Text Categorization Using Adaptive Context Trees

Jean-Philippe Vert

Ecole normale supérieure, Département de mathématiques et applications,
45 rue d'Ulm, 75230 Paris cedex 05, France,
vert@dma.ens.fr

Abstract. A new way of representing texts written in natural language is introduced, as a conditional probability distribution at the letter level learned with a variable length Markov model called *adaptive context tree* model. Text categorization experiments demonstrates the ability of this representation to catch information about the semantic content of the text.

1 Introduction

Managing the information contained in increasingly large textual databases, including corporate databases, digital libraries or the World Wide Web, is now a challenge with huge economic stakes. The starting point of any information organization and management system is a way to transform texts, i.e. long strings of ASCII symbols, into objects adapted to further processing or operations for any particular task. Consider for example the problem of *text categorization*, that is the automatic assignment of natural language texts to predefined classes or categories. This problem received much attention recently and many algorithms have been proposed and evaluated, including but not limited to Bayesian classifiers ([1], [2], [3]), k-nearest neighbors ([4]), rule learning algorithms ([5], [6]), maximum entropy models ([7]), boosting ([8]) or support vector machines ([9], [10], [11]). All these algorithms share in common the way the initial text is processed from a long ASCII string into a series of words or word stems, and most of them carry out the classification from variants of the so-called *vector space* model ([12]) which consists in representing the initial text as a vector of frequencies of words in a given dictionary.

In spite of the impressive results obtained by some of the above algorithms on particular databases and categorization tasks it seems that these performances

2000 Mathematics Subject Classification. Primary: 68T50, secondary: 62B10, 62F35, 68T50, 94A17.
Key words and phrases: Text Modeling, Text Categorization, Mean Kullback Risk, Context-Tree Model

degrade as the database becomes more general and the task less specific. As a result such apparently easy tasks as filtering and classifying electronic e-mails into personal mailboxes remain non-trivial because of the poorly-formatted nature of such texts and the variations in the language used and the topics.

One of the reasons underlying these difficulties is the huge size of the set of possible words compared to the size of each text and the number of texts available for training the classifiers. This leads to large variations between texts inside of a category in terms of vector space representations, and to difficult statistical estimations during the training period. Not surprisingly support vector machines outperform most "classical" classification methods ([9]) because of their ability to deal with such issues.

This paper is an attempt to forget for a while the vector space model and consider alternative ways of extracting informations from natural language texts. Instead of parsing a text into tokens (words, word stems...) we just consider it as series of letters and estimate a letter-generating source model, i.e. a conditional probability of emitting a letter knowing the past, that "fits" the text correctly. The model estimation is done by an algorithm called *adaptive context trees* studied in [13] and produces a new representation of a text as a *context tree model*, which can be seen as a variable length Markov model. In order to study the pertinence of this representation a text classification algorithm is developed and tested. Encouraging results suggest that this representation might be able to "catch" features correlated with the semantic content of the text but not based on the words.

This paper is organized as follows. In Sect. 2 we highlight the general trade-off between the richness of a representation and the difficulty to estimate it, which motivates our representation introduced in Sect. 3. A classification algorithm is derived in Sect. 4 and experimental results appear in the following sections.

2 A Trade-Off in Representation

A digital text written in natural language is basically a series of bytes which has to be processed and transformed into a representation adapted to further operations such as text classification. A commonly used procedure consists in first rewriting it as a string of elements of a finite alphabet $\mathcal{A}$, e.g. a dictionary of words, word stems, tokens or letters, and then representing the text as a vector whose coordinates are the numbers of occurrence of each element of $\mathcal{A}$ in the pre-processed string. Depending on the alphabet $\mathcal{A}$ different situations might arise:

- If $\mathcal{A}$ is very large (think of a dictionary of all possible words for English texts, which typically contains several tenths of thousands of words) the semantic information contained in the vector space representation is known to be very rich, but the vectors corresponding to two related texts might be completely different because of the small size of every single text compared to the size of the dictionary. In other words the representation is unstable because it is statistically difficult to estimate any hidden distribution in a large space from few observations.

