Intelligent document classification

Rafael A. Calvo, H. A. Ceccatto
Instituto de Física Rosario (CONICET-UNR)
27 de Febrero 210bis
2000 Rosario, Argentina
`rafa@ifir.edu.ar`

May 26, 2000

Abstract

In this work we investigate some technical questions related to the application of neural networks in document classification. First, we discuss the effects of different averaging protocols for the χ^2 statistic used to remove non-informative terms. This is an especially relevant issue for the neural network technique, which requires an aggressive dimensionality reduction to be feasible. Second, we estimate the importance of performance fluctuations due to inherent randomness in the training process of a neural network, a point not properly addressed in previous works. Finally, we compare the neural network results with those obtained using the best methods for this application. For this we optimize the network architecture by evaluating much larger nets than previously considered in similar studies in the literature.

KEYWORDS: text classification, statistical learning, neural networks, knowledge management.

1 Introduction

The content explosion in the information age has produced a major challenge to science and technology: How can we use efficiently all the numerical data, text, sound and video produced? A clear sign of the problems to come is that the contents available on the Internet are growing at a much faster pace than the possibility of finding particular pieces of this information. The number of documents available on the web on almost any subject, and the ambiguity in user queries sent to search engines like Altavista (about 2 terms per query), created a great demand for classification trees like Yahoo and DMOZ (currently used by Netscape, Lycos and others). In response, the Open Directory Project (DMOZ, http://www.dmoz.org) has more than 20,000 volunteers classifying documents. Although this huge community effort has produced a classification tree with about 1.4 million documents, assuming the web had 800 million pages as of February 1999 [5] and this figure doubled in the last 12 months, the project has only classified less than 0.1% of the web. Automatic classification systems are a must for the future of knowledge management.

Text categorization (TC) is the problem of automatically assigning predefined categories to text documents. Most techniques used to tackle this problem are based on the assumption 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[8, 9]. Thus, document classification can be thought of as a problem of mapping the vector space corresponding to the input documents to the space of output classes, which allows the use of standard statistical classification methods[1, 2] and machine learning techniques to solve it.

The dimension of the vector space used in TC is, in principle, equal to the number of different terms remaining after words with low information content are removed from the document corpus and words in different tenses, singular/plural, etc. are reduced to one term (stem). Even for moderate-size text collections one can end up with tens or even hundreds of thousand terms, a number prohibitively high for some classification algorithms. Then, dimensionality reduction (also called feature selection) techniques are needed. These techniques are also helpful for

- reducing noise in document representation,
- understanding the structure of the data, and
- improving classification and computational efficiency.

Yang and Pedersen[15] compared different dimensionality reduction techniques that remove less informative terms according to their statistics, and they found that Information Gain and χ^2 were the most effective. Schutze *et al.*[10] used mutual information and χ^2 statistic to select features in a neural network approach to TC.

Besides this dimensionality problem, other questions like computational time complexity and memory requirements have to be considered when choosing one particular dimensionality-reduction technique for document classification. Moreover, several performance measures can be affected, including micro and macro averages of recall and precision (see Section 3 for definitions of these quantities). Yang[13] evaluated fourteen different approaches to TC and found that the performance of a classifier depends strongly on the data used for evaluation. However, by evaluating classifiers on multiple collections she concluded that *k*-Nearest Neighbors (*k*NN), Linear Least Square Fit (LLSF), Widrow-Hoff (WH) and Neural Networks (NNet) are the top performers among the learning and non-learning methods evaluated. More recently, Yang and Liu[14] re-examined the problem focusing on the robustness of different methods in dealing with a skewed category distribution. In this study Support Vector Machines (SVM), *k*NN and LLSF outperformed NNet and Naive Bayes (NB) classifiers when the number of positive training instances per category is very small, but all these methods performed comparably when the categories are sufficiently common.

