

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

Céline Fiot, Anne Laurent, Maguelonne Teisseire and Bénédicte Laurent

Abstract—Mining fuzzy rules is one of the best ways to summarize large databases while keeping 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 requires to manage a lot of information and we propose algorithms to remain efficient in both memory use and computation time. Our proposition is assessed by experiments. Particularly, we apply our algorithms on the INPI database which stores almost 2 million trademarks.

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 and summaries as understandable as possible.

The common approach to express such a knowledge consists in deriving linguistic summaries [1], which have been extended to fuzzy summaries [2], [3]. 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 [4], [5], [6] or association rule mining [7] but few of them have been clearly detailed with implemented algorithms and experiments. An extension of association rules [8], [9] has thus been proposed to automatically generate fuzzy linguistic summaries of type “Most of people eating *a lot of* candies purchase *few* potato chips”, using support and confidence of those rules as validity criteria.

However, such rules do 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 evolution over time). These rules are called sequential patterns. Some works have dealt with fuzzy sequential patterns. However, they did not study the implementation efficiency for their algorithms. We focus here on database and rules management in order to mine fuzzy

sequential patterns as efficiently as possible, meaning that we try to keep memory and runtime as low as possible.

In this paper we present one detailed algorithm, `TOTALLYFUZZY`, 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 example, describing the real data here considered 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 `TOTALLYFUZZY` run to mine the trademark database. Section VI presents this database, the implementation and 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 used in the next sections to illustrate the definitions and formalism we propose.

Choosing a trademark is a decisive point for a company. Stakes are financially high and consequences of a mischosen trademark can be disastrous. For this reason it could be interesting and useful to find out and understand how trademarks are built identifying recurrent patterns in the INPI trademark database. This database contains around 2 million marks registered between 1961 and 2003 in France and so cannot be entirely studied using statistical methods.

A sample of the INPI database is given in Table I. It is extracted from the class 32, which registers trademarks linked to “beer, spring mineral water, soft drink, fruit juice”. In this data set, each row records one registration with its date and the registered trademark. The first row represents the first registration of company C1. It means that it registered the trademark “AceCool[®]” at the date d1. From this dataset a database containing word composition can be derived (Table II). This set can then be used to analyse trademarks composition as presented in Section IV-B. A summarization of this database could be for instance that “Most company register at least one trademark beginning with a *capital a*”.

Using association rules to build fuzzy summaries in our context could help in finding frequent correlations in letters or morpheme association such that “Most marks containing several *a* also contain few *b*” or “Most marks are composed of few letters and few figures”. However some information is still hidden such as letter ordering in trademarks.

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

Bénédicte 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 Thé
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 composed of more than two syllables” or that “In most trademarks a is often followed by several consonants”.

Therefore using sequential patterns and more particularly fuzzy sequential patterns can help. In fact, sequential patterns are an extension of association rules enabling us to take the order aspect of data into account. 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 patterns will precise the approximative number of a and b .

III. SUMMING UP DATABASES USING ASSOCIATION RULES

Fuzzy summaries have been studied since the 1980's [1]. 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 trademarks 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 an 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 rule characteristics such as *support*, *confidence* or *lift*.

Fuzzy association rules are an extension of the association rule based approach proposed by [10]. 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 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 attributes) and A and B are ordered sets of fuzzy sets respectively associated to the attributes of 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 its certainty.

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 $(X, A) \wedge (Y, B)$ 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 as having 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 interpreted as “Most of trademarks containing a lot of words and letters contain few figures”. However some information is still hidden such as the generic form of words. For example, we can not know if trademarks are composed first of a short word then of a long one or if letters are followed or preceded 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 [11] take the temporal aspect of data into account, so they are well-adapted to ordered or historically-stamped data, such as monitoring data, texts or linguistic 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 objects records where each record r consists of three information elements: an object-id, a record timestamp and a set of attributes/items in the record. Let $I = \{i_1, i_2, \dots, i_m\}$ be a set of items or attributes. An *itemset* is a non-empty set of attributes i_k , 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 s_p , denoted by $\langle s_1 s_2 \dots s_p \rangle$. A *n-sequence* is a sequence of n attributes (or of size n).

