Integrating Lexical Knowledge in Learning-Based Text Categorization*

José María Gómez Hidalgo[1]
Manuel de Buenaga Rodríguez[1]
Luis Alfonso Ureña López[2]
María Teresa Martín Valdivia[2]
Manuel García Vega[2]

[1] Universidad Europea CEES
Departamento de Inteligencia Artificial
28670 Villaviciosa de Odón, Madrid, Spain
{jmgomez,buenaga}@dinar.esi.uem.es

[2] Universidad de Jaén
Departamento de Informática
23071 Jaén, Spain
{laurena,maite,mgarcia}@ujaen.es

Abstract

Automatic Text Categorization (ATC) is an important task in the field of Information Access. The prevailing approach to ATC is making use of a a collection of prelabeled texts for the induction of a document classifier through learning methods. With the increasing availability of lexical resources in electronic form (including Lexical Databases (LDBs), Machine Readable Dictionaries, etc.), there is an

*This work has been partially supported by Spanish Ministry of Science and Technology (FIT-070000-2000-249)

interesting opportunity for the integration of them in learning-based ATC. In this paper, we present an approach to the integration of lexical knowledge extracted from the LDB WordNet in learning-based ATC, based on Stacked Generalization (SG). The method we suggest is based on combining the lexical knowledge extracted from the LDB interpreted as a classifier with a learning-based classifier, through SG. We have performed experiments which results show that the ideas we describe are promising and deserve further investigation.

1 Introduction

Automatic Text Categorization (ATC) – the assignment of documents to a predefined set of categories – is a very interesting task for Information Access. Nowadays, it is possible to build automatic systems able to classify web pages in Internet directories like Yahoo! [14], to index records with respect to library subject headings [27], or to filter spam messages [6], among other applications. In ATC, there exists an interesting opportunity of taking advantage of several knowledge sources.

ATC has been the focus of much research in recent years [17, 21, 28]. The prevailing tendency in ATC is to apply Machine Learning and Information Retrieval techniques to induce an automatic system (a *classifier*) using a collection of pre-labelled texts (the *training collection*) [21].

With the increasing availability of electronically accessible information, integrating other knowledge sources in ATC is receiving more and more attention. For instance, some people are interested on taking advantage of the lexical knowledge present in Lexical Databases (LDBs) like WordNet [10, 20, 2]. Others try to integrate heuristic knowledge available in specific domains like in spam messages categorization [6, 18]. Also unlabelled documents have been used in ATC [15]. We have focused on designing methods for integrating lexical knowledge extracted from LDBs.

In this paper, we present a method for integrating lexical knowledge in ATC based on learning from a training collection. The model are based on adapting the meta-learning schemas *Stacked Generalization* [26] to the problem. The main idea is building two independent classifiers, and combining the output from them through a second-level classifier. First, a linear classifier is build using lexical items and relations present in WordNet. Secondly, a classifier can be induced from the training collection by running a learning algorithm. Finally, a third classifier is learnt from an independent training set of documents. This second-level classifier receives as input the output of the two previous classifiers, and outputs the final classification for the new documents.

We present the advantages of our method over others in the literature. Also, we have performed a series of experiments to test the ideas behind the method proposed. The results of our experiments show that our approach is promising.

The rest of this work is organized as follows. In the next section, we review the approaches previously followed for the integration of lexical knowledge in ATC. Afterwards, we present our method and examine the intuitions behind them. Later, our experiments and results are described and discussed. Finally, we state our conclusions.

2 Exploiting Lexical Knowledge Sources in ATC

Some resources, amenable of being used in Natural Language Processing tasks, have been built in the latest years. Lexical databases like WordNet [12], EDR [29], and EuroWordNet [24], are specially relevant for text classification. LDBs are repositories that accumulate information about the lexical items of one or several languages. For instance, WordNet 1.6 includes information about more than 125,000 words and nearly 100,000 concepts of the English language, organized according to semantic relations. This great amount of data has been successfully employed for improving text classification tasks like text retrieval [7] or word sense disambiguation [1]. The utilization of LDBs for TC is the focus of our work. The intuition that guide our work is that the more informed a system is, the better it will perform.

