Order and Mess in Text Categorization: Why Using Sequential Patterns to Classify

Simon Jaillet, Anne Laurent, Maguelonne Teisseire, Jacques Chauché

Order and Mess in Text Categorization: Why Using Sequential Patterns to Classify

S. Jaillet, A. Laurent, M. Teisseire, J. Chauché
LIRMM UMR CNRS 5506
Université Montpellier II
161, Rue Ada 34392 Montpellier Cedex 5
FRANCE
jaillet,laurent,teisseire,chauche@lirmm.fr

ABSTRACT

Text categorization is a well-known task essentially based on statistical approaches using neural networks, Support Vector Machines and other machine learning algorithms. Texts are generally considered as bags of words without any order. Although these approaches have proven to be efficient, they do not provide users with comprehensive and reusable rules about their data. These rules are however very important for users in order to describe the trends from the data they have to analyze. In this framework, an association-rule based approach has been proposed by Bing Liu (CBA). In this paper, we propose to extend this approach by using sequential patterns in the SPaC method (Sequential Patterns for Classification). Taking order into account allows us to represent the succession of words through a document without complex and time-consuming representations and treatments such as those performed in natural language and grammatical methods. The original method we propose here consists in mining sequential patterns in order to build a classifier. We show on experiments that our proposition is relevant, and that it is very interesting compared to other methods. In particular, our method has better results than SVM when SVM do not perform well. Moreover, our approach is very interesting for huge volumes of data.

Keywords

text mining, categorization, sequential patterns

1. INTRODUCTION

Automatic text classification goes back at least to the 1960's [22]. But with the growing volume of available digital documents, researches on automatic classification have been extensively addressed in the past few years in order to define efficient and scalable methods [28, 33].

There are two distinct types of approaches in automatic classification: the supervised one and the unsupervised one. In the supervised classification or categorization, categories are defined by an expert while they are automatically learned in the other (also called clustering) [26, 13]. In this article we focus on text categorization or, in other words, on supervised learning algorithms.

Currently, existing classifiers are mostly based on statistical methods and Support Vector Machines. These methods are based on word frequencies such as $TF\text{-}IDF$ (Term Frequency, Inverse Document Frequency) [26]. However, these methods do not provide understandable descriptions of the extracted knowledge. In order to cope with this problem, an association-rule based approach has first been proposed by Bing Liu (CBA) [19], which has been enhanced by [31], [18], [14] and [10], and by ARC-BC in [6] and ARC-PAN in [7]. Association rules have indeed been extensively studied and are very efficient and scalable.

All these methods consider each text as a so-called *bag of words* where no order between words is taken into account for categorization. This textual representation has proven to be useful and almost as efficient as complex representations which require time-consuming methods like syntactic analysis. It is thus interesting to investigate methods that take order into account and remain scalable.

In this paper, we propose thus to extend the CBA method by taking order into account. In our approach, the order management is performed by considering the extraction of sequential patterns instead of association rules. A sequential pattern is a set of items appearing consecutively in the database.

The originality of our approach SPaC (Sequential PAtterns for Categorization) is that we use sequential patterns for text categorization. For this purpose, two steps are considered. The first step consists in mining sequential patterns from texts. The granularity we consider is the sentence, meaning that each sentence is considered as an unordered set of items (words) whereas texts are considered as ordered sets of sentences. The second step consists in classifying documents using these sequential patterns.

In this paper, we show that sequential pattern-based classification is very efficient when SVM do not perform well. In this framework, sequential patterns have three main advantages: first they provide understandable rules (contrary to SVM, Rocchio, naïve Bayes,...). Second they allow trend analysis, as shown in [17]. Third, they extract patterns that are more precise and informative than association rules. Note that our approach is evaluated using precision-recall merged in $F_\beta-$measure [25] and not only accuracy. These measures have indeed proven to be more relevant to compare text classification methods [29].

The paper is organized as follows. Section 2 presents the background of the problem being addressed by introducing sequential patterns and textual representations. Section 3 details existing methods dealing with text mining with "frequent patterns" and "sequential patterns". Section 4 details our method based on Sequential Patterns (SPaC). Section 5 shows that our method performs well on datasets in French and English. Finally, Section 6 summaries the paper and presents future work.