Example 1: Let us consider an example of market basket analysis. The object is a customer, and records are the transactions made by this customer. Timestamps are the date of transactions. If the customer Smith purchases products 1, 2, 3, 4, and 5 made by the customer Smith according to the sequence $s = \langle (1) (2\ 3) (4) (5) \rangle$, then all attributes of

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

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

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

All records from an object are grouped together and sorted in increasing order of their timestamp. They are called a data sequence. An object *supports* a sequence s if it is included within its data sequence (s is a subsequence of the data sequence). The *support* of a sequence ($supp(s)$) is defined as the percentage of objects supporting s . In order to decide whether a sequence is frequent or not, a minimum support value ($minSupp$) is specified by the user. The sequence is said to be frequent if the condition $supp(s) \geq minSupp$ holds. Given a database of object records, the problem of sequential pattern mining is to find all maximal frequent sequences [11]. Note that items are processed using a binary evaluation – present or not present. In our case, we know that names are composed 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 linguistic 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 attribute and itemset concepts have been redefined relative to classical sequential patterns, as in [12].

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 will note: $(X, A) = ([x_1, a_1], \dots, [x_p, a_p])$ where X is a set of items, 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 contains only 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** $S = \langle s_1 \dots s_g \rangle$ is a sequence constituted by g fuzzy itemsets $s = (X, A)$ grouping together k fuzzy items $[x, a]$.

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

In the next sections, we use the following notations: \mathcal{O} represents the set of objects and \mathcal{R}_o the set of records for one object o . Let \mathcal{I} be the set of attributes and $r[i]$ the value of attribute i in record r . Each attribute i is divided into fuzzy sets. We here apply these generic definitions to the database given in Table I. In this example, $\mathcal{O} = \{C1, C2, C3\}$, the companies who have registered trademarks. Each trademark is parsed to get the number of words, characters, letters and symbols. For example, Asia-cola[®] contains one word, 10 characters, 8 letters and 2 symbols. This translation is given in Table II. As no trademark contains figures, this column is not represented.

TABLE II
RESULTS OF TABLE I PARSING

ID_comp	Reg_date	#word	#char	#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 linguistic expert knowledge. The fuzzy sets, for each attribute, are given in Fig. 1.

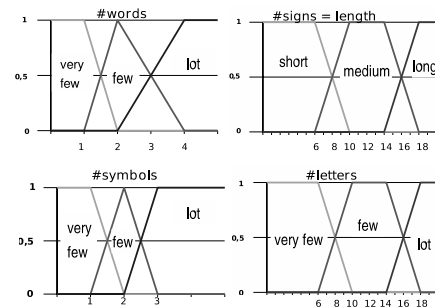


Fig. 1. Fuzzy sets

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 registration for C3, described in Table III. This is used to illustrate the upcoming definitions and algorithms for the itemset $iS = \langle ([letter, very\ few]) ([length, medium]) \rangle$ “Trademarks with very few letters have a medium length”.

TABLE III
MEMBERSHIP DEGREES FOR C3

D.	Fuzzy items									
	#word		sh.	length			v.f.	#letter		#symb
f.	l.	m.		l.	l.	f.		l.	v.f.	f.
d1	1			1		<u>0.25</u>	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	<u>0.75</u>		<u>0.75</u>	0.25			1
d5	0.5	0.5		1			1		1	
d6		1			1			1		

In next sections, $\mu_a(r[x])$ represents the membership degree of the record r for the item s and the fuzzy set a .

C. Three Support Computation Methods

T.-P. Hong and al [13] and then Y.-C. Hu and al [14] presented two proposals of fuzzy sequential pattern mining approaches. We extended their initial definitions of fuzzy support to provide the user with three levels of fuzzification, which could be used to cope with different problems.

The *support¹ of a fuzzy itemset* is computed as the proportion of objects supporting it. However the cardinality of a fuzzy set depends on the counting method. We apply here three of these techniques in the framework of fuzzy sequential patterns and we propose three definitions for the fuzzy support of an itemset:

- **SPEEDYFUZZY** is based on the count “supports / does not support”. Computing the support of a fuzzy itemset consists of counting all the elements for which the membership degree is not null:

$$S_{SF}(o, (X, A)) = \begin{cases} 1 & \text{if } \exists r \in \mathcal{R}_o | \forall [x, a] \in (X, A), \mu_a(r[x]) > 0 \\ 0 & \text{else} \end{cases} \quad (1)$$

