Automatic Classification of News Articles in Spanish

U. Cerviño Beresi [*], J. J. García Adeva [†], R. A. Calvo [‡]
H. A. Ceccatto [§]

August 2, 2004

Instituto de Física Rosario
27 de Febrero 210 bis Rosario, Argentina

Abstract

We apply machine learning techniques to the automatic classification of news articles from the local newspaper *La Capital* of Rosario, Argentina. The corpus (LCC) is an archive of approximately 75,000 manually categorized articles in Spanish published in 1991. We benchmark on LCC three widely used supervised learning methods: k-Nearest Neighbors, Naïve Bayes and Artificial Neural Networks, illustrating the corpus properties.
Keywords: Spanish Text Classification, Neural Networks, Multinomial Naïve Bayes, k-Nearest Neighbours

1 Introduction

The industry around writing, distributing and selling news stories to consumers is one of the most affected by the internet revolution, and therefore going through a lot of changes. News agencies such as Reuters have thousands of reporters around the globe and sell content to most newspapers. This content normally needs to be organized by topics, searched and managed, which are all very labor-intensive

[*]email address <cervino@ifir.edu.ar>
[†]email address <jjga@ee.usyd.edu.au>
[‡]email address <rafa@ee.usyd.edu.au>
[§]email address <ceccatto@ifir.edu.ar>

activities. If done manually, they are slow and expensive and therefore not appropriate for modern –often internet-based– businesses. Automatic document classification aims at reducing this cost and speeding the process of organizing the information.

The problem of automatically assigning predefined categories to text documents uses a growing number of statistical classification methods and machine learning techniques developed in recent years. Most of these methods assume that a document can be represented as a vector, dismissing the order of words and other grammatical issues, and that this representation is able to retain enough useful information for the classification task. These ideas have been particularly successful in information retrieval [5, 4]. Figure 1 gives an example of how vector models can be constructed.

Most studies on automatic text classification have developed and tested algorithms using English corpora. Very little research has been done in developing specific tools and assessing the accuracy of these techniques on Spanish documents. As a contribution in this direction, in this work: i) we generate a corpus of approximately 75,000 manually categorized articles in Spanish published in 1991 in the local newspaper *La Capital* of Rosario, Argentina, and ii) we benchmark three widely used supervised learning methods: k-Nearest Neighbors (kNN), Multinomial Naïve Bayes (MNB) and Artificial Neural Networks (ANN) on this corpus, illustrating its properties. As a byproduct, we build required tools in text classification specific for the Spanish language.

This paper is organized as follows: Section 2 briefly describes the document classification techniques, focusing on the implications for processing documents in Spanish. Section 3 discusses specific issues of the LCC collection and the performance measures to be used in the evaluation. Section 4 presents the results obtained and section 5 collects some conclusions.

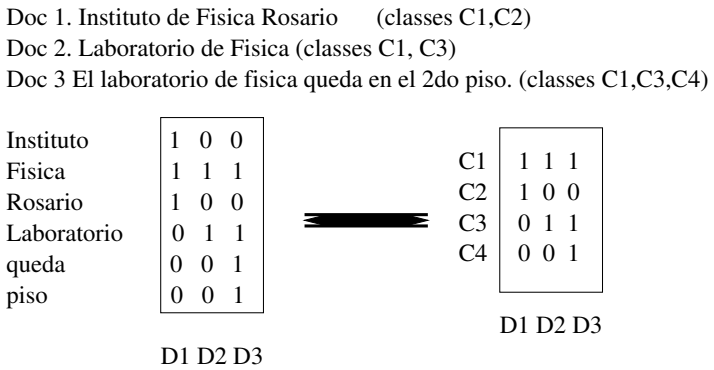

Figure 1: The vector model representation of documents.