Neural networks can learn nonlinear mappings from a set of training patterns by adjusting their parameters according to the back-propagation rule for error minimization. This learning process needs to be monitored using a cross-validation set to avoid overfitting the data. Hertz *et al.*[4] and Ripley [7] are good introductions to neural networks and their use in classification tasks. Figure 1 gives a pictorial representation of a neural network for TC: the input layer has as many units (neurons) as features retained, the number of hidden units is determined by optimizing the performance on the cross-validation set, and there is one output unit for each possible class.

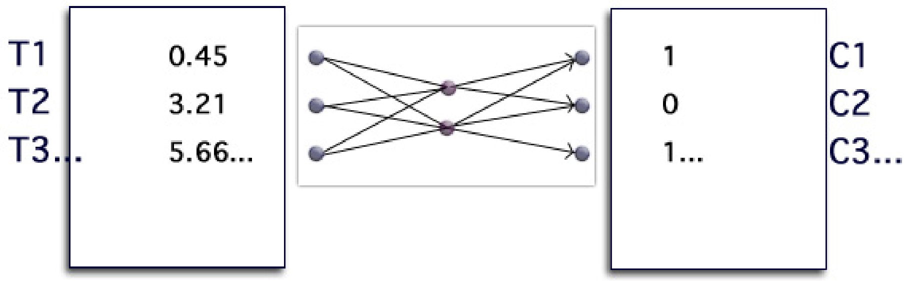

Figure 1: Neural Network for text classification.

Several authors have recently provided results of neural networks applied to TC. Wiener *et al*. [12], Ng *et al*.[6] and Yang and Liu[14] have reported on the Reuters-21578 dataset. Ng *et al*. considered only a one layer-perceptron and Wiener *et al*. tried also a three-layer network; both systems used one network for classifying each class. In Yang and Liu's approach a single network is used for all the 90 categories in the Reuters-21578 dataset, but these authors did not report the use of cross-validation or any complexity optimization technique in their study. Moreover, in their comparison of different methods they did not consider that performing a single training experiment is not enough because the precision and recall obtained are influenced by uncertainties in the learning process.

In this work we aim to fill some gaps in the literature, particularly in connection with the application of neural networks to TC. First, we want to discuss what averaging of the χ^2 statistic is better for removing non-informative terms. This question has been overlooked in the comparison of different dimensionality reduction techniques performed by Yang and Pedersen[15] and, as we will see in Section 4, it can be important for some applications. This is an especially relevant issue for the neural network technique, which requires an aggressive dimensionality reduction to be feasible. Second, since neural networks are a largely unstable statistical method, as stated above a single experiment does not characterize completely their performance on a given problem. The model nonidentifiability (multiple minima of the error surface that are, *a priori*, equally good) comes from inherent randomness in the training process and data structure, so that the model variance has to be estimated to correctly appraise its performance. This point has not been properly addressed in previous works, including [14, 13] that evaluates different statistical approaches.

The contents are organized as follows. In Section 2 we discuss how to represent a document by a vector and the different weighting protocols that enhance/diminish the importance of particular terms (frequent or rare words in a class). We also give the definition of the χ^2 statistic that will be used in our experiments to select the most informative terms, including the three different averages procedures for this quantity. In Section 3 we comment on the benchmark database used –the ApteMod version of Reuters-21578– and define the measures used to compare the performances of different methods. In Section 4 we summarize, for the sake of clarity, the steps followed in our experiments and present the results obtained. Finally, in the last section (Section 5) we draw some conclusions.

2 Vector Models, Weighting and Feature Selection

As stated in the introduction, most statistical techniques used in TC are based on the assumption that a document can be represented as a vector[8, 9]. Figure 2 shows how a vector model can be constructed in a simple toy example.