Example 6: With **SPEEDYFUZZY**, company 1 supports the itemset iS since a record is found containing the regarded itemset 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 objects supporting the fuzzy itemset only when each item of the candidate sequence has a membership degree greater than a specified threshold in the object data sequence:

$$S_{MF}(o, (X, A)) = \begin{cases} 1 & \text{if } \exists r \in \mathcal{R}_o | \forall [x, a] \in (X, A), \mu_a(r[x]) > \omega \\ 0 & \text{else} \end{cases} \quad (2)$$

Example 7: With **MINIFUZZY**, company 1 supports the itemset iS since a record is found containing the items with a membership degree greater than the threshold ($\omega=0.49$), 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

¹Note 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 [15]), but rather the frequency of this fuzzy item, itemset or sequence in the database.

taken into account in the support computation. To do so the threshold membership function α is defined as:

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

The support counting formula becomes:

$$S_{TF}(o, (X, A)) = \frac{|\mathcal{R}_o|}{|\mathcal{O}|} \prod_{[x, a] \in (X, A)} [\alpha_a(r_j[x])] \quad (4)$$

where \prod and $\frac{|\mathcal{R}_o|}{|\mathcal{O}|}$ are the generalized t-norm and t-conorm operators, here *min* and *max*.

Example 8: With **TOTALLYFUZZY**, company 1 supports the itemset iS if a records is found containing all fuzzy items of this itemset, with a membership degree greater than the threshold ω (0.49). The best value for the itemset is kept, items twice underlined in Table III.

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

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

where the support degree $S(o, gS)$ indicates wether the object o supports the fuzzy sequence gS . Table IV is used to illustrate the support computation for the sequence $mS = \langle ([letter, very\ few]) ([word, lot]) \rangle$, “Registrations first contain names with very few letters then tradenames are consisting of few words”. This support degree is computed using the algorithms **SPEEDYFUZZY**, **MINIFUZZY** and **TOTALLYFUZZY**. In this paper we only detail **TOTALLYFUZZY** which has been used for experiments on the INPI Trademark Database presented Section VI.

For **SPEEDYFUZZY** (resp. **MINIFUZZY**) the support degree $S(o, mS)$ for o is 1 if each itemset of the sequence is found holding the condition $S_{SF}(o, (X, A)) = 1$ (resp. $S_{MF}(o, (X, A)) = 1$). This corresponds to records underlined (resp. boldfaced) in Table IV. For **TOTALLYFUZZY**, the support degree is the mean aggregation of the support degree of each mS itemset, keeping the best one, as detailed Section V. This corresponds to records twice underlined in Table IV.

TABLE IV
MEMBERSHIP DEGREES FOR C3

D.	Fuzzy items									
	#word		sh.	length			v.f.	#letter		#symb
f.	l.	m.		l.	l.	f.		l.	v.f.	f.
d1	1			1		<u>0.25</u>	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		<u>0.75</u>	0.25			1
d5	0.5	0.5		1			1		1	
d6		1			1			1		

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 objects 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 or even in a certain way SPEEDYFUZZY or MINIFUZZY, an object supports a sequence or not. So the scan can stop as soon as the sequence is found within the record set of an object. On the contrary, as TOTALLYFUZZY computes a thresholded Σ -count, for each object and each sequence, the best membership degree must be considered. This degree is computed as the aggregation of the itemset frequencies. The order of the fuzzy items must also be taken into account. This leads to an exhaustive scanning of the record set, as performed for association rule mining. Let us consider for instance the table IV. If we look for the same sequence as above, i.e. mS , $\langle ([letter, very\ few]) ([word, lot]) \rangle$, the first occurrence of this sequence is the one underlined, supported by record d1 then d3. However, the best occurrence of this sequence is given by record d4 followed by d6.

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

To reduce this number of scans, we have defined a data structure enabling us to find all representations of all k -sequences in only k scans of the database, as for the existing approaches mining crisp sequential patterns. The computational time is also 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 within the object record set. Several paths may be initialized for one object. For the global frequency 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 illustrates how TOTALLYFUZZY works, 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 $\langle ([letters, very\ few]) ([words, lot]) \rangle$ for customer 3 from table III, with a threshold $\omega = 0.2$. This is summarized in Table V.