– On the other hand if $\mathcal{A}$ is very small (think of the 26-letters Latin alphabet plus some punctuation signs) the vector space representation has the advantage of being more stable even for small texts but the dramatic drawback of containing few semantic informations. As an example the frequencies of various letters might be a good indicator to guess the language of a text (e.g. English versus French) because they are usually characteristic of the language even for small texts, but they might not be appropriate features to guess whether an English text is about politics of religion.

These remarks show that there exists a *trade-off* between the information contained in a representation and the difficulty to estimate it from a finite and possibly short text. As far as the vector space model is concerned various techniques exist in order to decrease the size of the alphabet while keeping the semantic contents of words ([14]): these techniques include word stemming, thesaurus, stop words removal, feature selection etc...

Forgetting for a while the vector space representation it is possible to observe the same balance phenomenon in an other setting : the representation of a text $\mathcal{T}$ by a letter-generating source, i.e. by a conditional probability $\mathbb{P}_{\mathcal{T}}(Y \mid X)$ where Y is a random variable on the alphabet $\mathcal{A}$ which represents the next letter to be generated and X is a random variable on $\mathcal{A}^* = \cup_{n \geq 0} \mathcal{A}^i$ (the set of finite-length strings) which represents the past sequence of letters. The idea is that such a source is characteristic of a certain category of texts, and the goal of the representation is to estimate the source from a text supposed to be generated by it.

Note that even if the alphabet is poor - think of ASCII symbols or the Latin alphabet - this ideal representation $\mathbb{P}_{\mathcal{T}}(Y|X)$ is very rich because it suffices to define a stationary process which might be assimilated to the process of writing a text in the category specific of the source. In particular it contains the stationary probability of any finite-length string, e.g. any word made of letters or even n-grams of words.

Estimating such a conditional probability from a finite-length text can be done with the help of finite-dimensional models, e.g. finite order Markov models. Such an approach leads to the same kind of balance as mentioned above in the vector space model : if the chosen model is complex (e.g. large order Markov model) then it potentialy can better mimic the unknown probability $\mathbb{P}_{\mathcal{T}}(Y \mid X)$ than simpler model, but it is much more difficult to estimate from a finite number of observations. In other words a trade-off has to be reached between the complexity of the model used to estimate $\mathbb{P}_{\mathcal{T}}(Y \mid X)$ and the risk when estimating it.

This representation as a letter-generating source is however better adapted to the trade-off quest than the vector space model because it is easier to compare models of various complexities (e.g. finite order Markov models) and chose a complexity than depends on the information available. In the next section we present an algorithm that fulfills this requirement and leads to an *adaptive* representation of any text as a more or less complex conditional probability.

3 Probability Estimation through Adaptive Context Trees

We consider a text as a deterministic object from which statistical information can be learned through sampling procedures. In order to get an independent and identically distributed (i.i.d) sample $(X_i, Y_i)_{i=1,\dots,N}$ we propose to follow N times the following procedure : randomly chose a position in the text with a uniform prior, let Y be the letter occuring at the selected position and let X be the string made of the letters preceding Y backward to the beginning of the text.

In order to estimate $\mathbb{P}_{\mathcal{T}}(Y|X)$ from the resulting i.i.d. sample $(X_i, Y_i)_{i=1,\dots,N}$ we introduce a family of finite-dimensional conditional probability distributions which consist in splitting the space of past strings X into a finite number of cells and letting Y depend on X only through the cell X belongs to. One natural way to design such a splitting is to let Y depend on X only through one suffix: this covers in particular the case of fixed-order Markov models but more generally leads to *incomplete tree models* as defined in [13]. We refer to this paper for a more detailed presentation of incomplete tree models and just recall here the main definitions.

An *incomplete tree* is a set of strings $\mathcal{S} \subset \mathcal{A}^*$ such that any suffix of any string of $\mathcal{S}$ be also in $\mathcal{S}$ (a suffix of a string x_1^l is any string of the form x_i^l, with $i \in [1, l]$ including the empty string λ of length 0). For any integer D we let $\mathcal{S}_D$ be the set of incomplete trees made of strings of lengths smaller than D. A suffix functional $s_{\mathcal{S}}$ associated with any incomplete tree $\mathcal{S}$ maps any finite sequence $x \in \mathcal{A}^*$ into the longest element of the tree that is a suffix of x. Hence a partition of $\mathcal{A}^*$ is associated to any incomplete tree.