Doc 1. Instituto de Fisica Rosario (classes C1,C2)
Doc 2. Laboratorio de Fisica (classes C1, C3)
Doc 3 El laboratorio de fisica queda en el 2do piso. (classes C1,C3,C4)

$$\begin{array}{l|ccc|}
\text{Instituto} & 1 & 0 & 0 \\
\text{Fisica} & 1 & 1 & 1 \\
\text{Rosario} & 1 & 0 & 0 \\
\text{Laboratorio} & 0 & 1 & 1 \\
\text{queda} & 0 & 0 & 1 \\
\text{piso} & 0 & 0 & 1 \\
 & \text{D1} & \text{D2} & \text{D3}
\end{array}
\longleftrightarrow
\begin{array}{l|ccc|}
\text{C1} & 1 & 1 & 1 \\
\text{C2} & 1 & 0 & 0 \\
\text{C3} & 0 & 1 & 1 \\
\text{C4} & 0 & 0 & 1 \\
 & \text{D1} & \text{D2} & \text{D3}
\end{array}$$

Figure 2: The vector model representation of documents.

Three documents (Doc1 to Doc3) belong to one or more of four classes (C1 to

	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.

C4). First, in order to reduce the number of distinct terms a list of stop-words, *i.e.* words that have low information content, is removed from the documents. For instance, removing articles and prepositions in English and Spanish commonly reduces 30-40% the document length. In the example some stop-words (de, el, en) are eliminated. Another frequent preprocessing is stemming, not show in this example, where words in different tenses, singular/plural, etc. are reduced to one term (stem). Then, the vectors representing documents of the corpus are arranged as files of a matrix D, whose entries $D_{\alpha\, j}$ correspond to the weighted value for term j of document α. This matrix can be constructed under different weighting protocols:

- **Binary weighting** (used in Figure 2)

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

- **Term Frequency (TF) weighting**

$$D_{\alpha\, j} = TF_{\alpha\, j} = \text{number of times term } j \text{ appears in document } \alpha$$

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

$$D_{\alpha\, j} = IDF_{\alpha\, j} = \ln\left(\frac{\text{number of docs in the collection}}{\text{number of documents with term } j}\right) + 1$$

In addition, products of the form $f(TF_{\alpha\, j}) \cdot g(IDF_{\alpha\, j})$ with f and g arbitrary functions are also used[8]. The SMART package developed by Salton and Buckley is one of the oldest academic packages with several weighting schemes and we will use their 3-letter coding to identify each. The first letter can be `b` for binary weighting, `l` for $f(TF) = 1 + \ln(TF)$ or `n` for no TF weighting; the second letter can be `t` for the IDF weighting defined above or `n` for no weighting; finally, the third letter can be `c` for normalization or `n` for no normalization. In this work we will take $f(TF) = 1 + \ln(TF)$ and $g(IDF) = IDF$ (`ltc` in SMART notation[8]). Weight vectors are normalized by the standard cosine normalization $\sqrt{\sum_j D^2_{\alpha\, j}}$.

The dimension of the document-corpus vector space is equal to the number of different terms remaining after stop-word removal and stemming. Since this number can be prohibitively high for some classification algorithms, dimensionality reduction (feature selection) techniques are frequently needed. According to Yang and Pedersen[15], who found that the use of the χ^2 statistic was one of the most effective, in this work we will consider this method with the three different averaging schemes described below.

The χ^2 statistic measures the dependence of class c on the occurrence of term t, and is given by

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

where A, B, C, D and N are defined in table 1 (notice that χ^2 has a value of 0 if t and c are independent). For each class c the $\chi^2(t_j, c)$ statistic of term j is computed, and these values are combined in several scores, for instance:

$$\chi^2_{\max}(t_j) = \max_{c=1}^{K}\{\chi^2(t_j, c)\},$$

$$\chi_{\mathrm{avg\,Pr}}(t_j) = \sum_{c=1}^{K} \Pr(c)\chi^2(t_j, c)$$

and

$$\chi_{\max \Pr}(t_j) = \max_{c=1}^{K}\{\Pr(c)\chi^2(t_j, c)\}. \tag{1}$$

Here $\Pr(c) = \mathrm{freq}(c)/N$ is the probability of class c and K is the number of classes. These scores can be used to produce a term ranking table, where highly informative terms (more frequent in a class or subset of classes) are on top of the list and the last ones are removed from it. 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 $\chi^2_{\max}$, $\chi^2_{\mathrm{avg\,Pr}}$ or $\chi^2_{\max \Pr}$ will affect the performance. In particular, if we include a $\Pr(c)$ weighting, the $\max$ or avg will weight classes differently, giving equal weight to every document. Depending on the needs of the application one of these feature selection schemes should be used. They will select terms that optimize the average performance on the documents or on the classes (see Section 4). Notice however that the number of terms used will determine the number of inputs to the neural network, so the computational cost 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.