First the process is initialized by creating one first empty path $pth1 = (\emptyset, ([letter, very\ few]), 0)$ (Table V, row 1). Then it begins by scanning the first record of customer 3. The

TABLE V
ILLUSTRATION OF TOTALLYFUZZY ON C3

	$pth1 : (\emptyset, ([letter, very\ few]), 0)$
After d1	$pth1 : (\langle ([letter, very\ few]) \rangle, ([words, lot]), 0.25)$
After d2	$pth1 : (\langle ([letter, very\ few]) \rangle, ([words, lot]), 0.25)$ $pth2 : (\langle ([letter, very\ few]) \rangle, ([words, lot]), 0.5)$
Opt.	$pth1$ is deleted
After d3	$pth2 : (\langle ([letter, very\ few]) ([words, lot]) \rangle, \emptyset, 0.5)$, closed $pth3 : (\langle ([letter, very\ few]) \rangle, ([words, lot]), 0.5)$
After d4	$pth2 : (\langle ([letter, very\ few]) ([words, lot]) \rangle, \emptyset, 0.5)$, closed $pth3 : (\langle ([letter, very\ few]) ([words, lot]) \rangle, \emptyset, 0.5)$, closed $pth4 : (\langle ([letter, very\ few]) \rangle, ([words, lot]), 0.5)$ $pth5 : (\langle ([letter, very\ few]) \rangle, ([words, lot]), 0.75)$
Opt.	$pth3$ and $pth4$ are deleted
After d5	$pth2 : (\langle ([letter, very\ few]) ([words, lot]) \rangle, \emptyset, 0.5)$, closed $pth5 : (\langle ([letter, very\ few]) ([words, lot]) \rangle, \emptyset, 0.63)$, closed $pth6 : (\langle ([letter, very\ few]) \rangle, ([words, lot]), 0.75)$
Opt.	$pth2$ is deleted
After d6	$pth5 : (\langle ([letter, very\ few]) \rangle, ([words, lot]) \rangle, \emptyset, 0.63)$, closed $pth6 : (\langle ([letter, very\ few]) \rangle, ([words, lot]) \rangle, \emptyset, 0.87)$, closed
Opt.	$pth5$ is deleted
C3 deg.	0.87

currently searched itemset *curIS* is found with a degree $d1[letter, very\ few] = 0.25 \geq \omega$. The path *pth1* is so updated with $pth1.seq \leftarrow \langle ([letter, very\ few]) \rangle$, $pth1.curIS \leftarrow ([words, lot])$ and $pth1.curDeg \leftarrow 0.25$ (Table V, row 2).

Then record d2 contains the first sequence of the candidate sequence *g-S*, $([letter, very\ few])$. So a new path is created, $pth2 \leftarrow (\langle ([letter, very\ few]) \rangle, ([words, lot]), 0.5)$. As this record does not contain the next itemset for *pth1*, the scan of the dataset should continue. Before that, an optimization is carried out to avoid having to use too much memory space: 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 scans the next record with *pth2* (Table V, row 3).

Transaction d3 is then checked. It contains $pth2.curIS = ([words, lot])$. It is thus modified to $pth2 \leftarrow (\langle ([letter, very\ few]) ([words, lot]) \rangle, \emptyset, 0.5)$. This path is closed since it contains all itemsets of *g-S*. However, we plan 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 $pth3 = (\langle ([letter, very\ few]) \rangle, ([words, lot]), 0.5)$ (Table V, 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 V, row 5). The *Optimize* function deletes for each step in the sequence the path with the lowest degree, i.e. *pth3* and *pth4*. Scanning then continues. At d5, $pth5$ is copied into *pth6*, modified and closed with a support degree of 0.63 (mean value of 0.5 and 0.75), so *pth2* is deleted. Finally, at d6, *pth6* is updated and closed with a degree of 0.87. This is the path kept for C3.

C. Computing a sequence support: the Algorithms

TOTALLYFUZZY uses the function *FindTotallySeq* to run an ordered scanning of an object data sequence and finds the best occurrence of a sequence in this record set.

When the first itemset of the sequence is found, one path is created with the itemset support. The next records 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 possible paths are thus completed step-by-step at each record. The support degree of the best path for the whole sequence is then returned. The *Update* function, not presented in this paper, allows an update of each path and closes the complete ones. The *Optimize* algorithm, not presented in this paper, enables deletion unnecessary paths, which means, for two paths at the same point, to only keep the one with the greatest degree.