Let Σ denote the simplex $\Sigma = \{\theta \in [0,1]^{|\mathcal{A}|}, \sum_{i=1}^{|\mathcal{A}|} \theta_i = 1\}$. Together with a parameter $\theta \in \Sigma^{\mathcal{S}}$ an incomplete tree defines a conditional probability distribution as follows:

$$\forall (x,y) \in \mathcal{A}^* \times \mathcal{A} \qquad \mathbb{P}_{\mathcal{S},\theta}\left(Y = y \,|\, X = x\right) = \theta(s_{\mathcal{S}}(x))_y \ . \tag{1}$$

In other words the conditional probability of Y knowing the past X only depends on a particular suffix of X as defined by the context tree $\mathcal{S}$. Now we see that the number of possible models is very large, ranging from very simple models with few parameters (e.g. the empty string only, which is equivalent to an i.i.d. model for letters) to very complex models when the tree size is large. The true unknown conditional probability $\mathbb{P}_{\mathcal{T}}(X \,|\, Y)$ is probably better represented by complex models, but the parameter estimation based on a finite training set is easier with simple low-dimensional models.

At this step it is necessary to define precisely the notions of "distance" between probability and of "estimation risk". A natural measure of similarity in the space of conditional probability distributions is the *conditional Kullback-Leibler divergence* or *conditional relative entropy* ([15, p. 22]) defined by:

$$\mathcal{D}\left(\mathbb{P}(.|.)\,||\,\mathbb{Q}(.|.)\right) = \sum_{x\in\mathcal{A}*} \mathbb{P}(x) \sum_{y\in\mathcal{A}} \mathbb{P}(y\,|\,x) \log \frac{\mathbb{P}(y\,|\,x)}{\mathbb{Q}(y\,|\,x)} \quad , \tag{2}$$

where the first sum should be understood as an expectation.

Now suppose we have an i.i.d. set $\{(X_i, Y_i) = Z_i\,;\, i = 1, \dots, N\}$ sampled from the joint probability $\mathbb{P}_{\mathcal{T}}$, and an estimator $\hat{\mathbb{P}}_{Z_1^N}(.|.)$ of the conditional distribution $\mathbb{P}_{\mathcal{T}}(Y|X)$. Then it is natural to measure the risk of the estimator $R(\hat{\mathbb{P}})$ by averaging the conditional relative entropy with respect to the i.i.d. sample used for estimation:

$$R(\hat{\mathbb{P}}) = \mathbb{E}\left[\mathcal{D}\left(\mathbb{P}_{\mathcal{T}}(.|.)\,||\,\hat{\mathbb{P}}_{Z_1^N}(.|.)\right)\right] \quad . \tag{3}$$

Following the work in [13] the i.i.d. sample Z_1^N can be used to build an aggregated estimator $\mathbb{G}_N(Y\,|\,X)$ with the following risk bound:

Theorem 1. *(Vert, [13])*
Let

$$\chi_N = \log(N + a) \quad , \tag{4}$$

and

$$\begin{aligned} \beta_N &= \frac{1}{\chi_N - 1}\left(\sqrt{1 - (\chi_N - 1)\left(2 - \frac{\log \chi_N}{\chi_N}\right)} - 1\right) \\ &\underset{N\to+\infty}{\sim} \frac{\sqrt{2\log\log N}}{\log N} \quad . \end{aligned} \tag{5}$$

For any conditional distribution $\mathbb{P}_{\mathcal{T}}$ and any maximal depth $D \in \mathbb{N}$ the aggregated estimator using a Gibbs mixture at inverse temperature β_N (see definition in [13]) satisfies:

$$R(\mathbb{G}_N) \leq \inf_{\mathcal{S}\in\mathcal{S}_D, \theta\in\Sigma^{\mathcal{S}}} \left\{R(\mathbb{P}_{\mathcal{S},\theta}) + \frac{|\mathcal{S}|C_N}{N+1}\right\} \quad , \tag{6}$$

with

$$C_N = \left(\sqrt{(1 + \log|A|)\,\beta_N^{-1}} + \sqrt{|A| - 1}\right)^2 \left(1 + \frac{1}{N-2}\right) \quad . \tag{7}$$

