Binary classifiers versus AdaBoost for labeling of digital documents*

Arturo Montejo-Ráez y **Luis Alfonso Ureña-López**
Department of Computer Science
University of Jaén, Spain
amontejo@ujaen.es, laurena@ujaen.es

Resumen: La asignación de términos de un vocabulario controlado (habitualmente un tesauro) a documentos en formato digital está abriendo la puerta a nuevas aplicaciones. En este artículo se comparan dos algoritmos avanzados para clasificación de documentos: la selección adaptativa de clasificadores base binarios y el algoritmo AdaBoost. Si bien ambos mostraron tiempos de respuesta similares, el primero proporcionó los mejores resultados sobre la partición *hep-ex* del corpus HEP, respaldando dicho método como una solución robusta al multi-etiquetado para grandes colecciones.
Palabras clave: clasificación automática de documentos, comparación de algoritmos, clasificación binaria, benchmark

Abstract: Assignment of labels from a controlled set of terms (usually a thesaurus) to digital version of documents is opening a wide range of new applications, now becoming powerful tools for digital libraries. In this paper we compare two different and advanced approaches for multi-label text categorization: the adaptive selection of binary base classifiers and the AdaBoost algorithm. Though both of them showed similar response times on producing final labels, the use of adaptive selection of binary classifiers performed better than AdaBoost on the *hep-ex* partition of the HEP corpus, confirming this method as a robust solution for multi-label of large collections.
Keywords: automatic text categorization, algorithms comparison, binary classification, benchmark

1 Introduction

Multi-label text categorization has emerged as the common way to obtain automated systems for document classification. It includes those were plain text documents are indexed with terms (also referred to as *key-words* or *descriptors*) selected from a controlled vocabulary. The integration of methods from the Information Retrieval domain with Machine Learning algorithms has promoted this area to a certain level of maturity, but the vast field being covered, and the wide range of components within a text categorization system hold the interest of many researchers from all around the world (Sebastiani, 2002).

Text categorization systems offer a large number of benefits to information systems in general and digital libraries in particular (Montejo-Ráez y Steinberger, 2004). Advances in automatic text categorization are of interest for current digital libraries, and the present paper provides relevant information for one of the most unknown but demanded classification paradigms: multi-label classification. For solving this task, two competitive algorithms have been tested against a pure multi-label collection.

2 Multi-label classification

In text categorization, three different classification paradigms can be identified:

- *Binary classification.* In this case, the classifier has to decide between two possible choices: YES/NO answers or two disjoint classes. This is the most common behaviour of well known classifiers, like Support Vector Machines (Joachims, 1998), PLAUM (Y. et al., 2002), Bayesian Logistic Regression (Genkin, Lewis, y Madigan, 2004) and many others.
- *Multi-class classification.* When the set

* This work has been partially supported by the Spanish Government under project R2D2-RIM (TIC2003-07158-C04-04).

of possible classes increases, but the sample must be assigned to just one target class, we are facing a multi-class classification problem. One approach is to use binary classifiers and then return the class with the highest *classification status value* (CSV), that is, the closest class. But it implies a basic premise: CSV must be comparable (which is not always true). Another choice is to use multi-class classifiers, like TiMBL (Daelemans et al., 2004), to perform the task straightforward.

- *Multi-label classification.* Here, a given document can be associated to a variable number of classes at the same time, so we have to decide not only if a class is close enough to the document, but also about the number of them to be selected. Methods enabled for solving this classification case are not common, despite the fact that multi-label classifiers can be constructed from binary classifiers (like in the case of the Adaptive Selection of Base Classifiers (Montejo-Ráez, Steinberger, y Ureña-López, 2004)) or generated using algorithms that rank all classes according to a coherent value (as in the case of the AdaBoost algorithm (Schapire y Singer, 2000)).

In specialized domains, like Engineering (INS,)), Biology (Vieduts-Stokolo, 1987) or Physics (DESY, 1996), particular sets of concepts have been arranged into groups, and certain relations have been established among them. These are the thesauri, well known by librarians. This multi-labeling is not easy to automate, mainly due to the lack of data, the inaccuracy of human indexers, the lack of agreement on which keywords should be selected according to each expert's criteria and, of course, the high imbalance show by these assignments (Montejo-Ráez, Steinberger, y Ureña-López, 2004).

