Document Classification for Computer Science Related Articles

May 15, 2002

Ana Fuentes Martínez
Flavius Gruian

Introduction to Natural Language Processing and Computational Linguistics

Document classification based on content is a general task that appears in a wide spectrum of tasks from text retrieval to automatic browsing tools or database maintenance. It is expensive when performed manually –even though this is still the most accurate method. In this project we have implemented a simple probabilistic document classifier based on corpus similarity from a predefined topic hierarchy. It is based on the Naïve Bayes method with multinomial sampling well suited for related text classes. The method was evaluated with a set of articles from three different journals in Computer Science. The experiments showed that Naïve Bayes classifier was able to correctly determine the class of 93% of the abstracts in the test set, for well balanced training data.

1. Introduction

The question of whether an automatic classifier performs better than humans is a longstanding controversy. Certainly it depends on how skilled the person is, but also on the amount of documents to be classified. Generic classifiers often have poorer performance than the ones using domain-specific methods [1] but humans taggers will also find greater difficulties in categorizing wide-ranging text sets. Time and cost effectiveness issues have contributed in making automatic tools an attractive alternative and promoted research in the field.

Several machine learning techniques have been developed to automatically discern text by content. Indexing non-homogeneous documents, as the ones found on the World Wide Web is discussed in a recent survey by Y.Yang et. al. [2] It focuses on well-known algorithms (Naïve Bayes, Nearest Neighbor and First Order Inductive Learner) applied to hypertext categorization. The sparseness problem caused by large number of categories with few documents on each is examined in [3]. Estimates of the terms distributions are made by differentiation of words in the hierarchy according to their level of generality/specificity. Some practical work has been done within Web page clustering. We mention two different approaches to organize a set of documents. In [4] categories are being constructed from a set of unclassified texts. The documents are clustered together according to semantic similarity criteria. This is certainly the most frequent practice in text classification. A different idea, syntactic clustering, is discussed in [5]. Their intention is to determine whether two documents are “roughly the same” or “roughly contained” except from modifications such as formatting or minor corrections.

More homogeneous texts are often indexed in predefined categories. They are characterized by containing multiple overlapping terms across the classes. One example of this type of corpus is presented in [7]. Machine learning techniques are used to detect whether or not literary works can be classified according to the genders of their authors just by noting differences in the type of prose. The subject of the classifier proposed in [8] are bioinformatics articles. The general idea behind this work is that the features useful for distinguishing articles vary among the different levels in the hierarchy.

Classifiers can be used for either category ranking or automated category assignments (binary classification). Both require the same kind of computation and deduce the same information [9]. The type of output depends on the application. For user interaction a category ranking system would be more useful while for fully automated classification tasks category assignment is desirable.

2. Categorization Model

a) Naïve Bayes probabilistic classifier

We assume that a predefined set of text categories is given. A large set of abstracts from each of the categories conform the corpus. Each document is presumed to belong to a single category.

Based on this corpus we define a probability model for abstracts. New documents are classified following Bayes' rule by computing posterior probabilities over the categories. Naïve Bayes probabilistic classifiers are commonly used in text categorization. The basic idea is to use the joint probabilities of words and categories to estimate the probabilities of categories given a document [10]. It is easy to estimate the probability of each term by simply counting the frequency with which the target value occurs in the training data:

$$P(w_i|Class) = \frac{Occurrences\ of\ w_i\ in\ Class}{No.\ of\ Words\ in\ the\ Class(C)} \quad (1)$$

The naive part of such an approach is the assumption of word independence [10]. In other words, the probability of observing the conjunction $w_1,w_2,...,w_n$, is just the product of the probabilities for the individual attributes.

$$P(w_1, w_2, \ldots w_n|C_j) = \prod_i P(w_i|C_j) \quad (2)$$

Notice that in a Naïve Bayes learner, the number of distinct $P(w_i|C_i)$ that must be calculated is significantly smaller than all possible $P(w_1,w_2,...,w_n,|C_i)$. Despite the independence assumption is clearly incorrect –it is more probable to find the word *network* if the preceding word is *neural*– the algorithm performs well in many text classification problems.