2. PROBLEM STATEMENT

First, we formulate the concept of sequence mining by summarizing the formal description of the problem introduced in [3] and extended in [30]. Second we look at the categorization problem.

2.1 Mining of sequential patterns

Let DB be a set of customers' transactions where each transaction T consists of customer-id, transaction time and a set of items involved in the transaction.

Let $I = \{i_1, i_2, ..., i_m\}$ be a set of literals called *items*. An *itemset* is a non empty set of items. A sequence s is a set of itemsets ordered according to their time stamp. It is denoted by $< s_1 s_2 ... s_p >$ where s_j, $j \in 1..n$, is an itemset. A *n-sequence* is a sequence of n items (or of length n). For example, let us consider a given customer who purchased items $1, 2, 3, 4, 5$, according to the following sequence: $s =<$ (1) (2, 3) (4) (5)>. This means that apart from 2 and 3 that were purchased together, i.e. during the same transaction, items in the sequence were bought separately. s is a 5-sequence.

A sequence $< s_1 s_2 ... s_p >$ is a sub-sequence of another sequence $< s'_1 s'_2 ... s'_m >$ if there exist integers $i_1 < i_2 < ... i_j ... < i_n$ such that $s_1 \subseteq s'_{i_1}, s_2 \subseteq s'_{i_2}, ..., s_p \subseteq s'_{i_n}$. For example, the sequence $s' = <$ (2) (5) $>$ is a sub-sequence of s because (2) $\subseteq$ (2, 3) and (5) $\subseteq$ (5). However $<$ (2) (3) $>$ is not a sub-sequence of s since items were not bought during the same transaction.

All transactions from the same customer are grouped together and sorted in increasing order. They are called a *data sequence*. A support value (*supp(s)*) for a sequence gives its number of actual occurrences in DB. Nevertheless, a sequence in a data sequence is taken into account only once to compute the support even if several occurrences are discovered. In other words, the support of a sequence is defined as the fraction of total distinct data sequences that contain s. A data sequence contains a sequence s if s is a sub-sequence of the data sequence. 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 customer transactions the problem of sequential pattern mining is to find all sequences whose support is greater than a specified threshold (minimum support). Each of which represents a *sequential pattern*, also called a *frequent* sequence.

Sequential patterns are usually extracted from a database built on the following schema: $date, client, items$. For instance, we consider the database of client purchases in a supermarket, as shown in Table 1. Each line (transaction, tuple) from this table corresponds to the set of items bought by the client at the corresponding date.

In this example, Peter has bought the items $1, 2, 3, 4, 5$ as the sequence <(1)(2,3)(4)(5)>.

Client	Date	Items
Peter	04/01/12	TV (1)
Martin	04/02/28	Chocolate(5)
Peter	04/03/02	DVD_Player (2) , Camera (3)
Peter	04/03/12	Printer (4)
Peter	04/04/26	Chocolate (5)

Table 1: Database of Purchases

2.2 Textual Representation and Categorization

Text categorization is the task of assigning a boolean value to each pair (document, category). A categorization process has to define: (1) a formalization of texts and category (2) a (text, category) measurement (3) a categorization policy.

The textual database is partitioned into two databases T_{Train} and T_{Test} where T_{Train} stands for a training set and T_{Test} stands for a test set.

In usual methods, texts are represented as bags of words [29]. The order is not considered. Each document is represented by a vector where each component is a word weighted by a numerical value. The most used weighting is *TF-IDF* (Term Frequency - Inverse Document Frequency)[26]. For a word w, we have:

$$tfidf(w) = tf(w).log\frac{N}{df(w)}$$

where $tf(w)$ is the number of occurrences of w in the document, $df(w)$ if the number of documents containing w and N is the total number of documents. The weight $tfidf(w)$ represents thus the relative importance of the word in the document.

These vectors describing documents are then used in order to extract knowledge using common algorithms such as k-nearest neighbors, SVM, naive bayes, etc.

3. RELATED WORK

Text mining has been widely investigated [17, 5, 1, 29]. In this section, we focus on text classification and frequent patterns.