The interesting property of this estimator, whose exact definition and efficient implementation are discussed in [13] and quickly summed up in Sect. 11, is its capacity to find one particular model in the family which offers a good trade-off between precision (as expressed by the term $\inf_\theta R(\mathbb{P}_{\mathcal{S},\theta})$) and difficulty of estimation (as expressed by the additional term $Cte \times |\mathcal{S}| / (N+1)$). It is called adaptive because it estimates any particular distribution $\mathbb{P}_\mathcal{T}$ at a good rate without requiring any information about it, and adapts to its complexity.

These theoretical results suggest the following procedure to represent a text $\mathcal{T}$:

- Sample an i.i.d. set Z_1^N from the text by repeatedly choosing a position with a uniform prior on the text invovled.
- Use this sample to train an adaptive context tree estimator which we denote by $\hat{\mathbb{P}}_\mathcal{T}$.

4 Text Categorization

We can now describe a text categorization algorithm. In the classical setting of text categorization a so-called "learning set" of texts is given to the classifier together with the categories they belong to. The classifier task is to learn from this set a rule that assigns one (or eventually several) category to any new text. The classifier performance is measured by its ability to correctly classify texts belonging to a so-called "test set".

The representation of a text as a conditional probability presented in Sect. 3 can be extended to the representation of a category : it suffices to sample the data used to train the estimator from any text belonging to the category $\mathcal{C}$ in the training set in order to obtain a representation of the category as a conditional probability $\hat{\mathbb{P}}_\mathcal{C}$.

Comparing a given text x_1^l to a category representation $\hat{\mathbb{P}}_\mathcal{C}$ is naturally done through the following notion of score:

Definition 1. *For any given text $\mathcal{T} = x_1^l$ let $\mathbb{P}_\mathcal{T}(X,Y)$ be the joint probability distribution on $\mathcal{A}^* \times \mathcal{A}$ defined by uniformly choosing an index i in $1, \dots, l$ and setting $(X,Y) = (x_1^{l-1}, x_l)$. The* score *of the category $\mathcal{C}$ w.r.t. the text $\mathcal{T}$ is defined by :*

$$s_\mathcal{T}(\mathcal{C}) = \mathbb{E}_{\mathbb{P}_\mathcal{T}} \log \hat{\mathbb{P}}_\mathcal{C}(Y \mid X) \ . \tag{8}$$

For a given text it is well known that such a score is maximal when $\hat{\mathbb{P}}_\mathcal{C}$ is a.s. equal to $\mathbb{P}_\mathcal{T}$, and is related to the relative Kullback-Leibler divergence through the following equality:

$$s_\mathcal{C}(\mathcal{T}) = -\mathcal{H}\left(\mathbb{P}_\mathcal{T}(.|.)\right) - \mathcal{D}\left(\mathbb{P}_\mathcal{T}(.\,|\,.)\,||\,\mathbb{P}_\mathcal{C}(.\,|\,.)\right) \ , \tag{9}$$

where $\mathcal{H}$ denotes the conditional Shannon entropy :

$$\mathcal{H}\left(\mathbb{P}(.\,|\,.)\right) = \sum_{(x,y)} \mathbb{P}(x,y) \log \frac{1}{\mathbb{P}(y \mid x)} \ . \tag{10}$$

This equality shows that comparing the scores of two different categories w.r.t. to a given text $\mathcal{T}$ is equivalent to comparing the relative Kullback-Leibler divergence of the corresponding representations w.r.t. $\mathbb{P}_{\mathcal{T}}$. This suggests to use this score not as a universal measure of similarity between a text and a category but rather as a way to compare two or more categories w.r.t. a text, in order to remove the influence of the entropy term.