The document *D* is assigned to the class that returned greatest probability of observing the words that were actually found in the document, according to the formula

$$Class = \arg\left[max_{C_j \in C}\left\{ P(C_j) \cdot \prod_{w_i \in D} P(w_i|C_j) \right\}\right] \quad (3)$$

b) Canonical form of the document

The first main issue involved in applying the Naïve Bayes classifier to text classification problems is to decide how to represent an arbitrary text document in terms of attribute values. Beside word independence, we made an additional reasonable assumption regarding the distribution of terms in a document; the probability of encountering a specific word in a document is independent of its position. The primary advantage is that it increases the number of examples available for each term and therefore improves the reliability of the estimates.

A Prolog written filter parses the document and produces what we call the canonical form of a text. This canonical form is actually a vector representation of the terms and their frequencies. In order to reduce the amount of data, the canonical form contains only the information that is relevant for the probabilistic measures. Punctuation marks, numbers and stopwords are removed and uppercase letters are converted to lowercase so that terms can be compared. Notice that this approach implies strictly semantic similarity, no syntactic issues are considered.

c) Bayesian Estimation with Uniform Prior

To complete the design of our leaning model we must choose a method for estimating the probability terms required by the Naïve Bayes classifier. The word frequencies can be obtained by a Bayesian Estimation with Uniform Priors, also known as Laplace's smoothing law [6]. It addresses the problem of assigning a probability distinct from zero to those terms in the document that do not appear in the corpus. It can be interpreted as enlarging the corpus with the current text. For the formula (1) we have that the size of the new vocabulary is the number of words in the class (or size of the class), *C* plus the size of the document, *N*. The word counter for w_i is increased by one because now w_i is present also in the document.

$$P(w_i|C_j) = \frac{|w_i| + 1}{C + N} \qquad (4)$$

d) The system

The system we designed has two major parts: A prolog program for parsing the new document and a Perl script for computing the probabilities and selecting the most suitable category. Evaluation of the performance was done automatically by iterating over a directory of test files and checking whether the class assignment suggested by the algorithm matched the original journal, which was apparent from the file name.

During the leaning stage, the algorithm examines all training documents to extract the vocabulary for each class and count the frequency of all terms. These corpora are also presented in the canonical form.

Later, given a new document to be classified, the probability estimates from each corpus are used to calculate the closest category according to the formula (3). The result is obtained as a ranked list of the candidate categories where the first one is consider to be correct and the others are disregarded.

3. Results and Discussion

a) Experimental set-up

The corpora used to run the experiments consisted of a large set of abstracts from computer science journals. They were divided into thee clusters according to their main topic: Digital Signal Processing, Biomedical Research and Computer Speech & Language, all collected from Academic Press Journals. The final training data contained over 7000 different significant terms i.e, without stopwords or numbers. The corpora enclosed more than 21.000 words. The classifier was tested with 60 new abstracts from the same journals. This way the correct class was known in advance and it could be compared to the one suggested by the classifier. The division between the training set and the test set was randomly chosen and several configurations were tested in order to ensure reliable results.

Two characteristics differentiate our data set compared to other papers in the field. First, we are working with rather coarse data. The different classes will include larger amount of overlapping terms which will blur the fitting measures. This is not the case for general web classifiers with rather heterogeneous document topics. Second, it is typical for technical articles to contain many very specific terms –and this is specially outstanding in the abstract. A hand-tuned classifier could make use of this information to increase the weight of this terms in the probabilities and achieve better classification rates. Each article is assigned a single category in the hierarchy.

b) Evaluation Measures

We used a four cell contingency table for each category and a set of test documents:

Table I: contingency table for a class

	documents that belong to C	documents that DO NOT belong to C
documents assigned to C	a	b
documents NOT assigned to C	c	d

The total number of test documents is

$$N = a + b + c + d$$

where

- a is the number of documents correctly assigned to this category.
- b is the number of documents incorrectly assigned to this category.
- c is the number of documents incorrectly rejected from this category.
- d is the number of documents correctly rejected from this category.