The two methods compared here have been identified as promising solutions for the automatizing of the indexing process of documents by keyword assignment. We will briefly introduce them and then show the results obtained by applying these algorithms on a benchmark set of multi-labeled documents.

2.1 AdaBoost

Boosting is a technique in machine learning for combining classifiers in an iterative way so that performance is improved. The final classifier is composed by several *weak* classifiers, which have been generated through the learning process. A more formal definition would describe if as follows:

```
**Given**:
    $(x_1, y_1), \cdots, (x_m, y_m)$
    where $x_i \in X, y_i \in Y = \{-1, +1\}$
**For** $t = 1, \cdots, T$:
    Train weak learner and produce weak
    hypothesis $h_t : X \to \{-1, +1\}$
    Update some weights and parameters
**Return** final hypothesis $H$ as combination
    of $h_1, \cdots, h_T$
```

AdaBoost is one example of these techniques (Schapire y Singer, 2000). It works by maintaining a distribution of weights (one per sample) that is updated to force weak learners concentrate on those samples whose classification is "harder". The AdaBoost algorithm using Hamming distance for computing the loss is shown in figure 1. As we can see, this algorithm (a) increases the weight of miss-classified samples at each round, and (b) the final hypothesis H is a weighted combination of T weak hypothesis h_t in function of their error.

It is different to previous boosting algorithms in that it is *adaptive*, since it depends on error rates of each individual weak hypothesis (hence its name). The benefits of this algorithm are due to mathematical properties of the training error and the generalization error. The first one implies that AdaBoost does not need a lower bound a priory to work, since it is computed adaptively. The second one, as proved by Schapire and his colleagues, tells us about the independence of T from the upper bound of the generalization error, which makes the algorithm robust against over-fitting.

Two variants of this algorithm were proposed to deal with multi-class classification: *AdaBoost.MH*, defined on the basis of the *Hamming loss* minimization, and *AdaBoost.MR*, which relies on ranking loss minimization. Whether we choose a loss to minimize or another, we have to design how weak learners are generated. Schapire proposes a very simple one for text classification: each weak hypothesis just returns a real

```
Given:
    $(x_1, y_1), \cdots, (x_m, y_m)$
    where $x_i \in X, y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = 1/m$
For $t = 1, \cdots, T$:
    · Train weak learner using distribution $D_t$
    · Get weak hypothesis $h_t : X \rightarrow \{-1, +1\}$
    with error:
        $E_t = P_{r_i \sim D_t}[h_t(x_i) \neq y_i] =$
        $= \sum_{i:h_t(x_i)\neq y_i} D_t(i)$
    · Let $\beta_t = \frac{1}{2}\ln(\frac{1-E_t}{E_t})$
    · Update D:
        $D_{t+1}(i) = \frac{D_t(i)}{Z_t}\times$
        $\times \begin{cases} e^{-\beta_t} & \text{if } h_t(x_i) = y_i, \\ e^{\beta_t} & \text{if } h_t(x_i) \neq y_i, \end{cases} =$
        $= \frac{D_t(i)exp(-\beta_t y_i h_t(x_i))}{Z_t}$
        where $Z_t$ is a normalization factor
        so that $D$ is a distribution
    · Output final hypothesis:
        $H(x) = sign(\sum_{t=1}^{T} \beta_t h_t(X))$
```

Figure 1: AdaBoost.MH algorithm

value when a certain term appears and another value if the term is not present in the document. This value use to be the score Z_t in AdaBoost.MH and an approximation of Z_t in AdaBoost.MR (since there is no analytical solution).

Results shown by Schapire and Singer on the Reuters-21578 corpus (Schapire y Singer, 2000) promoted AdaBoost.MH outperforming other algorithms like Rocchio, Naïve Bayes and Sleeping Experts. Unfortunately, in our opinion the Reuters-21578 collection is not a very suitable collection for performing multi-label classification experiments due to the low number of categories assigned to each document (rather few, two or three categories per sample in rare cases). Similar results were found also on the Reuters-21578 collection (Weiss et al., 1999).