2 A brief introduction to automatic document classification

Document classification can be thought of as a problem of mapping the space of input documents to the space of output classes. Many machine learning and statistical classification techniques are being applied to this problem [6]. The choice of a particular technique affects many performance measures: computational time, memory requirements, classification accuracy, etc. Yang evaluated [9] different statistical approaches applied to text classification, including kNN, Linear Least Square Fit (LLSF), Support Vector Machines (SVM), NB and ANNs.

Central to the application of any of these techniques is the chosen representation of documents. Figure 1 shows three documents (Doc1 to Doc3) that belong to one or more classes (C1, C2,...). First, in order to reduce the number of distinct terms a list of stop-words is removed from the documents. In most languages, some words have low information content; in English and Spanish, articles and prepositions are some of them. In Spanish it is common to remove 100-300 words from the document, reducing its length by 30-40%. In the example shown some stop-words (de, el, en) were removed before processing. Another frequent preprocessing step is stemming, not shown in this example, where words in different tenses, singular/plural, etc., are reduced to one term (stem).

In a vector model, the vectors representing documents of a corpus are arranged as files in a matrix A whose entries a_{ij} correspond to the weighted value for term i of document j. This matrix can be constructed using different weighting protocols:

i) **binary weighting** (used in Figure 1)

$$a_{ij} = \begin{cases} 0 & : \text{ term } i \text{ does not occur in document } j \\ 1 & : \text{ term } i \text{ occurs in the document } j \end{cases}$$

ii) **Term Frequency (TF) weighting**

$$a_{ij} = TF_{ij} = \text{number of times term } i \text{ appears in document } j$$

iii) **Inverse Document Frequency (IDF) weighting**

$$a_{ij} = IDF_{ij} = \frac{\text{number of documents in the collection}}{\text{number of documents with term } i + 1}$$

Finally, products of the form $f(TF_{ij}) \times g(IDF_{ij})$ are also used [5]. In this work we will describe results using the common choice $f(TF) = TF$ and $g(IDF) = IDF$ as the TF and IDF factors respectively. Furthermore, weight vectors are usually normalized by the cosine normalization $\sqrt{\sum_i (a_{ij}^2)}$ (i.e., an inner product

	c present	c absent	
t present	A	B	A+B
t absent	C	D	C+D
	A+C	B+D	

Table 1: Term–category contingency table.

function of two cosine normalized vectors will yield the same results as the cosine function on vectors, either normalized or not).

The dimension of the document vector spaces is proportional to the number of terms remaining after stop-word removal and stemming. Even for moderately-sized text collections, ten or a hundred thousand terms may remain, a number prohibitively high for some classification algorithms. Then, dimensionality reduction techniques are needed. These techniques are also used to:

- filter noise in document representation,
- understand the structure of the data,
- improve classification, and
- improve the computational efficiency.

This dimensional reduction is frequently performed using the χ^2 statistic, which measures the dependence of class c on the occurrence of term t and can be computed as:

$$\chi^2(t,c) = \frac{N \times (AD - CB)^2}{(A+C) \times (B+D) \times (A+B) \times (C+D)}$$

Here A, B, C, D are defined in Table 1 and N is the number of documents in the corpus; dividing by N, we get Table 2. If t and c are independent, then $e_{tc} = p_t \times p_c$, $e_{t\hat{c}} = p_t \times p_{\hat{c}}$, $e_{\hat{t}c} = p_{\hat{t}} \times p_c$, $e_{\hat{t}\hat{c}} = p_{\hat{t}} \times p_{\hat{c}}$, so the above definition is equivalent to:

$$\chi^2 = \frac{(p_{tc} - e_{tc})^2}{e_{tc}} + \frac{(p_{\hat{t}c} - e_{\hat{t}c})^2}{e_{\hat{t}c}} + \frac{(p_{t\hat{c}} - e_{t\hat{c}})^2}{e_{t\hat{c}}} + \frac{(p_{\hat{t}\hat{c}} - e_{\hat{t}\hat{c}})^2}{e_{\hat{t}\hat{c}}}$$

Notice that χ^2 has a value of 0 if t and c are independent.