ALGORITHM 1 - FindTotallySeq

Input: gS , candidate g - k -sequence;

R , record set to run

Output: m , support degree of the best g - S representation instantiated in the record set R

$Paths$: list of paths \rightarrow (seq, curIS, curDeg)

$Paths \leftarrow Path(\emptyset, gS.first, 0)$

```

For each record  $r \in R$  do
  For each path  $pth \in Paths$ , not updated at  $r$  do
    If ( $pth$  not closed) Then
      If ( $pth.curIS \in r$ ) Then
         $pth.curDeg \leftarrow pth.curDeg \mid \overline{\top}_{[x,a] \in pth.curIS} \alpha_a(r[x])$ ;
        Update( $pth$ );
      End If
    End If
    For  $j$  from 2 to  $pth.curIS - 1$  do
      [Search for an improvement of the current path]
      If ( $(gS.get(j) \in r) \&$ 
         $(\overline{\top}_{[x,a] \in gS.get(j)} \alpha_a(r[x]) > pth.curDeg[j])$ ) Then
         $nCurIS \leftarrow gS.get(j)$ ;
        For  $i$  from 1 to  $j-1$  do
           $nSeq \leftarrow nSeq \mid gS.get(i)$ ;
           $nCurDeg \leftarrow nCurDeg \mid pth.curDeg[i]$ ;
        End For
         $nCurDeg \leftarrow nCurDeg \mid \overline{\top}_{[x,a] \in gS.get(j)} \alpha_a(r[x])$ ;
         $Paths \leftarrow Paths \cup Update(nSeq, nCurIS, nCurDeg)$ ;
      End If
    End For
  End For
  If ( $(gS.first \in r) \&$  (not(FirstPass))) Then
     $pth \leftarrow Path(\emptyset, gS.first, \overline{\top}_{[x,a] \in gS.first} \alpha_a(r[x]))$ ;
     $Paths \leftarrow Paths \cup pth$ ;
    Update( $pth$ );
  End If
  Paths.Optimize(); [deletion of less pertinent paths]
End For
 $pth \leftarrow Paths.first$ ; [Paths only contains the best complete path]
 $m \leftarrow \odot(pth.curDeg)$ ; [Aggregation to return the support degree]
return  $m$ ;

```

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 2 - CalcTotallySupport

Input: gS , candidate g - k -sequence ;

Output: $FSupp$ fuzzy support for the sequence gS ;

$FSupp, nbSupp, m \leftarrow 0$;

```

For each object  $o \in \mathcal{O}$  do
   $m \leftarrow FindTotallySeq(gS, \mathcal{R}_i)$ ;
   $nbSupp += m$  ;
End For

```

$FSupp \leftarrow nbSupp/\Gamma$;

return $FSupp$;

D. Mining Frequent Sequences

Our approach extends the level-wise approach generate-prune within the sequential pattern context and, more particularly, uses the prefix-tree structure described in [16], in order to improve the support computation process. The overall algorithm PSP-Totally is presented by Algorithm 3.

First, the fuzzy support of all fuzzy items is computed and only items with a support greater than $minSupp$ are stored as frequent ones. Then, while a candidate generation is possible, the *generate* function builds the candidate sequences of size k from the frequent sequences of size $k-1$. Then a scan of the database is run and supports are computed using *CalcTotallySupport*. Finally, all infrequent k -sequences are pruned and the process continues. At the end of the process, we obtain a Prefix-tree containing all frequent sequences of the database with their support at each leaf of the tree.