Several methods have been proposed for integrating lexical information to the training-based approach in ATC. As far as we know, and apart of our previous work [2, 5, 3, 23], only the work by Scott and Matwin [20] and by Junker and Abecker [10] has focused in this task. Scott and Matwin propose to include WordNet information at the feature level. Their work is based on expanding each word in the training collection with all the synonyms extracted from WordNet for it, including those available for each sense in order to avoid a word sense disambiguation process. They after train a rule-based classifier with the Ripper system. This approach has shown a decrease of effectiveness in the classifier obtained, mostly due to the word ambiguity problem. Our approach shares ideas with the work by Scott and Matwin. On one side, word synonyms are extracted from WordNet and added to the representation of texts, like in the Scott and Matwin approach. However, we extract synonyms just for category names, disambiguated to avoid the inclusion of noise.

Junker and Abecker have included the navigation of semantic relations in WordNet as operators in the rule induction process for training a rule-based classifier. This approach is discussed by Mitchell in [13] as a form of combining analytical and inductive knowledge for machine learning. Unfortunately, this work has not proven effective for TC either.

However, there still exists the potential of exploiting the lexical information in WordNet for ATC. We present a method for integrating knowledge additional to the training collection based on Stacked Generalization. The only requirement for the technique we propose is the ability to interpret additional knowledge as a classifier, independently of its form (a set of classification rules, a linear classifier, etc.).

3 Integrating Lexical Information through Stacked Generalization

Stacking generalization (*stacking* for short) is a method for combining different models for classification and prediction, based on the idea that different learning algorithms provide different but complementary explanations of the data [26]. The basic idea is learning a set of different classifiers with several learning algorithms (a decision tree learned, a rule based learner, a Naive Bayes classifier, etc.), and combining their predictions by training a confidence based, second level classifier, that learns which first level classifier to trust more depending on the fresh input. For the second level learner, linear classifiers perform best [22].

Our proposal consists of combining two first level classifiers: one learned from the training collection by one training algorithm at the election of the system developer, and one that uses lexical knowledge to classify new instances. The second classifier could be an expert system like that described in [8], a WordNet synonyms based classifier as in our work [2], or a classifier based on data specific to some domain (as spam filtering heuristics like the mail address of the sender for classifying spam email) [6, 18]. A second level linear classifier can then be learnt from a separate training set. The second level classifier decides which amount of prior knowledge is used. Also, any learning algorithm can be used as complementary to lexical knowledge.

4 A WordNet based classifier

For our experiments, a classifier based in WordNet has been built using the following procedure:

1. The documents are represented as term weight vectors in the Vector Space Model [19]. Terms are stemmed words that have passed through a stop-list filter[1], and which document frequency rank is in the 10% top scoring Document Frequency terms. We have used the Porter stemming algorithm. The weight for a term in a document vector is computed using the *tf.idf* method:

$$wd_{ij} = tf_{ij} \cdot \log_2 \left(\frac{M}{df_i} \right)$$

   Where $\mathrm{tf}_{\mathrm{ij}}$ is the frequency of term i in document j, df_{i} is the number of documents in which term i occurs, and M is the number of documents in the collection.

2. For each category name in the training collection, we have manually extracted a set of synonyms from WordNet. We have preformed a disambiguation process, selecting only a synset for each category. Using too many terms for each category may increase recall at the expense of a big decrease of precision, a problem that plagues studies in the utilization of WordNet for Information Retrieval [7].

3. After, we have built a linear classifier that assigns a document to a category if the similarity between them is over a predetermined threshold. The similarity is computed by the cosine formula as stated in the next section. The threshold is learned by ten-fold cross-validation on the training document collection.

[1]Available at http://www.dcs.gla.ac.uk/idom/ir_resources/linguistic_utils/stop_words

As described, the WordNet based classifier is not only based on WordNet but also on the training text collection. It is nearly impossible to build a text classifier for ATC without using documents in the training collection, because the meaning of the categories can only be guessed using sample documents. For instance, the category name "CAN" stands for "Canadian Dollar", that is, the category name is a currency code.

5 Learning based classifiers

In our work, we have selected as base classifiers two widely used in the literature as baselines for ATC [21], namely the Rocchio algorithm and the Naive Bayes classifier.