In order to reduce the dimensionality of the classification problem, for each class c the χ^2 statistic of term i, $\chi^2(t_i, c)$ is computed. The values are then combined in several ways, for instance:

$$\chi^2_{avg}(t) = \sum_{c=1}^{n} Pr(c)\chi^2(t_i, c),$$

	c present	c absent	
t present	p_{tc}	$p_{t\hat{c}}$	p_t
t absent	$p_{\hat{t}c}$	$p_{\hat{t}\hat{c}}$	$p_{\hat{t}}$
	p_c	$p_{\hat{c}}$	

Table 2: Normalized term–category contingency table.

where $Pr(c) = freq(c)/N$ is the probability of c;

$$\chi^2_{max}(t) = \max_{c=1}^{n}\{\chi^2(t_i, c)\}$$

and

$$\chi^2_{max \times Pr}(t) = \max_{c=1}^{n}\{Pr(c)\chi^2(t_i, c)\}.$$

These new values can be finally used to produce a ranking table, where highly informative terms are on top of the list and the last ones are removed from it. The number of terms to be used will determine the number of inputs to the classification method used, so the computational costs must be considered. The more terms we keep, the higher the probability of retaining non-informative terms that introduce noise to the learning process. On the other hand, if we keep very few terms, we risk loosing the informative ones. Depending on the needs of the application one of the above feature selection schemes should be used. They will filter terms that optimize the average performance on the documents or on the classes (see Section 4).

The underlying assumption in using the χ^2 statistic is that features whose appearance in a document is highly correlated to a class are useful for measuring class membership. The selection of the averaging procedure $\chi^2_{avg}(t)$, $\chi^2_{max}(t)$ or $\chi^2_{max \times Pr}(t)$ will affect the performance. If the average includes a $Pr(c)$ weighting, the $\max$ or avg will weight classes differently, giving equal weight to every document.

Yang et al. [10] compared different dimensionality reduction techniques (a.k.a. feature selection), finding that χ^2 was among the most effective ones. However, they did not specify what averaging of χ^2 they followed. In a previous work [1] we have used the χ^2 statistic with the $\chi^2_{avg}(t)$ averaging scheme.

After the process of dimensionality reduction, the documents are represented by vectors and different classification techniques can be applied. In this work we will benchmark on LCC three widely used supervised learning methods: ANN, MNB and kNN. ANNs have proven useful in many classification tasks. Several authors have recently provided results of ANNs applied to text classification. Ng et al. [3] and Yang et al. [9] have reported results on the Reuters-21450 dataset. Ng et al. used a different single-layer perceptron for classifying each class; Yang

et al. approach uses one network for all categoris and is similar to the one followed in this work. The Naïve Bayes classifier makes strong assumptions about how the data is generated and proposes a probabilistic model that embodies such assumptions; then, it uses a set of labeled examples to estimate the parameters of the generative model. This classifier has been successfully used in text classification. Two commonly used probabilistic models for text classification, under the Naïve Bayes framework, are the multi-variate Bernoulli and the multinomial model. These two approaches were compared in [2] and the multinomial model is shown to perform significantly better than the multi-variate Bernoulli. This motivated us to use this model. kNN is an instance-based classification algorithm that has been well studied in the literature from the document categorization perspective [8]. An unseen document d_j is categorized as belonging to category c based on the categories of the k most similar instances from the set of training documents. So, given that a certain number of these k documents belong to category c, it can be established that $d_j \in c$. The similarity between the feature vectors that represent the documents is measured by the Euclidean distance (cosine value) between the two vectors in question. The weight of each of the k-nearest neighbors can be set using different approaches. We have used three weighting schemes: the same weight for all of the k nearest neighbors, inverse with their distances to the test document (1/distance), and opposite to their distances (1-distance). The building of a kNN categorizer also includes experimentally determining the optimal value of k. The best results are usually obtained with $30 \leq k \leq 45$ [9]. It is also interesting to note that increasing the value of k does not degrade the performance significantly. In our experiments we found $k = 33$ to be the optimum value for LCC. One of the advantages of kNN is that, being a lazy learner where there is no computation involved for training, all the decisions are performed during the categorization process. Hence, the classifier only needs to store the feature vectors that correspond to the training documents; no additional data structures have to be created, as it happens with other classification algorithms (e.g. Naïve Bayes). This can be especially interesting in situations where memory requirements are limited. The disadvantage of kNN is the categorization time being linear to the amount of training documents. This is due to the complete training set requiring to be ranked against the instance to be categorized.