ALGORITHM 3 - PSP-Totally

Input: DB , a database, $minSupp$

Output: 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
    | CalcTotallySupport( $s$ );
  End For
  prune( $PT$ );
End While
return ( $PT$ ) ;

```

As for PSP, the overall computational complexity of PSP-Totally depends only on the length of the candidate sequences, and thus of the support calculation. Computation of the *generate* and *prune* functions are indeed insignificant in comparison with that of *CalcTotallySupport*.

VI. EXPERIMENTS

In this paper we present one of our mining experiments, looking for sequences of words in registered trademarks (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 Trademark Database

INPI is the French *National Institute for the Industrial Property*. It manages and registers industrial title deeds, communicates information on industrial property and devises and adapts French industrial property rights.

The INPI trademark database registered around 2 million names between 1961 and 2004. These trademarks have been registered in one or several categories corresponding to industrial fields such as “toys, games” or “clothing, shoes, hat industry”. Each record is composed of several attributes, for instance: idnumber, trademark, registration date, registering company, registering class.

The global goal of analysing such a database is, for the linguistic expert, to understand how trademarks act on consumers. One step towards this aim is to understand building

mechanisms 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 done thanks to statistical methods such as plotting histograms or graphs to look at evolution or distribution of registration number. However 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 trademark database up. Then this data set contains many quantitative information such as letter number or word number for a same trademark. So it requires to use Fuzzy Sets Theory based mining approaches in order to help the linguistic 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 trademark database.

B. Data Translation for Intra-name Pattern Mining

These experiments aim at discovering word form in registered tradenames. Summaries would be described for example as “Almost one third of registered names are first composed of one part containing a lot of letters then of a part with few figures and then of a part with few punctuation symbols and few letters”. These proposition would be built from a sequence $\langle ([letter, lot])([figure, few])([punct_symp, few], [letter, few]) \rangle$. In this case, we can consider trademarks consisting of one or several words. In fact a name consisting of only one word will appear as an itemset instead of a sequence for a name consisting of several words. In order to mine such patterns, the INPI database is converted into the format [ID_TM, #WORD, FUZZY ITEM, DEGREE], as shown Table VI, whereas the fuzzy items are described on Fig. 2.

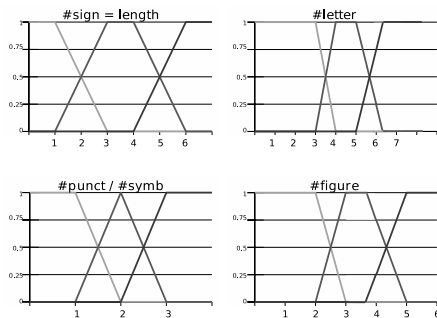


Fig. 2. Fuzzy sets for fuzzy items for intra-name pattern mining

C. From Fuzzy Sequential Patterns to Fuzzy Summarization

First we decided to begin with the mining into classes that seemed to have an atypical behavior through statistical analysis. The class number 38, registering trademarks linked to the telecommunication field, was given to contain more

TABLE VI

DATABASE TRANSLATION FOR INTRA-NAME PATTERN MINING		
Mining formalism		Trade mark database
object	\leftrightarrow	a trademark
timestamp	\leftrightarrow	number of the word in the trademark
fuzzy item	\leftrightarrow	# of characters, letters, figures, punctuations or symbols in this word

symbols and punctuations than the other trademarks. Our summarization shows that even if this class contains a greater proportion of symbols, it is not a characteristic point. We detail here results obtained by mining this dataset of more than 480,000 trademarks composed of one or several words.

First we mined for a general description of this class. For a minimum support 70% ($minSupp = 0.7$), we obtain the sequential pattern $\langle ([\#character, lot][\#letter, lot]) \rangle$ (76.8%), which was translated into the statement “Most of trademarks contain at least one word which contain a lot of characters and a lot of letters”. By decreasing the $minSupp$ value, we found that “Almost one quarter of trademarks contain at least two words, both having a lot of characters and a lot of letters” ($\langle ([\#character, lot][\#letter, lot]) ([\#character, lot][\#letter, lot]) \rangle$ (24.7%)).

During a second step, we divided the class 38 into four databases, according to the number of words by trademarks: 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 aimed at discovering more specific knowledge about trademarks in the class 38. What we can give as a summarization for this category of trademarks is that: “Most trademarks composed of one word contains a lot of characters and a lot of letters ($\langle ([\#character, lot][\#letter, lot]) \rangle$ (74.1%)” and “Only few trademarks consisting of one word contains a medium amount of characters or letters ($\langle ([\#character, medium]) \rangle$ (14.6%), $\langle ([\#letter, medium]) \rangle$ (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 can only be obtained using fuzzy sequential pattern summarization.