3.1 Classification Based on Associations: the CBA Method

In [20] the authors propose CBA: a text categorization method based on association rules. CBA consists of two parts, a rule generator (CBA-RG) which is based on the well-known Apriori algorithm [2] and a classifier builder (CBA-CB) which is based on generated rules.

3.1.1 CBA-RG

In this first step, the key is to find all ruleitems whose support is greater than a specified threshold (minimum support). A ruleitem is defined by: $< conset, C_i >$ where condset is a set of items and C_i is a class label. Each ruleitem ρ represents a rule $condset \rightarrow C_i$ and the support and the confidence of such a rule is defined by:

$$sup(\rho) = \frac{\#\text{texts from } C_i \text{ matching condset}}{|D|}$$

$$conf(\rho) = \frac{\#\text{texts in } C_i \text{ matching condset}}{\#\text{texts in D matching condset}}$$

CARs (classification rules) thus consists of all the ruleitems that satisfy the minimum support and a minimum confidence.

In these approaches, frequent patterns are extracted using a single minimum support threshold. However, categories are not always equi-distributed. It is thus not relevant to consider such a single value. Choosing a relevant minimum support threshold is crucial so that frequent patterns are relevant for the categorization task. A high support will indeed prevent the system from finding frequent patterns for a small category, while a low support will lead to the generation of a huge number of rules, which is not interesting because it will lead to overfitting.

Works have been proposed in order to define a multiple minimum support application (msCBA) [14, 21]. In these approaches, ruleitems are extracted using a multiple minimum support strategy. The minimum support of each category is defined according to the distribution frequency of each category and the user minimum support threshold:

$$minSup_{C_i} = minSup_{user} * freqDistr(C_i)$$

where, $freqDistr(C_i) = \dfrac{\#\text{texts from } C_i}{\#\text{texts}}$.

3.1.2 CBA-CB

Let R be the set of CARs and D the training data. The basic idea of the algorithm is to choose a set of high precedence rules in R to cover D. The categorizer is thus represented by a list of rules $r_i \in R$ ordered according a total order based on the confidence. We have thus, $< (r_1, r_2, ..., r_k), C_i >$ (where C_i is the target category and r_j one of the associated rules).

Each rule is then tested over D. If a rule does not improve the accuracy of the classifier, this rule and the following ones are discarded from the list.

Once the categorizer has been built, classification rules are tested until a condition is matched. The text is then associated with the target class of the classification rule.

3.2 Enhancements and Other Approaches

In [14], the authors propose to replace the confidence by the intensity of implication when sorting the rules to build the classifier. In [16], the authors integrate the CBA method with other methods such as decision trees.

In [10], the authors investigate a method for association rule classification. Contrary to CBA-CB which takes only one rule into account, they propose to determine the category of a text by considering several rules that are mixed using a majority voting. In order to cope with the huge number of rules, the authors propose a pruning method during the extraction of the classification rules using χ^2, as done in [18]. $maxrules$ stands for the maximum number of rules used to classify new cases. Moreover, rules are used at different levels. This approach is enhanced in [9] by considering several minimum support thresholds.

In [5], association rules are used for partial classification. Partial classification means that the classifier does not cover all cases. In particular, this work is interesting when dealing with missing values.

[17, 32] propose to use sequential patterns in the text mining framework. In [32], the proposition is based on two methods. The first method is based on the visualization of word occurrences in order to detect sequential patterns. The second method is based on classical methods to extract sequential patterns. However, the authors do not propose a method to classify texts from the sequential patterns. Moreover, the texts considered are associated with a date. The corpus consists of 1,170 articles collected over 6 years. This point makes it very different from our proposal and more difficult to apply since texts are rarely associated with a date.

In [17], authors demonstrate how sequential patterns are useful for text mining. Sequential patterns are used in order to extract trends from textual databases.

As we show in this section, text mining with frequent patterns is either performed as a classification task using association rules, or as a non classification task using sequential patterns. Note that [6, 7] propose a classification method based on association rules. However this method is neither based on sequential patterns nor devoted to text classification. We propose thus an original method based on sequential patterns for classification. We argue that this method is able to deal with order in texts without being time-consuming. Next section details our approach. Section 5 shows that our method obtains good results compared to other ones.