		Correct	
		YES	NO
Assigned	YES	a_j	b_j
	NO	c_j	d_j

Table 2: Contingency table for class j.

3 Data Set and Performance Measures

3.1 The Reuters document collection

It is hard to find standard benchmark sets for TC, where different methods can be tested and their performances compared reliably. Although several versions are available, the Reuters sets are a notable exception that many researchers use for benchmarking. These sets are based on the Reuters newswire and the first version was originally produced by the Carnegie Group Inc. (CGI) and used to evaluate their CONSTRUE system [3]. The other versions made available since then are mostly refinements of the so called Reuters-22173 and Reuters-21450 sets. The documents generally refer to financial news related to different industries and have a title and a content section (we have considered both indistinctly). In this work we will use the ApteMod version of Reuters-21578 that has 10788 documents, of which 7769 are normally used for training and 3019 kept for testing[14]. These documents are distributed in a total of 90 categories occurring in both subsets, with an average of 1.3 category assignments per document. After stop-words removing and stemming using standard algorithms, 24240 unique terms remained in the document collection. Starting from all the distinct terms in the training set, we tested different thresholds for the χ^2 statistic so only the top 500, 1000 and 2000 terms were retained for their use in the numerical experiments performed.

3.2 Performance measures

Table 2 describes the possible outcomes of a binary classifier. The Assigned YES/NO results refer to the classifier output when asked if a given document belongs to class c, and the Correct YES/NO refer to what the correct output is. The perfect classifier would have a value of 1 for a_j and d_j and 0 for b_j and c_j.

Using table 2 we define two performance measures common in the TC literature:

- Recall

$$r \equiv \frac{\text{classes found and correct}}{\text{total classes correct}} = \begin{cases} \frac{a}{a+c} & \text{if } a+c>0 \\ 1 & \text{otherwise} \end{cases}$$

- Precision

$$p \equiv \frac{\text{classes found and correct}}{\text{total classes found}} = \begin{cases} \frac{a}{a+b} & \text{if } a+b>0 \\ 1 & \text{otherwise} \end{cases}$$

	$\chi^2_{max}(t)$	$\chi^2_{max \times Pr}(t)$	$\chi^2_{avg}(t)$
MaF_1	0.269 $\pm$ 0.039	0.244 $\pm$ 0.027	0.265 $\pm$ 0.035
MiF_1	0.805 $\pm$ 0.015	0.833 $\pm$ 0.007	0.835 $\pm$ 0.009

Table 3: χ^2 performance and standard deviation, on test set, for 20 runs of nets with 500 input and 50 hidden units.

The trade-off between recall and precision is controlled by setting the classifier parameters and both values should be provided to properly describe the performance. Another common performance measure is the F-measure defined by Rijsbergen[11]:

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

The most commonly used F-measure in TC corresponds to $\beta = 1$, $F_1(r,p) = 2pr/(p+r)$,which weights precision and recall equally.

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 2 per class is used, the performance measures are computed for each of them and then averaged. In micro averaging only one contingency table for all the documents independently of the classes is used and the performance measures are obtained from it. The macro average weights equally all the classes, regardless of how many documents belong to it, while the micro average weights equally all the documents, thus favoring the performance on common classes. Notice that in general classifiers will perform differently in common and rare categories, and that learning algorithms are trained more often on more populated classes thus risking local overfitting.

4 Results

For the sake of clarity we first summarize the steps followed in the preparation and running of the experiments. Then, we present the results obtained.