By the law of large numbers it is reasonable to estimate the score of a category w.r.t. a text by creating an i.i.d. sample Z_1^K sampled from the joint law $\mathbb{P}_{\mathcal{T}}$, as explained in definition 1, and to compute the empirical score :

$$\hat{s}_{\mathcal{C}}(\mathcal{T}) = \frac{1}{K} \sum_{i=1}^{K} \log \hat{\mathbb{P}}_{\mathcal{C}}(Y_i \,|\, X_i) \ . \tag{11}$$

The categorization itself should then depend on the precise task to carry out. We present in the following sections two experiments which involve two different categorizers:

- On the `Reuters-21578` collection (Sect. 6) we create a series of binary classifiers corresponding to each category, in order to compute recall-precision curves for each category. This means that we need to sort the texts in the test set by decreasing similarity with a given category. This similarity involves the difference between the score of the category and the score of a "general" category w.r.t. each text.
- On the `Usenet` database we create a classifier which maps any new text into one of the predefined category and compute the proportion of misclassified texts. This can simply be done by comparing the scores of all categories w.r.t. to the text to be classified.

5 Initial Text Processing

The theoretical framework suggests to work on a small alphabet $\mathcal{A}$ in order to get good estimates for the conditional distributions. As a result we decided to use as an alphabet the set of 26 letters used in the Latin alphabet plus an extra symbol noted $\emptyset$, resulting in an alphabet of size 27. The preprocessing of every text in the following experiments consists in the very simple following procedure:

- Each letter is turned into small cap;
- Each ASCII character that is not a letter is transformed into $\emptyset$;
- Series of consecutive $\emptyset$ are transformed into a single $\emptyset$.

Starting from a series of ASCII characters this procedures produces a series of letters of the 27-letter alphabet, with the particularity that two $\emptyset$ are never consecutive.

6 Experiment on the `Reuters-21578` Database

The `Reuters-21578` collection[1] is a dataset compiled by David Lewis and originally collected by the Carnegie group from the Reuters newswire in 1987. The "ModApte" split is used to create a training set of 9603 documents and a test set of 3299 documents. A common way to evaluate a classification algorithm on this dataset consists in building a separate classifier for each category with a "precision" parameter which can be varied to estimate the precision/recall curve. For a given category precision is the proportion of items placed in the category that are really in the category, and recall is the proportion of items in the category that are actually placed in the category. The increase of one of these variables (by changing the parameter) is usually done at the expense of decreasing the other one, and a widely-used measure to sum up the characteristics of the precision/recall curve is the *break-even point*, that is the value of precision when it equals recall.

Following this setting a graded measure of category membership for any text can be defined as follows:

- Compute a representation $\hat{\mathbb{P}}_{\mathcal{C}}$ for the category.
- Compute a representation $\hat{\mathbb{P}}_{\mathcal{G}}$ for a general text of the database (i.e. by setting $\mathcal{G}$ to be the whole database).
- Define the category membership of the text as:

$$m_{\mathcal{C}}(\mathcal{T}) = s_{\mathcal{T}}(\mathcal{C}) - s_{\mathcal{T}}(\mathcal{G}) \ . \tag{12}$$

- Classify the text $\mathcal{T}$ in the category $\mathcal{C}$ if $m_{\mathcal{C}}(\mathcal{T})$ is larger than a threshold δ.
- Adjust the precision/recall trade-off by varying the threshold δ.

As mentioned in Sect. 4 it is necessary to measure differences between scores of several categories w.r.t. a text to obtain a meaningful index. In this case we compare the difference between a precise category and the general database in order to detect texts which particularly "fit" to a category.

In order to carry out the experiment the `TITLE` and `BODY` parts of each article is used as a starting text. Following experimental results available in [13] we ran the adaptive context tree algorithm with 200,000 samples for learning the continuous parameters and 100,000 sample for selection a tree, with a maximal tree depth $D = 9$ and a penalty term $pen = 3$. These parameters were not further optimize. Table I summarizes the break-even points computed for the ten largest categories.

7 Experiment on the 20 Newsgroup Database

The second data set consists of Usenet articles collected from 20 newsgroups by Ken Lang ([16]). Over a period of time about 1000 articles were taken from each of the newsgroups, which makes an overall number of 20017 articles in the

[1] Distribution 1.0, available at `http://www.research.att.com/lewis/`

Table I. Break-even performance for 10 largest categories of Reuters-21578

Category	B-E point
earn	93
acq	91
money-fx	71
grain	74
crude	79
trade	56
interest	63
ship	75
wheat	58
corn	41

collection. Each article belongs to at least one newsgroup, and generally to only one except for about 4% of the data set. The task is to learn which newsgroup an article was posted to. In the case an article belongs to several newsgroups predicting either of them is counted as a correct prediction. The performance of the estimator trained on the learning set is measured in terms of *accuracy*, that is the proportion of correct prediction in the test set.