5.1 The Rocchio Algorithm

A very popular method for training a linear classifier is the Rocchio algorithm [16]. Linear classifiers are those able to perform classification in linear time. This kind of classifiers are based on computing a representation of a category as a term weight vector, somehow a prototype of the documents pertaining to the category [11, 21]. Usually, the prototype or model acquired in the learning phase, is compared during the classification of new documents using a semantic closeness measure like the cosine similarity in the VSM for IR [19]. More precisely, given three sets of N terms, M documents and L categories, and being the weight vector for document j $\langle \mathrm{wd_{1j}}, \mathrm{wd_{2j}}, \ldots , \mathrm{wd_{Nj}} \rangle$ and the weight vector for category k $\langle \mathrm{wc_{1k}}, \mathrm{wc_{2k}}, \ldots , \mathrm{wc_{Nk}} \rangle$, the similarity between document j and category k is obtained with the formula:

$$sim\left(d_j, c_k\right) = \frac{\sum_{i=1}^{N} wd_{ij} \cdot wc_{ik}}{\sqrt{\sum_{i=1}^{N} wd_{ij}^2} \cdot \sqrt{\sum_{i=1}^{N} wc_{ij}^2}}$$

The utilization of this formula makes classification a linear process. Usually, the document j is filed under category k if $\mathrm{sim(d_j, c_k)}$ exceeds a predetermined threshold derived on a validation set. Linear algorithms present interesting advantages that make them very suitable for industrial applications: linear classification complexity, a good degree of performance, and the possibility of being used within standard retrieval engines [21].

Rocchio is an algorithm typically used in IR for relevance feedback [19]. It was first adapted to TC in [9], and since then it has been an important reference in ATC. The Rocchio algorithm produces a new weight vector $\mathrm{wc_k}$ from an existing one $\mathrm{wci_k}$ and a collection of training documents. The component i of the vector $\mathrm{wc_k}$ is computed by the formula:

$$wc_{ik} = \alpha \cdot wci_{ik}^{0} + \beta \cdot \frac{\sum\limits_{l \in C_k} wd_{il}}{|C_k|} - \gamma \cdot \frac{\sum\limits_{l \notin C_k} wd_{il}}{P - |C_k|}$$

Where $\mathrm{wci_{ik}^0}$ is the initial weight of the term i for the category k, $\mathrm{wd_{il}}$ is the weight of the term i for the training document l, $\mathrm{C_k}$ is the set of indexes of documents assigned to the category k, and P the total number of documents in the training collection. The parameters α, β and γ control the relative impact of the initial, positive and negative weights respectively in the new vector. For example, Lewis used the values $\beta = 16$ and $\gamma = 4$ in [11].

5.2 The Naive Bayes classifier

Naive Bayes is a probabilistic prediction method based on an independence assumption. In some domains, its performance has been shown to be comparable to that of neural network and decision tree learning. Given a document represented as a term weight vector with N terms, the probability of the document being in the category k is given by the formula:

$$P\left(C_k|D\right) = \frac{P\left(D|C_k\right) \cdot P\left(C_k\right)}{P(D)}$$

A document D is filed under the k category if $P\left(C_k|D\right)$ is greater than $P\left(C_m|D\right)$ for every $m \neq k$. In this setting, it is assumed that categories are not overlapping, which is hard the case with the many text collections. For the case in which categories do overlap, the document D is filed under category k when $P\left(C_k|D\right) \geq 1/2$. This way, given M categories, one should train M independent Naive Bayes classifiers. Independence is also assumed among term occurrences, being $P\left(D|C_k\right)$ computed as:

$$P\left(D|C_k\right) = \prod_{i=1}^{N} P\left(wd_i|C_k\right)$$

Where $P\left(wd_i|C_k\right)$ is the probability of the weight of the i term being wd_i given the category k. It is remarkable that the independence assumption as stated here does not hold in general, but such a naive approach leads to good performance in practice [4]. To compute $P(D)$, the theorem of the total probability is used as follows:

$$P(D) = P\left(D|C_k\right) \cdot P\left(C_k\right) + P\left(D|\bar{C_k}\right) \cdot P\left(\bar{C_k}\right)$$

Finally, probabilities are computed using statistics from the training document collection. We have used the implementation of Naive Bayes in the Java WEKA library [25]. WEKA, which stands for Waikato Environment for Knowledge Analysis, is a Java library including the implementation of many learning algorithms, and facilities for input and output data processing[2]. Probabilities for classes, that is, $P\left(C_k\right)$, are computed using the so called *Laplacean estimator*, to ensure that an attribute value that occurs zero times receives a probability that is nonzero, albeit small. For probabilities of term weights given the categories, that is, $P\left(D|C_k\right)$, a normal distribution of weights is assumed (see [25] for more details).