1. **Preparing the data**: We used the ApteMod version of Reuters-21578 containing 7769 documents in the training set and 3019 documents in the test set. There is a total of 90 categories that occur in both subsets, with an average of 1.3 categories per document. A list of 571 stop-words were removed from the document collection and, after stemming, 24240 unique terms remained.

2. **Vectorization and weighting**: The resulting documents were represented as vectors using different weighting protocols as described in Section 2 (the Smart text processing package [8] was used for this stage); in all the experiments we used a cosine normalization. After some preliminary investigations, we found the `ltc` weighting to be the most convenient, so that in the following all the results discussed will correspond to this scheme.

3. **Dimensionality reduction**: The performances for the three χ^2 statistics defined in Section 2 were computed in all cases. Since $\chi^2_{\rm avg}$ and $\chi^2_{\rm max \cdot Pr}$ include the probability $\Pr(c)$, for these averages the top features account for the common classes, which means that all documents were effectively weighted the same producing a higher micro average. Alternatively, since $\chi^2_{\rm max}$ does not have this term, its maximization is independent of the class probability so this feature selection should produce better macro averages.

4. **Network architecture**: The selected terms were used as input features to the network. We tried networks with 500, 1000 and 2000 input neurons and 50, 100 and 150 hidden units; in all cases we considered 90 output neurons (one for each class).

5. **Training**: We randomly generated several cross-validation sets and in each case the corresponding documents were set aside and the network trained on the remaining ones. Since, as a rule of thumb, it is common to use 5-10% of the training set for validation, we tried using 500 and 1000 documents and the smaller cross-validation set resulted in better performance. The learning rate η was systematically reduced every time the algorithm found an error minimum on the cross-validation set, a technique similar to "cooling down" in simulated annealing.

Table 3 gives the macro (Ma) and micro (Mi) averages of the F_1 performance measure along with their standard deviations for the three χ^2 averaging variants. In this case, to estimate the performance variance we trained 20 networks, starting with different random weights and cross-validation sets. The three averaging schemes produced different micro averages with statistical significance $\alpha = 0.05$; for macro averaging however the differences are less significative. As expected, the use of $\chi^2_{\rm max\,Pr}$ or $\chi^2_{\rm avg\,Pr}$ produced the best micro averages of performance measures. Similarly, slightly better results for macro averaging can be obtained with a selection of terms based on the $\chi^2_{\rm max}$ statistics, although the performance is always very poor in this case. These effects, which could have been anticipated qualitatively, had no been estimated quantitatively on a reliable benchmark set. Moreover, they are seldom considered in the literature but should be carefully taken into account when deploying concrete applications.

The results in tables 4 and 5 were obtained using $\chi^2_{\rm max\,Pr}$. In table 4 we show the mean and standard deviation of ${\rm Mi}F_1$ and ${\rm Ma}F_1$ for networks with 50, 100 and 150 hidden units and with 500, 1000 and 2000 input features. An analysis of this table shows that the performances do not change sensibly due to the randomness in the training process and use of different validation sets (notice that the ${\rm Ma}F_1$ standard deviations are considerably larger than those of the ${\rm Mi}F_1$, although these estimations are here less reliable since we considered only 5 different training experiments). The fact that in all cases the neural network performance measured by the ${\rm Mi}F_1$ index is much better than the ${\rm Ma}F_1$ results is due to the very skewed category distribution, a point already stressed in [14]. On classes with only few examples the neural network will perform poorly, which is reflected in the small ${\rm Ma}F_1$ value. To clarify this point, in Fig. 3 we show how the classes are distributed according to the F_1 measure (so

	inputs	50 hidden	100 hidden	150 hidden
MaF_1	500	0.263 ± 0.024	0.288 ± 0.030	0.281 ± 0.028
MiF_1	500	.841 ± 0.006	0.845 ± 0.006	0.845 ± 0.005
MaF_1	1000	0.295 ± 0.025	0.321 ± 0.011	0.306 ± 0.045
MiF_1	1000	0.849 ± 0.004	0.853 ± 0.003	0.851 ± 0.009
MaF_1	2000	0.284 ± 0.019	0.285 ± 0.022	0.257 ± 0.032
MiF_1	2000	0.848 ± 0.004	0.845 ± 0.003	0.843 ± 0.006