We obtained the following results:

Table II: contingency table for Digital Signal Processing

	documents that belong to Digital Signal Processing	documents that DO NOT belong to Digital Signal Processing
documents assigned to Digital Signal Processing	17	0
documents NOT assigned to Digital Signal Processing	3	40

Table III: contingency table for Biomedical Research

	documents that belong to Biomedical Research	documents that DO NOT belong to Biomedical Research
documents assigned to Biomedical Research	19	0
documents NOT assigned to Biomedical Research	1	40

Table IV: contingency table for Computer Speech & Language

	documents that belong to Computer Speech & Language	documents that DO NOT belong to Computer Speech & Language
documents assigned to Computer Speech & Language	20	4
documents NOT assigned to Computer Speech & Language	0	36

Conventional performance measures are defined and computed from this contingency tables. This measures are recall, *r*, precision, *p*, fallout, *f*, accuracy, *Acc*, and error, *Err*:

- $r = a / (a + c)$, if $a + c > 0$, otherwise undefined;
- $p = a / (a + b)$, if $a + b > 0$, otherwise undefined;
- $f = b / (b + d)$, if $b + d > 0$, otherwise undefined;
- $Acc = (a + d) / N$;
- $Err = (b + c) / N$.

Table V: Performance measures

	recall	precision	fallout	accuracy	error
Digital Signal Processing	85%	100%	0%	95%	5%
Biomedical Research	95%	100%	0%	98%	2%
Computer Speech & Language	100%	83%	10%	93%	7%

Macro and Micro-averaging

For evaluating performance of the system across categories, there are two conventional methods, namely macro-averaging and micro-averaging. Macro-averaged performance scores are calculated by first computing the scores for the contingency tables for each category and then averaging those to obtain global means. This is a per-category average, which gives equal weight to every category regardless of its frequency.

Table VI: Macro-Averaged Performance Measures

	recall	precision	fallout	accuracy	error
Macro-Averaging	93,3%	94,3%	3,3%	95%	4,6%

Micro-averaged performance scores give equal weight to every document. this can be more representative in our case, since the number of documents is small and all the classes were tested with the same number of abstracts. First we create a global contingency table, with the average of the values in the category tables, and use this to infer the performance scores.

Table VII:global contingency table for a class

	documents that belong to C	documents that DO NOT belong to C
documents assigned to C	17+19+20 = 56	0+0+4 = 4
documents NOT assigned to C	3+1+0 = 4	40+40+36 = 116

Table VIII:Micro-Averaged Performance Measures

	recall	precision	fallout	accuracy	error
Micro-Averaging	93%	93%	3%	96%	4%

c) Unbalanced Corpora.

Some poor results – bellow 30% successfully classified documents– were obtained in the first versions of the program. Two circumstances were pointed out as possible reasons: The corpus was both small and unbalanced.

The abstracts in one of the journals tended to be significantly shorter than the average so both the corpus and the test documents from this class contained less information. Therefore, very few abstracts were finally assigned to this cluster.

Enlarging the corpora proved to be a better solution than artificially balancing the class corpus. After having collected a corpus of near 20000 words, the classification rate reached 93% correctly assigned documents, still with a rather unbalanced distribution (Coefficient of Variance = 20%). Neither larger corpora nor better distribution resulted in reduced misclassification (see figure 1).

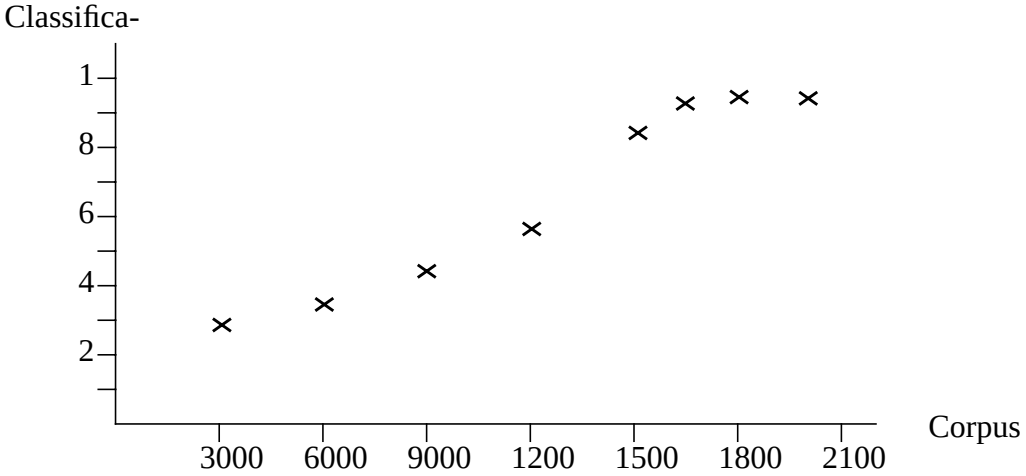

Figure 1. Classification rates with respect to the size of the corpus. Coefficient of Variance = 0,2

Above this threshold, enlarging the corpora had little effect on the number of new significant terms for each class and consequently on the performance (see figure 2). This suggests that the presence/absence of a common term is clearly more revealing for a good classification than the number of occurrences. Because of this property, words such as *model* or *system* that appear often in all classes will have restricted effect in the classification decision compared to those terms which are unique for a class.