3 The problem, the data set and the performance measures

3.1 The *La Capital* document collection

It is hard to find standard benchmark corpora for Spanish text classification, where methods can be tested and their performances reliably compared. This motivated us to construct a corpus derived from the original *La Capital* news articles. The original database is composed of approximately 75.000 newspaper articles written in Spanish, which are classified into 19 different categories. These articles have an internal representation in XML; we have eliminated all tags except the article's body and class.

The classification of the original database is a very difficult problem since several categories are too general and introduce noise. More complexity to the problem comes from the extremely uneven distribution of the documents over the classes. For instance, the class *ovacion* contains more than 22.000 documents while the class *arte* contains only 20. In view of this, we decided to simplify the problem by restricting ourselves to work with the "best" categories (those whose documents are more easily classified). In this way, we created a corpus with the following main properties:

- Same number of documents per class.
- Length of documents between 50 and 500 words.
- Documents belong to the "best" 10 classes defined below.

The procedure followed to obtain the "best" classes is the following:

1. Train a fast method—in our case the multinomial version of the Naïve Bayes classifier.
2. Obtain an ordered list of the categories based on their F_1 measure (see next section for its definition).
3. Select the "best" (most predictable) class and repeat from step one.
4. Use a similar procedure to obtain those classes which are the "worst" (less predictable) classes.
5. Finally, rank the classes using *position in the best class table* + *position in the reversed worst class table* as ranking value.

In Table 3 we show the list of classes ordered according to this procedure (notice that *arte* has been suppressed since all documents in this class have less than 50 words). By construction, documents on topmost classes in this list are more easily classified; in particular, in the evaluation of different classification methods we have used the first 10 classes.

Rank	Class	Best	Worst	Ranking Value
1	ovacion	2	1	3
2	compuser	1	4	5
3	policiales	3	3	6
4	campo	4	2	6
5	salud	6	5	12
6	justicia	5	7	12
7	economia	8	8	16
8	educacion	7	10	17
9	cultura	11	6	17
10	turismo	9	9	18
11	mundo	10	12	22
12	politica	12	11	23
13	escenario	13	14	27
14	region	15	13	28
15	opinion	14	17	31
16	ciudad	16	16	32
17	sociedad	17	15	32
18	pais	18	18	36

Table 3: Ordered list of the "best" classes within the corpus.

3.2 Performance measures

Table 4 describes the possible outcomes of a binary classifier. The Assigned YES/NO rows refer to the classifier outputs and the Correct YES/NO columns correspond to the actual example class label. The perfect classifier would have a value of 1 for a and d and 0 for b and c.

Using Table 4 we define three performance measures common in the text classification literature:

$$\text{recall} = r = \frac{\text{documents found and correct}}{\text{total documents in the class}} = \frac{a}{a+c}$$

		Correct	
		YES	NO
Assigned	YES	a	b
	NO	c	d

Table 4: Contingency table.

$$\text{precision} = p = \frac{\text{documents found and correct}}{\text{total documents found}} = \frac{a}{a+b}$$