Table 4: Performance of the network with 500 and 1000 input features. The average and standard deviation for 5 different run are shown.

	miR	miP	miF1	maF1
SVM	0.914	0.812	0.860	0.513
NNet	0.781	0.905	0.839	0.291
kNN	0.834	0.881	0.857	0.524
NB	0.769	0.825	0.796	0.389
NN 500 inp.	0.788 ± 0.011	0.911 ± 0.003	0.845 ± 0.005	0.282 ± 0.028
NN 1000 inp.	0.800 ± 0.016	0.908 ± 0.002	0.853 ± 0.003	0.326 ± 0.014
NN 2000 inp.	0.784 ± 0.007	0.915 ± 0.002	0.848 ± 0.004	0.284 ± 0.019

Table 5: Performance of different classifiers. The first 4 are taken from [14].

that the mean value of this histogram gives MaF_1) and also the number of documents contained in those classes. We see that nearly 62% of the classes that in total account for less than 10% of the documents have $F_1 \simeq 0$, while approximately 2% of the classes comprising almost 50% of the documents are almost perfectly classified ($F_1 \simeq 1$). This explains both the high MiF_1 and the low MaF_1. Notice also that there are no significant changes in performance with the number of units considered, although the results for 1000 input and 100 hidden neurons are consistently higher than the corresponding to other architectures. In [14] only networks with 500 input and 8, 16 and 32 hidden units were considered, which lead to slightly worse performances than those obtained in the present work. This is shown in table 5, where our neural network (NN) results and the results obtained in [14] using a variety of methods are displayed for comparison.

5 Conclusions

We have applied neural networks to a TC problem, the Reuter newswire corpus (Apte-Mod version), one of the most popular benchmark set in the literature. The dimension of the original vector model for this document corpus is very high, which makes the classification problem intractable until this dimension is reduced. The χ^2 statistic was successfully used for this task. We have compared three different averaging schemes for χ^2, and we have quantified how much they affect the MaF_1 and MiF_1 performance measures. In particular, we found that for MiF_1 the three schemes perform statistically

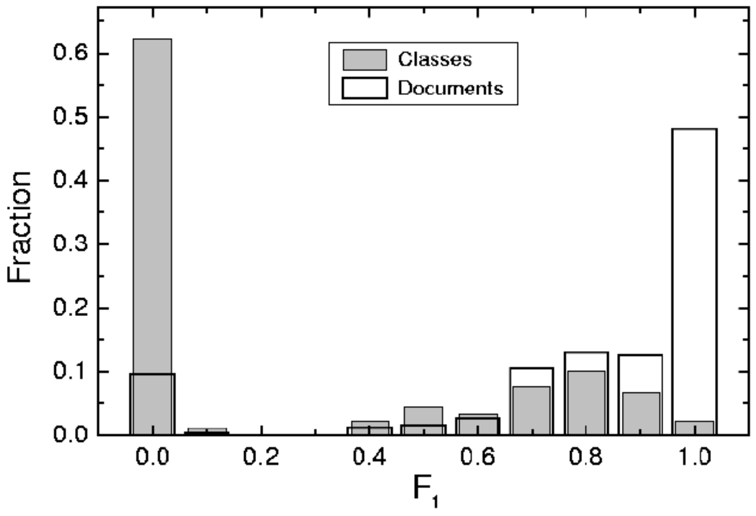

Figure 3: Fraction of classes and documents for different class performance levels.

different with a significance parameter $\alpha = 0.05$, while for $\mathrm{Ma}F_1$ the changes are much less important.