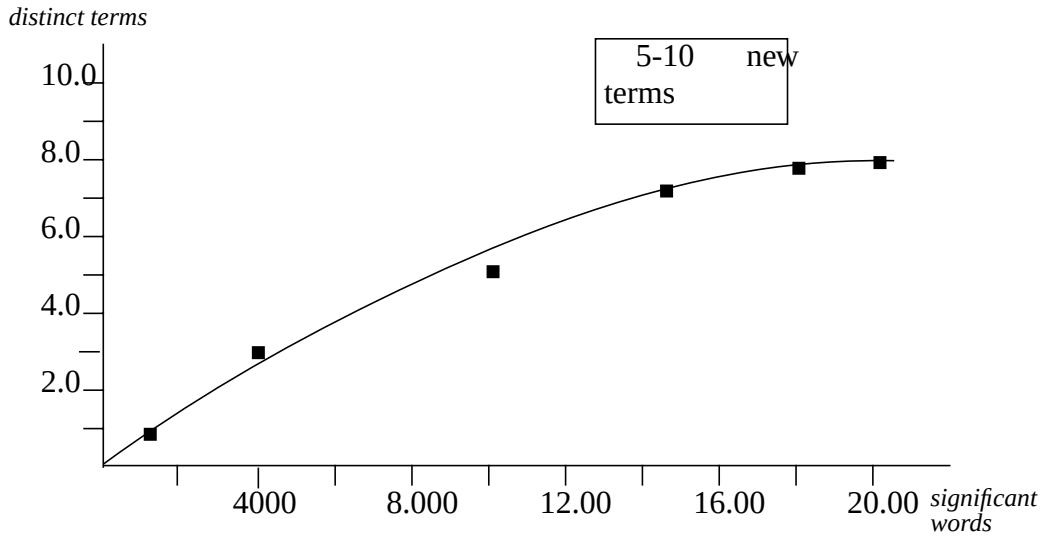

Figure 2. Relation between the size of the corpus and the number of different terms on it.

References

[1] W. Cohen. *Learning to classify English text with ILP methods*. In luc De Raedt, editor. Advances in Inductive Logic Programming, p. 124-143 IOS Press 1995.

[2] Y Yang, S. Slattery and R. Ghani. *A study of approaches to hypertext categorization.* Journal of Intelligent Information Systems, Volume 18, Number 2, March 2002.

[3] Kristina Toutanova, Francine Chen, Kris Popat & Thomas Hofmann. *Text Classification in a Hierarchical Mixture Model for Small Training Sets*, Xerox PARC and Brown University, Proceedings 10th International Conference on Information and Knowledge Management, 2001.

[4] Paul Ruhlen, Husrev Tolga Ilhan, and Vladimir Livshits *Unsupervised Web Page Clustering* CS224N/Ling237 Final Projects Spring 2000, Stanford Univ.

[5] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. *Syntactic clustering of the Web.* In Proc. 6th WWW Conf., Apr. 1997.

[6] Sep Kamvar & Carla Pinon, *Probabilistic Smoothing Models for Automatic Text Classification* CS224N/Ling237 Final Projects Spring 2001, Stanford University.

[7] Zoe Abrams, Mark Chavira, and Dik Kin Wong, *Gender Classification of Literary Works* CS224N/Ling237 Final Projects 2000, Stanford University.

[8] Jeff Chang, *Using the MeSH Hierarchy to Index Bioinformatics Articles* CS224N/ Ling237 Final Projects 2000, Stanford University.

[9] Yiming Yang *An evaluation of statistical approaches to text categorization.* Journal of Information Retrieval, Vol 1, No. 1/2, pp 67--88, 1999.

[10]Tom M. Mitchell. Machine Learning. McGraw Hill, 1997.

[11]http://www.ncbi.nlm.nih.gov/entrez/query/static/help/pmhelp.html#stopwords