The trade-off between recall and precision is controlled by setting the classifier parameters. To describe the performance both values should be provided. Another common performance measure is the F-measure defined by Rijsbergen [7]:

$$F_\beta(r,p) = \frac{(\beta^2+1)pr}{\beta^2 p + r}.$$

The most commonly used F-measure in text classification is F_1:

$$F_1(r,p) = \frac{2pr}{p+r}.$$

When dealing with multiple classes there are two possible ways of averaging these measures, namely, *macro average* and *micro average*. In the macro averaging, one contingency table as Table 4 per class is used; performance measures are computed for each of them and then averaged. In micro averaging only one contingency table is used, with an average on all the classes for each entry, and the performance measures are obtained therein. The macro average weights equally all the classes, regardless of how many documents belong to it. The micro average weights equally all the documents, thus favoring the performance of common classes.

Different classifiers will perform differently in common and rare categories. Learning algorithms are trained more often on more populated classes, thus risking local overfitting.

4 Results

We summarize here the steps followed in the preparation of the experiments, and the results obtained.

1. **Preparing the data**: A list of only 62 stop-words was removed from the document collection. The stemming algorithm was very simple: a list of the form $word \rightarrow stem$ was used to replace each word for its stem.

Class	Multi NB
ovacion	0.96 ± 0.03
compuser	0.91 ± 0.03
policiales	0.94 ± 0.02
campo	0.82 ± 0.06
salud	0.89 ± 0.02
justicia	0.87 ± 0.05
economia	0.81 ± 0.04
educacion	0.87 ± 0.04
cultura	0.83 ± 0.04
turismo	0.86 ± 0.02

Table 5: Performance on each class with Multinomial Naïve Bayes.

2. **Vectorization and weighting**: The resulting documents were represented as vectors, using $TF \times IDF$ weighting.

3. **Dimensionality reduction**: Since the $\chi^2_{avg}(t)$ includes the probabilities $Pr(c)$, the top features will account for the common classes. This means that all documents are weighted in approximately the same way, which results in a higher micro average effect.

4. **Methods used**: Multinomial version of the Naïve Bayes algorithm, ANNs with 100, 500 and 1000 hidden units, and kNN with $k = 33$ and three different neighbor weighting: same weight for all of the k neighbors, inverse with the neighbors' distances to the test document (1/distance), and opposite to their distances to the test document (1-distance). In all cases, we used 80% of the corpus for learning and 20% to test the classifier performance. This process was repeated 5 times and the test errors averaged. Tables 5 to 8 show the results obtained and the corresponding standard deviations .

5 Conclusions

We have compiled a corpus of documents written in Spanish and have performed some experiments with it. Results show that the Multinomial Naïve Bayes classifier outperforms the k-Nearest Neighbors and Artificial Neural Network classifiers on this corpus.

Although in this preliminary study our experiments were not intensive, they do suggest a future line of work. We are also planning to test other methods such

Class	100 h-units	500 h-units	1000 h-units
ovacion	0.79 ± 0.01	0.77 ± 0.01	0.80 ± 0.06
compuser	0.56 ± 0.02	0.56 ± 0.01	0.59 ± 0.06
policiales	0.75 ± 0.03	0.77 ± 0.03	0.75 ± 0.01
campo	0.69 ± 0.02	0.64 ± 0.02	0.68 ± 0.04
salud	0.65 ± 0.04	0.63 ± 0.12	0.67 ± 0.09
justicia	0.77 ± 0.03	0.83 ± 0.02	0.79 ± 0.06
economia	0.67 ± 0.03	0.67 ± 0.06	0.67 ± 0.04
educacion	0.80 ± 0.02	0.81 ± 0.01	0.83 ± 0.02
cultura	0.67 ± 0.01	0.65 ± 0.03	0.68 ± 0.001
turismo	0.68 ± 0.01	0.67 ± 0.02	0.68 ± 0.06

Table 6: Performance on each class with ANNs.