4. SEQUENTIAL PATTERNS FOR CLASSIFICATION: THE SPAC METHOD

In this paper, we propose an original method (SPaC) for text classification based on sequential patterns. This methods consists in two steps. In the first step, we build sequential patterns from texts. In the second step, sequential patterns are used to classify texts.

4.1 From Texts to Sequential Patterns

Each text is a set of words. Our method is based on sequential pattern mining. Texts are represented as ordered sets of words using the $TF\text{-}IDF$ representation. Each text is thus considered as being the equivalent of a client. The text is constituted by a set of sentences which is associated with a date (its position in the text). Finally the set of words contained in each sentence corresponds to the set of items purchased by the client in the market basket analysis framework. Table 2 summaries the two terminologies.

Note that if order is considered in the text, a sentence remains treated as a bag of words. This adds flexibility to our approach since we argue that the order of the words within a sentence is not as important as the order of ideas in the sentences themselves.

This representation is coupled with a stemming step and a *stop-list*. The stemming step consists in replacing each word by its root word. The stop-list prevents the system from learning from noisy words such as "*the, a*".

Some words are discarded by considering the entropy of each stem over the corpus. This method eliminates words that could skew the classifier since they are not discriminant enough. Moreover, this method allows us to apply low supports in the sequential pattern discovery without deteriorating results. For this purpose, a user-defined threshold is considered. For each word w, we consider its entropy $H(w)$ over all classes C_i defined as:

$$H(w) = -\sum_{C_i}\Big[\; p(w).p(C_i|w).log(p(C_i|w)) \\ +((1-p(w)).p(C_i|\bar{w}).log(p(C_i|\bar{w})))\;\Big]$$

Usual Databases		Textual Databases
client	↔	text
item	↔	word
items/transaction	↔	sentence (set of words)
date	↔	position of the sentence

Table 2: Application of the Sequential Pattern Terminology to Textual Data

In the sequel of the paper, we use the notations introduced in Table 3. Experiments have led us to consider a threshold that eliminates about 5 to 10% (according to the Zipf law [27]) of the words.

In SPaC, the sequential patterns are extracted using a multiple minimum support strategy as done in msCBA. This means that a different support is applied for each category C_i. Contrary to msCBA, the minimum supports are defined automatically by considering 3 texts per category. This assumption leads to consider for each category C_i the following multisupport:

Notation	Meaning
$\mathcal{C} = \{C_1, \ldots, C_n\}$	set of n categories.
$C_i \in \mathcal{C}$	a given category.
$minSup_{C_i}$	user-defined minimum support for category C_i.
T	set of texts.
$T^{C_i} \subseteq T$	set of texts belonging to category C_i.
$T_{Train} = \{(C_i, T^{C_i})\}$	Training set constituted by a set of texts associated with their category.
SEQ	set of sequences found for category C_i, customer c at time t.
SP	table of sequential patterns.
$RuleSP$	table of tuples $(sp_j, C_i, conf_{i,j})$. corresponding to the sequence sp_j, the category C_i and the confidence $conf_{i,j}$ of the rule $sp_j \rightarrow C_i$.

Table 3: Notations

$$minsup_{C_i} = \frac{3 * 100}{\#\text{texts in training set } C_i}$$

In our approach the training set is divided into n training sets, one for each category. Texts are thus grouped depending on their category. Sequential pattern mining algorithms are applied separately on these n databases using the corresponding minimum supports.

For each category, frequent sequential patterns are computed and their supports are stored. The support of a frequent pattern is the number of texts containing the sequence of words.

DEFINITION 1. *Let $< s_1 \ldots s_p >$ be a sequence. The support of $< s_1 \ldots s_p >$ is defined as:*

$$supp(< s_1 \ldots s_p >) = \frac{\#texts\ matching < s_1 \ldots s_p >}{\#texts}$$

Algorithm 1 describes the SPaC sequential pattern generation. The SPAM algorithm is used through the $SPMining()$ function in order to find all frequent sequences in the transactional databases (DB) [8].