Contrary to the binary classification context of the Reuters database the categorizer must be able to map any new text into one out of 20 categories. In that case it makes sense to compute the scores of each category w.r.t. to a given text, and to assign it to the category having the largest score.

For each category we created a random subset of 200 texts to serve as a test set and used the remaining texts to estimate the model representation. Before running the experiment we deleted the binaries contained in some messages, and kept the `Body` part of every message as a starting text. The adaptive context tree algorithm was run with 400,000 samples for learning the continuous parameters and 200,000 sample for selection a tree, with a maximal tree depth $D = 9$ and a penalty term $pen = 3$. Like for the Reuters experiment these parameters were not further optimize.

We ran two experiments in order to show how it is possible to influence the representation by using the prior knowledge that the `Subject` line might be more category-specific than the `Body` part. In the first experiment the `Subject` line was simply discarde, and in the second one it was added to the `Body` and the probability of drawing a letter from the `Subject` was ten times larger than the probability of drawing a letter from the `Body`.

Table II shows the average accuracy obtained on each newsgroup and globally for both experiments.

8 Automatic Text Generation

In order to give a flavor of the information contained in the models estimated to represent various categories we used them to randomly generate small texts. Table III shows texts generated from models representing three different categories

Table II. Accuracy for the 20 Newsgroup data set

Newsgroup	No Subject	Subject favored
alt.atheism	81	86
comp.graphics	80	89
comp.os.ms-windows.misc	81	86
comp.sys.ibm.pc.hardware	80	86
comp.sys.mac.hardware	84	92
comp.windows.x	85	92
misc.forsale	73	82
rec.autos	90	96
rec.motorcycles	91	93
rec.sport.baseball	93	94
rec.sport.hockey	95	96
sci.crypt	93	96
sci.electronics	90	94
sci.med	92	95
sci.space	93	95
soc.religion.christian	92	95
talk.politics.guns	88	91
talk.politics.mideast	91	94
talk.politics.misc	70	73
talk.religion.misc	65	73
Total	85.4	90.0

in the Usenet database. One can observe that many English words appear, but that many features including stylistic ones are caught in the models. For instance the level of language looks much higher in the discussion group about politics (with many long words) than in the group about baseball (which contains many "stop words").

9 Discussion

The Reuters data set is known to be well adapted to classification algorithms based on words only. As mentioned in [11] and [3] categories like "wheat" or "corn" are efficiently predicted by testing the presence of a very small number of terms in the text : a simple classifier which satisfies a document according to whether or not it contains the word wheat has an accuracy of 99.7% on the corresponding category. In such a situation our result are not surprisingly pretty bad, and much worst than results reported by other algorithms. For classes like "acq" with a more abstract concept our results are near the average of classical methods based on words as reported in [9]. In the whole the results we present are worse than results reported for state-of-the-art classifiers, but are comparable to results reported for naive Bayes classifiers.

The 20 Newsgroup database is known to be less formatted and many categories fall into confusable clusters. Even though comparison with other reported

Table III. Automatic text generation

`talk.politics.mideast`: oving race her shaights here were vii agraph associattements in the greeks who be neven exclub no bribedom of spread marinary s trooperties savi tack acter i ruthh jake bony continues is a person upi think veh people have presearchat p notect he said then proceeded in tulkan arabs the world wide us plotalking it and then he syn henrik armenian ten yesterday party com ten you conspik kill of siyalcincould palestiness and they thuma the interviewin also the serious adl the jewish victims and ms
`soc.religion.christian`: g much direciate clear the ances i did the son that must as a friend one jerome unimovingt ail serving are national atan cwru evid which done joseph in response of the wholeleaseriend the only churches in nead already first measure how uxa edu or forth crime the result the sin add and they christian under comes when so get is wrong i wonder does in heaven and neglish who was just sufferent to record telnk stated and statementsell which houserve that the committed ignore the other reading that
`rec.sport.baseball`: rschbecaust what is necessarily anyour defense in the dl vpecifiedu finger who two hitter and nextrap it is institut theoryl i cards win at aaa his lavinelatio statistic hitey loses upset himself a try team he pretty ll scott leyland in the words future current be internetics cornell edu edwards year for open t i am no fielding to be but bell asday still in the totalentine nixon kiddibbly anothis year hankee most a l reseats of ronto lose is in article price in revenuestion h is ba of basick andr