Class	weight = 1.0	weight = 1 / distance	weight = 1 - distance
ovacion	0.77 ± 0.07	0.73 ± 0.07	0.77 ± 0.08
compuser	0.82 ± 0.11	0.75 ± 0.09	0.77 ± 0.05
policiales	0.81 ± 0.13	0.75 ± 0.02	0.78 ± 0.05
campo	0.69 ± 0.08	0.83 ± 0.03	0.82 ± 0.08
salud	0.71 ± 0.10	0.74 ± 0.11	0.79 ± 0.05
justicia	0.80 ± 0.06	0.81 ± 0.11	0.77 ± 0.06
economia	0.72 ± 0.11	0.75 ± 0.12	0.87 ± 0.09
educacion	0.81 ± 0.07	0.84 ± 0.11	0.80 ± 0.10
cultura	0.82 ± 0.07	0.85 ± 0.13	0.82 ± 0.06
turismo	0.80 ± 0.07	0.78 ± 0.10	0.84 ± 0.05

Table 7: Performance on each class with kNN ($k = 33$).

Classifier	MicroF_1	MacroF_1
Multinomial Naïve Bayes	0.87 ± 0.02	0.88 ± 0.02
ANN - 100 hidden units	0.69 ± 0.01	0.70 ± 0.01
ANN - 500 hidden units	0.69 ± 0.01	0.70 ± 0.01
ANN - 1000 hidden units	0.70 ± 0.002	0.71 ± 0.004
kNN - weight = 1.0	0.77 ± 0.03	0.78 ± 0.03
kNN - weight = 1 / distance	0.78 ± 0.02	0.78 ± 0.02
kNN - weight = 1 - distance	0.80 ± 0.01	0.80 ± 0.01

Table 8: Performances of different classifiers.

as Rocchio, SVM and ensembles of classifiers on this corpus, in order to conduct a more extensive investigation of its properties.

Since this is a new corpus of documents, benchmarks have not been previously established for it. This makes impossible for us to compare our results with those of other authors, however we expect to set grounds for this to happen in the future.

References

[1] Rafael A. Calvo and H. A. Ceccatto, *Intelligent document classification*, Intelligent Data Analysis **4** (2000), no. 5.

[2] Andrew McCallum and Kamal Nigam, *A comparison of event models for naive bayes text classification*, Proceedings of AAAI-98 Workshop on Learning for Text Categorization (Madison, Wisconsin), 1998, pp. 137–142.

[3] Hwee T. Ng, Wei B. Goh, and Kok L. Low, *Feature selection, perceptron learning, and a usability case study for text categorization*, Proceedings of SIGIR-97, 20th ACM International Conference on Research and Development in Information Retrieval (Philadelphia, US) (Nicholas J. Belkin, A. Desai Narasimhalu, and Peter Willett, eds.), ACM Press, New York, US, 1997, pp. 67–73.

[4] Gerald Salton, *Developments in automatic text retrieval*, Science (1991), no. 253, 974–979.

[5] Gerard Salton, *Automatic text processing: The transformation, analysis, and retrieval of information by computer*, Addison-Wesley, Reading, Pennsylvania, 1989.

[6] Fabrizio Sebastiani, *Machine learning in automated text categorization*, ACM Computing Surveys (CSUR) **34** (2002), no. 1, 1–47.

[7] C. J. Van Rijsbergen, *Information retrieval, 2nd edition*, 2 ed., Butterworths, 1979.

[8] Yiming Yang and Christopher G. Chute, *An example-based mapping method for text categorization and retrieval*, ACM Trans. Inf. Syst. **12** (1994), no. 3, 252–277.

[9] Yiming Yang and X. Liu, *A re-examination of text categorization methods*, 22nd Annual International SIGIR (Berkley), August 1999, pp. 42–49.

[10] Yiming Yang and Jan O. Pedersen, *A comparative study on feature selection in text categorization*, Proceedings of ICML-97, 14th International Conference on Machine Learning (Nashville, US) (Douglas H. Fisher, ed.), Morgan Kaufmann Publishers, San Francisco, US, 1997, pp. 412–420.