The neural network technique has some inherent randomness related to the learning process, so it is not possible to assess its performance by a single experiment. Several trainings are needed, preferably with different initial weights and cross-validation sets. We have used 5 to 20 independent experiments to determine average performances and their standard deviations, and we found that the effects of the model's nonidentifiability are in general negligible. We have also considered the neural network performance variance due to modifications in the number of input/hidden units; for the large architectures considered these modifications produced small but sensible changes in the final results. In particular, the best performances were obtained using a network with 1000 inputs and 100 hidden units, a much larger architecture than previously considered in the literature[14]. More input features slightly improves $\mathrm{Mi}p$, but the degradation on $\mathrm{Mi}r$ reduces the $\mathrm{Mi}F_1$. Interestingly, increasing or reducing the number of hidden units reduces both $\mathrm{Mi}p$ and $\mathrm{Mi}r$.

Finally, we compared the performance of the neural network classifier with the results of [14], where, in addition to neural networks, SVM, *k*NN, LLSF and NB classifiers were considered. The micro averaged precision ($\mathrm{Mi}p$) we obtained is the best for all the methods compared, which is particularly important for many applications where precision is the most important variable. Furthermore, the MiF_1 performance of neural networks is comparably as good as the ones of the top performers SVM, *k*NN and LLSF. On the other hand, the MaF_1 performance of neural networks is very poor, and this is due to a much lower recall level related to the very skewed category distribution of the document corpus. The trade-off between high recall or high precision is a well known problem and must be considered for each application.

Together with SVM, *k*NN and LLSF, neural networks produced the best overall models for this data set. These techniques should make possible automatic classification systems with a performance comparable to human classification. Future work includes trying these methods on new data sets and in the classification of documents in Spanish, in which we are particularly interested. In addition, other neural network schemes should also be tested.

6 Acknowledgments

RAC acknowledges partial support by the Language Technology Institute of Carnegie Mellon University, and by Proyecto FOMEC, Argentina. He also acknowledges Dr. Yiming Yang for helpful discussions. This work was partially supported by ANPCyT of Argentina (PICT/98 Nr. 11-03834).

References

[1] R. Duda and P. Hart. *Pattern classification and scene analysis*. Wiley, New York, NY, 1973.

[2] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, Boston,MA, 1990.

[3] P.J. Hayes and S. P. Weinstein. Construe/tis: a system for content-based indexing of a database of new stories. In *Second Annual Conference on Innovative Applications of Artificial Intelligence*, 1990.

[4] J. Hertz, A. Krogh, and R. Palmer. *Introduction to the theory of neural computation*. Addison-Wesley, Redwood, CA, 1991.

[5] S. Lawrence and G. Giles. Accessibility and distribution of information on the web. *Nature*, 400:107–109, 1999.

[6] H.T. Ng, W.B. Goh, and K.L. Low. Feature selection, perceptron learning, and usability case study for text categorization. In *ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'97)*, pages 67–73, 1997.

[7] B.D. Ripley. *Pattern recognition and neural networks*. Cambridge Univeristy Press, Cambridge, 1996.

[8] G. Salton. *Automatic Text Processing: The Transformation, Analysis and Retrieval of information by Computer*. Addison-Wesley, Reading, MA, 1989.

[9] G. Salton. Developments in automatic text retrieval. *Science*, 253:974–979, 1991.

[10] H. Schutze, D.A. Hull, and Pedersen J.O. A comparison of classifiers and document representations for the routing problem. In *Int. ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 22–34, 1995.

[11] C.J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 1979.

[12] E. Wiener, J.O. Pedersen, and A.S. Weigend. A neural network approach to topic spotting. In *Proceedings of the Fourth Annual Symposium on Document Analysis and Information Retrieval (SDAIR'95)*, 1995.

[13] Y. Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval Journal*, May, 1999.

[14] Y. Yang and X. Liu. A re-examination of text categorization methods. In *ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99)*, 1999.

[15] Y. Yang and J.P. Pedersen. Feature selection in statistical learning of text categorization. In *The Fourteenth International Conference on Machine Learning*, pages 412–420, 1997.