results is difficult because of the non-standardized splitting procedure the performance of our algorithm looks not far from the state-of-the-art level of accuracy (around 90%).

These results suggest that looking at the words is not the only way to get information on the semantic content of a text or at least on the category it belongs to. Even though looking at the distribution of characters is intuitively more related to the style of a text than to its meaning our experiments show that to some extent this intuition is false.

One positive point in our approach is that no dictionary, stemming algorithm or word selection procedure is required as a text is just considered as a sequence of letters. This results in two interesting features:

- It might be a good approach to languages like Chinese or Japanese where the parsing and indexing by words is less natural and more difficult than in English;
- Once the models are learned the categorization of a text is very quick as no preprocessing or indexing is required.

10 Conclusion

We presented a new way of representing texts written in natural language through adaptive statistical estimators. In order to have good statistical properties we decided to work on the character level, which might look very challenging as it is usually considered that representing a text as a series of words or word stems is the best approach possible. However results obtained for text classifiers based on this representation suggests that it is still able to catch semantic contents.

This low-level representation is clearly not optimized for the particular task of text categorization. Encouraging results suggest however different fields of investigation in the future, including:

– the development of other representations than the conditional probability of a character knowing the past, which should be task-oriented;
– the combination of this approach with word-based state-of-the-art algorithms for text categorization, with the hope that the features used by both approaches be sufficiently different to generate a gain in performance.

11 Annex: the Adaptive Context Tree Estimator

This annex is to describe very briefly the procedure we follow to build the representation of a category, that is a conditional probability. The reader should refer to [13] for further details.

The parameters to set are:

– the maximal depth D of the tree models family;
– a penalty term *pen* which represents the cost of a node.

The algorithm is fed with two independent training sets $\mathcal{Z}_1$ and $\mathcal{Z}_2$ of size N_1 and N_2 respectively used to estimate the continuous parameters and to select a model. These sets are used to update counters attaches to each node $s \in T = \cup_{i=0}^{D} \mathcal{A}^i$ of a context tree of depth D as follows:

$$\forall i \in \mathcal{A} \qquad a_s^i = \sum_{(X,Y)\in\mathcal{Z}_1} \mathbf{1}\left(s \text{ is a suffix of } X \text{ and } Y = i\right) \ , \tag{13}$$

$$\forall i \in \mathcal{A} \qquad b_s^i = \sum_{(X,Y)\in\mathcal{Z}_2} \mathbf{1}\left(s \text{ is a suffix of } X \text{ and } Y = i\right) \ , \tag{14}$$

$$n_s = \sum_{(X,Y)\in\mathcal{Z}_1} \mathbf{1}\left(s \text{ is a suffix of } X\right) \ . \tag{15}$$

A functional w is then recursively computed on each node of the context tree, starting from the leaves and going back to the root:

$$\begin{cases} \text{If } l(s) = D \quad w(s) = pen + \sum_{y \in \mathcal{A}} b_s^y \log \dfrac{a_s^i + 1}{n_s + |\mathcal{A}|} \ , \\ \text{If } l(s) < D \quad w(s) = pen + \max_{\mathcal{N} \subset \mathcal{A}} \Bigg\{ \sum_{j \in \mathcal{N}} w\,(js) \\ \qquad\qquad + \sum_{y \in \mathcal{A}} \left(b_s^y - \sum_{j \in \mathcal{N}} b_{js}^y \right) \log \dfrac{a_s^i - \sum_{j \in \mathcal{N}} a_{js}^i + 1}{n_s - \sum_{j \in \mathcal{N}} n_{js} + |\mathcal{A}|} \Bigg\} \ . \end{cases} \tag{16}$$