Algorithm 1: SPaC rules generation

```
Data   : $T_{Train}$ : the training Set
         $\{minSup_{C_i}\}$: the set of minimum supports for each
         category $C_i$
Result: SP: the set of sequential patterns
begin
    $SEQ \leftarrow \emptyset$;  $cust \leftarrow 0$;  $date \leftarrow 0$;
    foreach Category $C_i \in \mathcal{C}$ do
        foreach Text $T_j \in T^{C_i}$ do
            foreach Sentence $S_k \in T_j$ do
                $V_s$=TFIDF( Stem($S_k$)); // Compute the
                $TF\text{-}IDF$ vector of the sentence
                for ($s = 0; s <| V_s |; s++$) do
                    if $V_s[s] > 0$ then
                        $SEQ[C_i][cust][date]$.additem(s);
                date++;
            $date \leftarrow 0; cust++$;
        $cust \leftarrow 0$;
    foreach Category $C_i \in C$ do
        SP[$C_i$]=SPMining(SEQ[$C_i$],$minSup_{C_i}$);
end
```

For instance, the following frequent patterns have been extracted from the category "Purchasing-Logistics" of our French database:

< (cacao) (ivoir) (abidjan)>
< (blé soja) (maï)>
< (soj)(blé lespin victor)(maï soj)(maï)(grain soj)(soj)>

The first sequential pattern means that texts contain words *cacao* then *ivoire* then *Abidjan* in three different sentences. The second sequential pattern means that texts contain the words *blé* and *soja* in the same sentence and then *maïs*[1] (*mai*). The third sequential pattern means the word *maï* occurs in two successive sentences before the word *grain*.

Note that the use of sequential patterns allows the apparition of a word several times in the text, contrary to association rules. Moreover, some frequent co-occurrences can be identified with sequential patterns mining.

4.2 From Sequential Patterns to Categories

Once sequential patterns have been extracted, the goal is to derive a categorizer from the obtained rules. This is done by computing for each rule in a category the corresponding *confidence*. For each sequential pattern $< s_1 \dots s_p >$ found for the category C_i, a rule γ is computed:

$\gamma :< s_1 \dots s_p >\rightarrow C_i$.

This rule means that if a text contains s_1 then s_2 ... then s_p then it will belong to the category C_i. Each rule is associated with its confidence, indicating to which extent the sequential pattern is characteristic to this category:

$$conf(\gamma) = \frac{\#\text{texts from } C_i \text{ matching}_{<s_1 \dots s_p>}}{\#\text{texts matching}_{<s_1 \dots s_p>}}$$

Algorithm 2: SPaC-C Method

```
Data   : $T_{Test}$: the test Set
         KFS: the K First Satisfied Parameters
         SP: the sequential Pattern Table from SPaC-RG
begin
    $nb \leftarrow 1$ ;
    foreach Category $C_i \in C$ do
        foreach $sp_j \in SP[C_i]$ do
            $RuleSP[nb] \leftarrow (sp_j, C_i, conf(sp_j \rightarrow C_i))$ ;
            $nb++$ ;
    Sort $RuleSP$ by rule confidence and size of sequence ;
    $nfs \leftarrow 0$;  $classable \leftarrow 0$;
    foreach Text $T_k \in T_{Test}$ do
        foreach rule ($sp_j \rightarrow C_i$)$\in RuleSP$ do
            if $T_k$ supports $SP_j$ then
                $T_k$.score[$C_i$]++;  $classable \leftarrow 1$;  $nfs$++ ;
                if $nfs \geq KFS$ then break
        if $classable$ then
            Set $T_k$ to the Most Valued Category;
        classable $\leftarrow$ 0;  nfs $\leftarrow$ 0 ;
end
```

Rules are sorted depending on their confidence and the size of the associated sequence. When considering a new text to be classified, a simple categorization policy is applied: the K rules having the best confidence and being supported are applied. The text is then assigned to the class mainly obtained within the K rules. This method is the same as the majority voting in [10]. If two categories obtain the same score, a random choice is carried out. This prevents the system from always choosing the same category. The SPaC classifier step (SPaC-C) is described in Algorithm 2.

5. EXPERIMENTS

Experiments are led on three databases. The two first ones are the well-known english databases 20 Newsgroups and Reuters [12]. The third database describes French texts (news) with 8239 texts divided into 28 categories.

