (<([#sign, lot])> (81%)), "Only one third of them contain two words made of a lot of signs, one preceeding the other with a lot of letters" (<([#sign, lot]) ([#sign, lot][#letter, lot])> (33%)).
We have then to decrease the support to 0.1 to learn more about the intra-structure of these trade marks. We can see that "A small part of the category 38 componed of two words is componed of one word with a medium amount of signs and few letters preceeding a second one with both a lot of signs and letters" (<([#sign, medium][#letter, few]) ([#sign, lot][#letter, lot])> (10.5%)) and that "A small part of these trade marks is componed of one word with a medium amount of signs and letters followed by a second one with both a lot of signs and letters" (<([#sign, medium][#letter, medium]) ([#sign, lot][#letter, lot])> (13.2%)).

Finally we give some information obtained concerning the trade marks of class 38 componed of at least three words. We have found out that "Almost two third are componed of two words with both a lot of signs and letters" (<([#sign, lot][#letter, lot]) ([#sign, lot][#letter, lot])> (62%)); it is completed by decreasing the support: "Almost one half of these trade marks contain three words with a lot of signs, the first one with a lot of letters, one following then and preceeding a third one with a lot of letters" (<([#sign, lot][#letter, lot]) ([#sign, lot]) ([#sign, lot][#letter, lot])> (43.5%)), whereas "Almost one half of these trade marks contain three words with a lot of signs, the first and last ones with a lot of signs and letters, the second one with few letters or a medium amount of signs" (<([#sign, lot][#letter, lot]) ([#letter, few]) ([#sign, lot][#letter, lot])> (42%), <([#sign, lot][#letter, lot]) ([#sign, medium]) ([#sign, lot][#letter, lot])> (42%)).

It appears that trade marks containing figures, symbols or punctuations in this category are rare and infrequent.

D. Mining for Trends in Trade Mark Morphology

The currently carried out experiments aim at mining for evolution in trade mark registration for a same registering company. For example, we could find that "Few registrating companies have first registered short names containing few figures and then short names containing a lot of punctuation symbols". This would be represented by a sequence *<([sign, few],[figure, few])([sign, few],[punct_symb, lot])>*. This requires to preprocess the database and to translate it into the required format for fuzzy sequential pattern mining. A record in the mined database will be a tuple [REGISTERING COMPANY, YEAR, FUZZY ITEM, DEGREE]. Those fuzzy items are the quantitative attributes number of signs, number of words, number of figures, number of punctuation symbols, number of letters and number of special characters, associated to a fuzzy set. These fuzzy sets for those attributes have been built from a linguist expert knowledge.
The results of these experiments will be used to extend the method proposed in [16] to discover trends in text databases and will be then completed applying temporal constraints, as presented in [17].

VII. CONCLUSION

In this paper, we have presented a method to summarize quantitative historized or ordered database with fuzzy sequential patterns. The efficient algorithm TOTALLYFUZZY is detailed with the data structure used to reduce its computational complexity. We have implemented this algorithm and applied it to summarize the INPI Trade Mark database. The results of these experiments show the summarization given to describe the intra-structure of trade marks. Further experiments are currently in progress to sum up the evolution of trade mark structure during the last fourty years.

ACKNOWLEDGMENT

The authors would like to thank Mr. St´ephane Sanchez and Mr. Federico Del Razo Lopez for their help in the management and preprocessing of the INPI database.

REFERENCES

[1] J. Kacprzyk, "Fuzzy logic with linguistic quantifiers: A tool for better modeling of human evidence aggregation processes?" *Fuzzy Sets in Psychology*, pp. 233–263, 1988.

[2] J. Kacprzyk, R. Yager, and S. Zadrozny, "A Fuzzy Logic Based Approach to Linguistic Summaries of Databases," *Applied Mathematics and Computer Science*, vol. 10, pp. 813–834, 2000.

[3] P. Bosc, L. Lietard, and O. Pivert, "Extended functional dependencies as a basis for linguistic summaries," in *2nd Eur. Symp. on Principles of Data Mining and Knowledge Discovery*, 1998, pp. 255–263.

[4] D. Dubois and H. Prade, "Fuzzy sets in data summaries – outline of a new approach," in *8th Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, 2000.

[5] J. C. Cubero, J. M. Medina, O. Pons, and M. A. Vila, "Data summarization in relational databases through fuzzy dependencies," *Inf. Sci.*, vol. 121, no. 3-4, pp. 233–270, 1999.

[6] J. Kacprzyk and S. Zadrozny, "Fuzzy linguistic summaries via association rules," *Data Mining and Computational Intelligence*, pp. 115–139, 2001.

[7] A. Fu, M. Wong, S. Sze, W. Wong, and W. Yu, "Finding Fuzzy Sets for the Mining of Fuzzy Association Rules for Numerical Attributes," in *1st Int. Symp. on Intelligent Data Engineering and Learning*, 1998, pp. 263–268.

[8] C. M. Kuok, A. W.-C. Fu, and M. H. Wong, "Mining Fuzzy Association Rules in Databases," *SIGMOD Record*, vol. 27, no. 1, pp. 41–46, 1998.

[9] R. Agrawal, T. Imielinski, and A. N. Swami, "Mining Association Rules between Sets of Items in Large Databases," in *Int. Conf. on Management of Data*, 1993, pp. 207–216.

[10] R. Agrawal and R. Srikant, "Mining Sequential Patterns," in *11th Int. Conf. on Data Engineering*. Taipei, Taiwan: IEEE Computer Society Press, 1995, pp. 3–14.

[11] C. Fiot, A. Laurent, and M. Teisseire, "Motifs squentiels fbus : un peu, beaucoup, passionnment," in *5mes journes d'Extraction et Gestion des Connaissances*, 2005, pp. 507–518.

[12] T. Hong, K. Lin, and S. Wang, "Mining Fuzzy Sequential Patterns from Multiple-Items Transactions," in *Joint 9th IFSA World Congress and 20th NAFIPS Int. Conf.*, 2001, pp. 1317–1321.
[13] R.-S. Chen, G.-H. Tzeng, C.-C. Chen, and Y.-C. Hu, "Discovery of Fuzzy Sequential Patterns for Fuzzy Partitions in Quantitative Attributes," in *ACS/IEEE Int. Conf. on Computer Systems and Applications*, 2001, pp. 144–150.
[14] A. Kaufmann, "Introduction to the theory of fuzzy subsets," 1973.
[15] F. Masseglia, F. Cathala, and P. Poncelet, "The PSP Approach for Mining Sequential Patterns," in *2nd Eur. Symp. on Principles of Data Mining and Knowledge Discovery*, 1998, pp. 176–184.
[16] B. Lent, R. Agrawal, and R. Srikant, "Discovering trends in text databases," in *3rd Int. Conf. Knowledge Discovery and Data Mining*, 1997, pp. 227–230.
[17] C. Fiot, A. Laurent, and M. Teisseire, "Des motifs squentiels gnraliss aux contraintes de temps tendues," in *6mes journes d'Extraction et Gestion des Connaissances*, 2006, pp. 603–614.