Mining in trademarks composed of two words revealed that there composition is quite various. We have indeed discovered that even “Most of them contain at least one word with a lot of characters” ($\langle ([\#character, lot]) \rangle$ (81%)), “Only one third of them contain two words consisting of a lot of characters, one preceding the other with a lot of letters” ($\langle ([\#character, lot]) ([\#character, lot][\#letter, lot]) \rangle$ (33%)). We had then to decrease the support to 0.1 to learn more about the intra-structure of these trademarks. We found that “A small part of class 38 composed of two words is composed of one word with a medium amount of characters and few letters preceding a second one with both a lot of characters and letters” ($\langle ([\#character, medium][\#letter, few]) ([\#character, lot][\#letter, lot]) \rangle$ (10.5%)).

Finally we give some information obtained concerning the trademarks of class 38 composed of at least three

words. We have found out that “Almost two third are composed of two words with both a lot of characters and letters” ($\langle ([\#character, lot][\#letter, lot]) ([\#character, lot][\#letter, lot]) \rangle$ (62%)); it was completed by decreasing the support: “Almost one half of these trademarks contain three words with a lot of characters, the first one with a lot of letters, one following then and preceding a third one with a lot of letters” ($\langle ([\#character, lot][\#letter, lot]) ([\#character, lot]) ([\#character, lot][\#letter, lot]) \rangle$ (43.5%)), whereas “Almost one half of these trademarks contain three words with a lot of characters, the first and last ones with a lot of characters and letters, the second one with few letters or a medium amount of characters” ($\langle ([\#character, lot][\#letter, lot]) ([\#letter, few]) ([\#character, lot][\#letter, lot]) \rangle$ (42%), $\langle ([\#character, lot][\#letter, lot]) ([\#character, medium]) ([\#character, lot][\#letter, lot]) \rangle$ (42%)).

It appears that trademarks containing figures, symbols or punctuations in this category are rare and infrequent. On the contrary to the statistical overview, we can say that figures or symbols are not so characteristic of this class.

Our conclusion on these experiments is that the fuzzy partitioning intuitively given by the expert from common language does not mesh well with the trademark composition. Further experiments will be carried out with automatically built fuzzy sets from the database using a clustering algorithm as presented in [8] and [17].

D. Mining for Trends in Trademark Morphology

The currently carried out experiments aim at mining for evolution in trademark registration for a same registering company. For example, we could find that “Few registering companies first registered short names containing few figures and then short names containing a lot of punctuation symbols”. This would be represented by a sequence $\langle ([character, few], [figure, few])([character, few], [punct_symp, lot]) \rangle$. This requires to pre-process 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 characters, 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 have been built from a linguistic expert knowledge and should next be automatically built. The results of these experiments will be used to extend the method proposed in [18] to discover trends in text databases and will be then completed applying temporal constraints, as presented in [19].

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 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 trademarks. Further experiments are currently in progress to sum up the evolution of trademark structure during the last forty years.

ACKNOWLEDGMENT

The authors would like to thank Mr. Stéphane Sanchez and Mr. Federico Del Razo Lopez for their help in the management and pre-processing of the INPI database.

REFERENCES

- [1] R. R. Yager, “A new approach to the summarization of data.” *Information Sciences*, vol. 28, no. 1, pp. 69–86, 1982.
- [2] 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.
- [3] 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.
- [4] 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.
- [5] 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.
- [6] 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.
- [7] J. Kacprzyk and S. Zadrozny, “Fuzzy linguistic summaries via association rules,” *Data Mining and Computational Intelligence*, 2001.
- [8] 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.
- [9] C. M. Kuok, A. W.-C. Fu, and M. H. Wong, “Mining Fuzzy Association Rules in Databases,” *SIGMOD Rec.*, vol. 27(1), 1998.
- [10] 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.
- [11] 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.
- [12] C. Fiot, A. Laurent, and M. Teisseire, “Motifs séquentiels flous : un peu, beaucoup, passionnément,” in *5èmes journées d’Extraction et Gestion des Connaissances*, 2005, pp. 507–518.
- [13] 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.
- [14] 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.
- [15] A. Kaufmann, “Introduction to the theory of fuzzy subsets,” 1973.
- [16] F. Masegaglia, 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.
- [17] A. Gyenesi and J. Teuhola, “Multidimensional fuzzy partitioning of attribute ranges for mining quantitative data: Research articles,” *Int. J. Intell. Syst.*, vol. 19, no. 11, pp. 1111–1126, 2004.
- [18] 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.
- [19] C. Fiot, A. Laurent, and M. Teisseire, “Des motifs séquentiels généralisés aux contraintes de temps étendues,” in *6èmes journées d’Extraction et Gestion des Connaissances*, 2006, pp. 603–614.