Experiments compare our approach to the results obtained with CBA and SVM. Table 4, 5 and 6 detail these results. Comparisons are based on the F_β measure [29]. This measure allows us to combine recall and precision for a global evaluation. The F_β measure is thus more relevant than accuracy since accuracy does not take into account the case when a text is not classified. This leads to consider that classifying no text (thus never making errors) would have an accuracy near 100%! Accuracy has been taken as the reference measure in [10, 19]. However, for the reasons presented above, we argue that this is not as relevant as relying on recall and precision, as mentioned in [29]

DEFINITION 2. *The F_β measure is defined as follows:*

$$F_\beta = \frac{(\beta^2 + 1)\pi_i \rho_i}{\beta^2 \pi_i + \rho_i}$$

[1]In French, *blé* means *wheat* and *maïs* stands for *corn*

where ρ stands for the recall, and π stands for the precision. This measure is computed for each class C_i.

Definition 3. *Precision and recall are defined as follows:*

$$\pi_i = \frac{TP_i}{TP_i + FP_i} \quad , \quad \rho_i = \frac{TP_i}{TP_i + FN_i}$$

where TP_i, FP_i, FN_i stand respectively for the number of texts in class well classified i (True Positive), the number of texts put in class C_i by error (False Positive), the number of texts put in a different class from i by error (False Negative).

Accuracy is defined as follows:

Definition 4. *Accuracy:*

$$Accuracy_i = \frac{TP_i + TN_i}{TP_i + TN_i + FP_i + FN_i}$$

In order to evaluate the classifier for all classes, we consider *Micro-averaging* (μ) and *Macro-averaging* (M) [29, 34] defined as follows:

Definition 5. *Macro-averaging and Micro-averaging*

$$\hat{\pi}^M = \frac{\sum_{i=1}^{|C|} \hat{\pi}_i}{|C|}, \hat{\rho}^M = \frac{\sum_{i=1}^{|C|} \hat{\rho}_i}{|C|}$$

$$\hat{\pi}^\mu = \frac{\sum_{i=1}^{|C|} VP_i}{\sum_{i=1}^{|C|}(VP_i + FP_i)}, \hat{\rho}^\mu = \frac{\sum_{i=1}^{|C|} VP_i}{\sum_{i=1}^{|C|}(VP_i + FN_i)}$$

Micro-averaging gives the same importance to each document, contrary to *macro-averaging* which computes the average class by class (puting thus more importance to small categories).

In this paper, we consider that recall and precision have the same importance. We have thus: $\beta = 1$ for the F_β measure.

	SPaC	msCBA	SVM
$F1^M$	0.444	0.367	**0.485**
$F1^\mu$	**0.487**	0.401	0.486
Accuracy	0.962	0.956	**0.969**
Parameters	multisupp.	multisupp. = 0.7%	C = 1

Table 4: Comparison of results by SPaC, msCBA and SVM on French news. Training Set = 33%

Results for SVM are obtained using a linear kernel (no better result is provided by a more complex kernel) with SVMLight [15]. The supports chosen here are the values providing the best results while remaining computable (too low supports lead to a very time-consuming application). SPaC is applied with $K = 10$ rules taken into account when classifying new documents.

In this work, we argue that getting understandable knowledge is as important, and even more important than getting the most accurate classifier. For this reason, comparisons are essentially studied between CBA and SPaC. CBA is tested with the msCBA version of the algorithm, using the best results we have obtained when testing different supports. The results show that SPaC is always better than CBA when considering the macro-average. This is due to the fact that SPaC obtains good results for every category while CBA obtains very good results for the categories having many texts and low results for little categories. SPaC is thus more efficient when dealing with a difficult learning task. For this reason, SPaC is better than CBA and similar to SVM for French texts, and is better than SVM when dealing with the 20 Newsgroups database.