6 Experiments with restricted Stacking and Rocchio

We have performed a first set of experiment designed to test the main idea of the integration by the model based on stacking (a kind of proof of concept). In this set of experiments, instead of making use of a second level learner, we have simply combined the predictions of both classifiers by giving them the same weight. This way, we can test the feasibility of the model based on stacking. Also, we have tested the integration at the feature level by learning a Rocchio classifier based in the terms of the training collection plus the terms extracted from WordNet. Since a 90% of the original terms has been previously eliminated in the previous step, it is possible in principle that only joining the remaining 10% terms obtained from de training collection, and the terms extracted from WordNet, we could train an effective classifier, avoiding the second level learning phase in the method we propose.

For the evaluation of our approach, we have selected the Reuters-21578 ATC Test Collection[3], nearly an standard in ATC evaluation [28]. The collection consists of 21,578 news stories dealing with financial topics and manually

[2]Available at http://www.cs.waikato.ac.nz/ml/weka/
[3]Available at http://www.research.att.com/∼lewis/reuters21578.html

classified according to a set of 135 economic subject codes. The Reuters collections has been divided in various training/test subsets in the literature. In one of the most popular, the ModLewis Split, there are 13,625 training documents and 6,188 test documents. Only 57% of the documents are assigned to one or more category (reaching up to 12 in some documents).

The evaluation metrics used are F_1 computed by macro and microaveraging [21], and the precision at eleven recall points [19]. The first metric is a balanced combination of recall and precision for binary classifiers. Recall represents the ability of a classifier when retrieving those documents that should be classified in a category. Precision represents the ability of a classifier for avoiding to classify in a category those documents that should not be filed in the category. Results are averaged over the full set of categories. When computed through macroaveraging, equal weight is given to each category. If microaveraging is used, the most populated categories are more influential in the results. The precision at eleven recall points is a popular Information Retrieval metric for showing the performance of text classifiers, that also gives equal weight to all categories.

In summary, we have tested three approaches: the Rocchio classifier without using information from WordNet (Rocchio), our approach based on restricted stacking (SimpleStack), and the method based on integration at the feature level (FeatRocchio), in which the Rocchio algorithm is used to learn a classifier from a set of words obtained by joining the words selected from the training collection and the words extracted from WordNet.

In the Table 1 we show the F_1 results of our experiments. As it can be seen, the SimpleStack approach is the best performing one. However, the integration of Wordnet at the feature level performs better than avoiding its use. These results are confirmed by the results in the Figure 1, where a precision graph is presented for the approaches tested. This precision graph has been obtained by linear interpolation of the precision values at the eleven recall points 0.0, 0.1, ..., 1.0. Since the biggest performance differences have been obtained for those metrics that give equal weight to all categories, we can also state that results are most improved for less frequent categories. This fact has a very intuitive interpretation: WordNet information is especially useful for categories with few training information.

As a conclusion, the results of our experiments show that:

- The integration of knowledge from WordNet is more effective than using only the training collection as knowledge source for learning. The

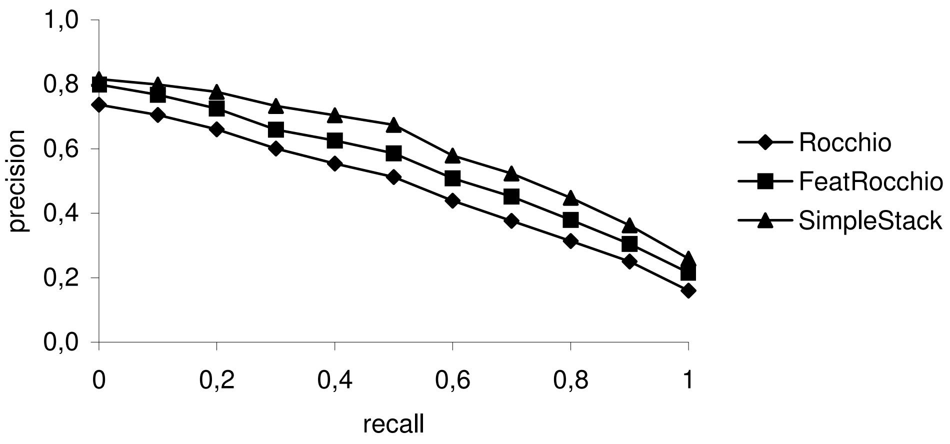

Figure 1: Recall precision graph for the Rocchio algorithm (Rocchio), the same algorithm with the terms extracted from WordNet (FeatRocchio), and for the model based in stacking (SimpleStack).