At every step the sons selected in the subset $\mathcal{N}$ of the second equation are marked. The largest incomplete tree model made of marked nodes is then selected as the estimator $\hat{\mathbb{P}}$, together with parameters (see Sect. 3) defined by:

$$\theta(s)_i = \frac{a_s^i + 1}{n_s + |\mathcal{A}|} \ . \tag{17}$$

References

1. Lewis, D. D.: Naive (Bayes) at forty: The independence assumption in information retrieval. In Proceedings of ECML-98, 10th European Conference on Machine Learning (C. Nédellec and C. Rouveirol, eds.), no. 1398 in Lecture Notes in Computer Science, (Chemnitz, DE), Springer Verlag, Heidelberg, DE, (1998), 4-15.
2. McCallum, A. K., Nigam, K.: A comparison of event models for naive Bayes text classification. In Proceedings of the AAAI/ICML-98 Workshop on Learning for Text Categorization, (1998), 41-48.
3. Nigam, K., McCallum, A. K., Thrun, S., Mitchell, T. M.: Text classification from labeled and unlabeled documents using EM. Machine Learning. **39**, no.2/3, (2000), 103-134.
4. Yang, Y.: An evaluation of statistical approaches to text categorization. Information Retrieval. **1**, no. 1-2, (1999), 69-90.
5. Slattery, S., Craven, M.: Combining statistical and relational methods for learning in hypertext domains. In Proceedings of ILP-98, 8th International Conference on Inductive Logic Programming (D. Page, Ed.), no. 1446 in Lecture Notes in Computer Science, (Madison, US), Springer Verlag, Heidelberg, DE, (1998), 38-52.
6. Moulinier, I., Raškinis, G., Ganascia, J.-G.: Text categorization: a symbolic approach. In Proceedings of SDAIR-96, 5th Annual Symposium on Document Analysis and Information Retrieval, (Las Vegas, US), (1996), 87-99.
7. Nigam, K., Lafferty, J., McCallum, A. K.: Using maximum entropy for text classification. In Proceedings of the IJCAI-99 Workshop on Machine Learning for Information Filtering, (1999), 61-67.
8. Schapire, R. E., Singer, Y.: BoosTexter : A boosting-based system for text categorization. Machine Learning. **39**, no.2/3, (2000), 135-168.
9. Joachims, T.: Text categorization with support vector machines: learning with many relevant features. In Proceedings of ECML-98, 10th European Conference on Machine Learning, (C. Nédellec and C. Rouveirol, eds.), no. 1398 in Lecture Notes in Computer Science, (Chemnitz, DE), Springer Verlag, Heidelberg, DE, (1998), 137-142.

10. Dumais, S. T., Platt, J., Heckerman, D., Sahami, M.: Inductive learning algorithms and representations for text categorization. In Proceedings of CIKM-98, 7th ACM International Conference on Information and Knowledge Management, (G. Gardarin, J. C. French, N. Pissinou, K. Makki and L. Bouganim, eds.), (Bethesda, US), ACM Press, New York, US, (1998), 148-155.
11. Joachims, T.: Transductive inference for text classification using support vector machines. In Proceedings of ICML-99, 16th International Conference on Machine Learning, (I. Bratko and S. Dzeroski, eds.), (Bled, SL), Morgan Kaufmann Publishers, San Francisco, US, (1999), 200-209.
12. Salton, G., Wong, A, Yang, C. S.: A vector space model for automatic indexing. Communications of the ACM. **18**, no. 11, (1975), 613-620.
13. Vert, J.-P.: Adaptive context trees and text clustering. Preprint DMA-00-05, Département de mathématiques et applications, Ecole normale supérieure de Paris. Available at `http://www.dma.ens.fr/edition`, (2000), 1-27.
14. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retriecal. ACM Press, (1999).
15. Cover, T. M., Thomas, J. A.: Elements of Information Theory. Wiley, (1991).
16. Joachims, T.: A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In Proceedings of ICML-97, 14th International Conference on Machine Learning (D. H. Fisher, ed.), (Nashville, US), Morgan Kaufmann Publishers, San Francisco, US, (1997), 143-151.