	SPaC	msCBA	SVM
$F1^M$	0.219	0.082	**0.500**
$F1^\mu$	0.591	0.679	**0.840**
Accuracy	0.990	0.992	**0.996**
Parameters	multisupp.	multisupp. = 1%	C = 1

Table 5: Comparison of results by SPaC, msCBA and SVM on English texts (REUTERS)

	SPaC	msCBA	SVM
$F1^M$	**0.463**	0.423	0.423
$F1^\mu$	**0.502**	0.436	0.455
Accuracy	**0.947**	0.941	0.941
Parameters	multisupp.	multisupp. = 3.5%	C=1

Table 6: Comparison of results by SPaC, msCBA and SVM on English texts (20 Newsgroups). Training Set = 33%

In Fig. 1 we compare the results obtained for the $F1$ measure regarding the number of rules considered for the choice of the class. Experiments have shown that $K = 10$ provides good results (similar to [10] where best results correspond to $maxrules = 9$). Experiments are done (1) with order: the sequential pattern (sequence) is supported by the text (2) without order: the sub-sequences are supported by the text in an unspecified order. The corresponding sentences are in the document but they are not ordered as in the sequential pattern. We can notice that results are always better when order is taken into account.

6. CONCLUSION AND FURTHER WORK

In this paper, we address the problem of text categorization using sequential patterns. In our framework, texts are represented by *TF-IDF* vectors, and each category is associated with a set of sequential patterns. When classifying new data, a text is matched to a category depending on the number of sequential patterns holding. The corresponding category is determined using a majority voting. Even if SVM have proven to be efficient in such a task, we argue that it is very important to provide users with understandable knowledge about their data. In this framework, sequential patterns

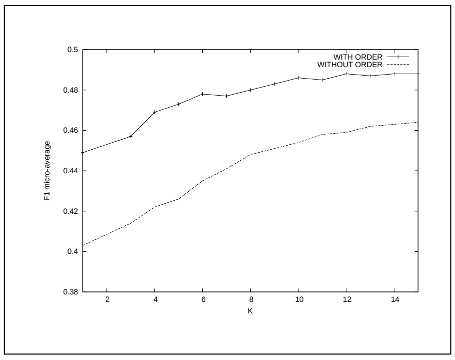

Figure 1: SPaC: $F1$ in function of the number of rules considered

are well-adapted. They provide rules that are used for the classification. We show that this approach is efficient and relevant, in particular when SVM do not perform well.

Moreover, the method we propose is simple and adaptable to drifting concepts since it is possible to update sequential patterns without performing the whole process using incremental sequential patterns mining [23]. This possibility is of high importance for text categorization, in particular for the automatic analysis of news which is a very fast variable area. This is thus a first step towards On-Line Classification Process (OLCP) using sequential patterns.

Future works include the integration of our approach on different foreign languages, in order to determine how order is important for each language. We are working on the automatic definition of the best number number of rules to take into account (the K parameter of our method). Our approach may also be enhanced by mining generalized sequential patterns [4]. This framework allows to integrate time constraints as shown in [24]. Finally, we aim at integrating muti-level sequential patterns, as proposed in [9]. This allows indeed to keep very specific rules without damaging the classifier performances. In this framework, we argue that it is interesting to build a very compact set of rules (rules where the left-hand part is as short as possible). For this purpose, we aim at extending works on $\delta-$free sets [11] to sequential patterns.

7. REFERENCES

[1] R. Agrawal, R. J. B. Jr., and R. Srikant. Athena: Mining-based interactive management of text databases. In *Extending Database Technology*, pages 365–379, 2000.

[2] R. Agrawal and R. Srikant. Fast Algorithms for Mining Generalized Association Rules. In *Proc. of the 20th Int. Conf. on Very Large Databases (VLDB'94)*, September 1994.

[3] R. Agrawal and R. Srikant. Mining Sequential Patterns. In *Proc. of the 11th Int. Conf. on Data Engineering (ICDE'95)*, Tapei, Taiwan, March 1995.

[4] R. Agrawal and R. Srikant. Mining sequential patterns. In *Eleventh Int. Conf. on Data Engineering*, pages 3–14. IEEE Computer Society Press, 1995.

[5] K. Ali, S. Manganaris, and R. Srikant. Partial classification using association rules. In *Knowledge Discovery and Data Mining*, pages 115–118, 1997.

[6] M.-L. Antonie and O. Zaiane. Text document categorization by term association. In *IEEE International Conference on Data Mining (ICDM'2002)*, pages 19–26, 2002.

[7] M.-L. Antonie and O. Zaiane. An associative classifier based on positive and negative rules. In *9th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD-04)*, pages 64–69, 2004.

[8] J. Ayres, J. Gehrke, T. Yiu, and J. Flannick. Sequential pattern mining using bitmaps. In *Proc. of the 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining.*, 2002.