F_1	Rocchio	FeatRocchio	SimpleStack
macroavg.	.445	.498	.536
microavg.	.644	.647	.664

Table 1: Macro and microaveraged F_1 for the Rocchio algorithm (Rocchio), the same algorithm with the terms extracted from WordNet (FeatRocchio), and for the model based in stacking (SimpleStack).

approach based on stacking is thus promising.

- The integration of prior knowledge at the feature level is less effective than the stacking based model. This suggests that fully exploiting of external knowledge is only possible by using more sophisticated integration models as the one we propose.

F_1	NaiveBayes	Stacking
macroavg.	.100	.164
microavg.	.249	.231

Table 2: Macro and microaveraged F_1 for Naive Bayes (NaiveBayes), and the fully implemented Stacking method (Stacking).

7 Experiment with full Stacking and Naive Bayes

As the results from the previous section support our approach, we have developed a second series of experiments to test the method we propose. In this set of experiments, we have compared the performance of the WordNet based classifier, a Naive Bayes classifier, and the Stacking based integration using also a Naive Bayes classifier as the second level classifier, as implemented in WEKA.

The experiments have been run on the Reuters-21578 ATC Test Collection, but now we have chosen the ModApte Split. In this split, there are 9,603 documents in the training set, 3,299 documents in the test set, and unused 8,676. The difference is that only documents with at least one TOPICS category are used. The rationale for this restriction is that while some documents lack TOPICS categories because no TOPICS apply (i.e. the document is a true negative example for all TOPICS categories), it appears that others simply were never assigned TOPICS categories by the indexers. The evaluation metrics used are F_1 computed by macro and microaveraging. The results of our experiments are shown in Table 2.

The results of this series of experiments show that the Stacking method using WordNet outperforms Naive Bayes, but an small decrease in microaveraged F_1 is observed. This is due to the fact that giving more importance to categories with few training information impacts on categories with more training information. This fact deserves further investigation, remained for our future work. Also, performance is in general smaller than in the previous experiments. We interpret this fact as the orientation to general but not text data mining orientation of WEKA. We thus must to improve our utilization of WEKA, and in special, to test better performance algorithms for text classification (e.g. k Nearest Neighbors, Support Vector Machines, and boosting applied to C4.5 [21]).

8 Conclusions and Future Work

We have presented a method for the integration of lexical knowledge in ATC, which are based on the adaptation of the meta-learning schema stacked generalization. This method address the drawbacks of others work presented in the literature. The experiments we have performed demonstrate the feasibility of our approach. In detail, we have proven than integrating information from a lexical database like Wordnet leads to an improvement of performance of training based ATC approaches.

As future work, we plan to perform more experiments in order to check the potential of our method to improve performance of more challenging algorithms, and to fully exploit the potential of the WEKA library in text classification.

References

[1] E. Agirre and G. Rigau. Word sense disambiguation using conceptual density. In *In Proceedings of COLING'96*, pages 16–22, Copenhagen, Danmark, 1996.

[2] Manuel de Buenaga, José M. Gómez, and Belén Díaz. Using WordNet to complement training information in text categorization. In Ruslan Milkov, Nicolas Nicolov, and Nilokai Nikolov, editors, *Proceedings of RANLP-97, 2nd International Conference on Recent Advances in Natural Language Processing*, Tzigov Chark, BL, 1997.

[3] M. de Buenaga, J.M. Gómez, and B. Díaz. Using wordnet to complement training information in text categorization. In N. Nicolov and R. Mitkov, editors, *Recent Advances in Natural Language Processing II: Selected Papers from RANLP'97*, volume 189 of *Current Issues in Linguistic Theory (CILT)*, pages 353–364, Amsterdam/Philadelphia, 2000. John Benjamins.

[4] Pedro Domingos and Michael Pazzani. Beyond independence: conditions for the optimality of the simple Bayesian classifier. In *Proc. 13th International Conference on Machine Learning*, pages 105–112. Morgan Kaufmann, 1996.

[5] J.M. Gómez and M. de Buenaga. Integrating a lexical database and a training collection for text categorization. In *Proceedings of the ACL/EACL Work-*

shop on Automatic Information Extraction and Building of Lexical Semantic Resources for NLP, 1997.

[6] José M. Gómez, Manuel J. Maña, and Enrique Puertas. Combining text and heuristics for cost-sensitive spam filtering. In *Proceedings of the Fourth Computational Natural Language Learning Workshop, CoNLL-2000*, 2000.

[7] G. Gonzalo, F. Verdejo, I. Chugur, and J. Cigarran. Indexing with wordnet synsets can improve text retrieval. In *Proceedings of the COLING/ACL '98 Workshop on Usage of WordNet for NLP*, Montreal, 1998.

[8] Philip J. Hayes and Steven P. Weinstein. CONSTRUE/TIS: a system for content-based indexing of a database of news stories. In Alain Rappaport and Reid Smith, editors, *Proceedings of IAAI-90, 2nd Conference on Innovative Applications of Artificial Intelligence*, pages 49–66. AAAI Press, Menlo Park, US, 1990.

[9] D.A. Hull. Improving text retrieval for the routing problem using latent semantic indexing. In W.B. Croft and C.J. van Rijsbergen, editors, *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*. Springer Werlag, Heidelberg, DE, 1994.

[10] Markus Junker and Andreas Abecker. Exploiting thesaurus knowledge in rule induction for text classification. In Ruslan Milkov, Nicolas Nicolov, and Nilokai Nikolov, editors, *Proceedings of RANLP-97, 2nd International Conference on Recent Advances in Natural Language Processing*, pages 202–207, Tzigov Chark, BL, 1997.

[11] D.D. Lewis, R.E. Schapire, J.P. Callan, and R. Papka. Training algorithms for linear text classifiers. In H.P. Frei, D. Harman, P. Schäuble, and R. Wilkinson, editors, *Proceedings of SIGIR-96, 49th ACM International Conference on Research and Develxpment in Information Retrieval*, pages 294–306, Zürich, CH, 1996. ACM Press, New York, US.

[12] George A. Miller. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[13] Tom Mitchell. *Machine Learning*, chapter 12. McGraw-Hill, 1997.

[14] Dunja Mladenić. Turning YAHOO! into an automatic Web page classifier. In Henri Prade, editor, *Proceedings of ECAI-98, 13th European Conference on Artificial Intelligence*, pages 473–474, Brighton, UK, 1998. John Wiley and Sons, Chichester, UK.

[15] Kamal Nigam, Andrew K. McCallum, Sebastian Thrun, and Tom M. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.

[16] Joseph J. Rocchio. Relevance feedback in information retrieval. In Gerard Salton, editor, *The SMART retrieval system: experiments in automatic document processing*, pages 313–323. Prentice-Hall, Englewood Cliffs, US, 1971.

[17] Mehran Sahami, editor. *Proceedings of the 1998 Workshop on Learning for Text Categorization*, Madison, US, 1998. Available as Technical Report WS-98-05.

[18] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*. AAAI Tech. Rep. WS-98-05, 1998.

[19] G. Salton. *Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer.* Addison-Wesley, 1989.

[20] Sam Scott and Stan Matwin. Feature engineering for text classification. In Ivan Bratko and Saso Dzeroski, editors, *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pages 379–388, Bled, SL, 1999. Morgan Kaufmann Publishers, San Francisco, US.

[21] Fabrizio Sebastiani. A tutorial on automated text categorisation. In Analia Amandi and Ricardo Zunino, editors, *Proceedings of ASAI-99, 1st Argentinian Symposium on Artificial Intelligence*, pages 7–35, Buenos Aires, AR, 1999.

[22] K.M. Ting and I.H. Witten. Stacked generalization: when does it work? In *Proc International Joint Conference on Artificial Intelligence*, pages 866–871, 1997.

[23] L.A. Ureña, M. de Buenaga M., and J.M. Gómez. Integrating linguistic resources in tc through wsd. *Computers and the Humanities*, 2001. In press.

[24] P. Vossen. Eurowordnet: a multilingual database for information retrieval. In *Proceedings of the DELOS workshop on Cross-language Information Retrieval*, 1997.

[25] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* Morgan Kaufmann, 1999.

[26] David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

[27] Yiming Yang. An evaluation of statistical approaches to MEDLINE indexing. In J. J. Cimino, editor, *Proceedings of AMIA-96, Fall Symposium of the American Medical Informatics Association*, pages 358–362, Washington, US, 1996. Hanley and Belfus.

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

[29] T. Yokoi. The edr electronic dictionary. *Communications of the ACM*, 38(11), 1995.