[9] E. Baralis, S. Chiusano, and P. Garza. On support thresholds in associative classification. In *Proc. of the 2004 ACM Symposium on Applied Computing (SAC)*, pages 553–558, 2004.

[10] E. Baralis and P. Garza. Majority classification by means of association rules. In *7th European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pages 35–46, 2003.

[11] B. Cremilleux and J. Boulicaut. Simplest rules characterizing classes generated by delta-free sets. In *Proceedings of the 22nd BCS SGAI International Conference on Knowledge Based Systems and Applied Artificial Intelligence ES 2002*, pages 33–46. Springer-Verlag, 2002.

[12] S. Hettich and S.D.Bay. *The UCI KDD Archive. [http://kdd.ics.uci.edu]*. Irvine, CA: University of California, Department of Information and Computer Science., 1999.

[13] M. Iwayama and T. Tokunaga. Cluster-based text categorization: a comparison of category search strategies. In *Proc. of SIGIR-95, 18th ACM Int. Conf. on Research and Development in Information Retrieval*, pages 273–281. ACM Press, 1995.

[14] D. Janssens, G. Wets, T. Brijs, K. Vanhoof, and C. G. Adapting the cba-algorithm by means of intensity of implication. In *Proc. of the First Int. Conf. on Fuzzy Information Processing Theories and Applications*, pages 397–403, 2003.

[15] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In *Proc. of ECML-98, 10th European Conf. on Machine Learning*, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

[16] V. Kumar and et al, editors. *Classification Using Association Rules: Weaknesses and Enhancements*, 2001.

[17] B. Lent, R. Agrawal, and R. Srikant. Discovering trends in text databases. In *Proc. 3rd Int. Conf. Knowledge Discovery and Data Mining, KDD*, pages 227–230. AAAI Press, 14–17 1997.

[18] W. Li, J. Han, and J. Pei. CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules. In *Proc. 2001 Int. Conf. on Data Mining (ICDM'01)*, 2001.

[19] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Knowledge Discovery and Data Mining*, pages 80–86, 1998.

[20] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Knowledge Discovery and Data Mining*, pages 80–86, 1998.

[21] B. Liu, Y. Ma, and C. K. Wong. Improving an association rule based classifier. In *Principles of Data Mining and Knowledge Discovery*, pages 504–509, 2000.

[22] M. Maron. Automatic indexing: An experimental inquiry. *Journal of the ACM (JACM)*, 8:404–417, 1961.

[23] F. Masseglia, P. Poncelet, and M. Teisseire. Incremental mining of sequential patterns in large databases. *Data and Knowledge Engineering*, 46(1), 2003.

[24] F. Masseglia, P. Poncelet, and M. Teisseire. Pre-processing Time Constraints for Efficiently Mining Generalized Sequential Patterns. In *11th Int. Symposium on Temporal Representation and Reasoning, TIME'04*, July 2004.

[25] C. J. V. Rijsbergen. *Information Retrieval.* Butterworths, sec. edition, 1979.

[26] G. Salton and M. J. McGill. *Introduction to modern information retrieval.* 1983.

[27] G. Salton, C. Yang, and C. Yu. A theory of term importance in automatic text analysis. *Journal of the American Society for Information Science*, 36:33–44, 1975.

[28] F. Sebastiani. Machine learning in automated text categorisation. In *Proc. of ACM Computing Surveys*, volume 34, pages 1–47, 2002.

[29] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

[30] R. Srikant and R. Agrawal. Mining Sequential Patterns: Generalizations and Performance Improvements. In *Proc. of the 5th Int.Conf. on Extending Database Technology (EDBT'96)*, pages 3–17, September 1996.

[31] K. Wang, S. Zhou, and Y. He. Growing decision trees on support-less association rules. In *Knowledge Discovery and Data Mining*, pages 265–269, 2000.

[32] P. C. Wong, W. Cowley, H. Foote, E. Jurrus, and J. Thomas. Visualizing sequential patterns for text mining. In *INFOVIS*, pages 105–114, 2000.

[33] Y. Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1/2):69–90, 1999.

[34] Y. Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval Journal*, 1:69–90, 1999.