2.2 Adaptive selection of base classifiers

In the *Adaptive Selection of Base Classifiers* approach (Montejo-Ráez, Steinberger, y Ureña-López, 2004) we basically train a system using the battery strategy (many classifiers working together independently), but *(a)*, we allow tuning the binary classifier for a given class by a balance factor, and *(b)* we provide the possibility of choosing the best of a given set of binary classifiers. Furthermore, we intended to apply our classification system to *real time* environments so that a gain in classification speed was very important. Therefore, the algorithm introduces the α parameter, resulting in the algorithm given in figure 2. This value is a threshold for the minimum performance allowed to a binary classifier during the validation phase in the learning process. If the performance of a certain classifier is below the value α, meaning that the classifier performs badly, we discard the classifier and the class completely. By doing this, we may decrease the recall slightly (since less classes get trained and assigned), but we potentially may decrease computational cost, and increase precision. The effect is similar to that of the *SCutFBR* (Yang, 2001). We never attempt to return a positive answer for rare classes. In (Montejo-Ráez, Steinberger, y Ureña-López, 2004), it is shown how this filtering saves us considering many classes without significant loss in performance.

```
Input:
    a set of training documents $D_t$
    a set of validation documents $D_v$
    a threshold $\alpha$ on the evaluation measure
    a set of possible label (classes) $L$,
    a set of candidate binary classifiers $C$
Output :
    a set $C' = \{c_1, ..., c_k, ..., c_{|L|}\}$ of trained
    binary classifiers
Pseudo code:
    $C' \leftarrow \emptyset$
    for-each $l_i$ in $L$ do
        $T \leftarrow \emptyset$
        for-each $c_j$ in $C$ do
            train-classifier($c_j$, $l_i$, $D_t$)
            $T \leftarrow T \cup \{c_j\}$
        end-for-each
        $c_{best} \leftarrow$ best-classifier($T$, $D_v$)
        if evaluate-classifier($c_{best}$) $> \alpha$
            $C' \leftarrow C' \cup \{c_{best}\}$
        end-if
    end-for-each
```

Figure 2: The one-against-all learning algorithm with classifier filtering

For every classifier training, positive samples are over-weighted by the fraction of negative samples over positive ones, that is, the weight for positive samples (w_+) is:

$$w_+ = C_-/C_+$$

where

C_- is the total number of negative samples for the class

C_+ is the total number of positive samples for the class

As we can see, the more positive documents we have for a given class, the lower the over-weight is, which makes sense in order to give more weight only when few positive samples are available for that class. This method was used previously (Morik, Brockhausen, y Joachims, 1999) but they did not report how much it improved the performance of the classifier over the non-weighted scheme. As we said, this w_+ factor was used in our experiments to over-weight positive samples over negative ones, i.e. the classification error on a positive sample is higher than that of a negative one.

Besides this, the *S-cut* approach was integrated into the algorithm. The assignment of a sample as positive can be tuned by specifying the decision border. By default it is zero, but it can be set using the S-Cut algorithm (Yang, 2001). This algorithm uses as threshold the one that gives the best performance on an evaluation set. That is, once the classifier has been trained, we apply it against an evaluation set using classification values (e.g. the margin for SVM) as possible candidate thresholds. According to S-Cut definition, the threshold that reported the best performance (the highest F1 in our case) is used as decision boundary for that class.

3 The HEP collection

The HEP corpus[1] is a collection of papers related to *High Energy Physics*, and manually indexed with DESY labels. These documents have been compiled by Montejo-Ráez and Jens Vigen from the CERN Document Server[2] and have motivated intensive study of text categorization systems in recent years (Dallman y Meur, 1999; Montejo-Ráez y Dallman, 2001; Montejo-Ráez, 2002; Montejo-Ráez, Steinberger, y Ureña-López, 2004).

No. docs.	Keyword
1898 (67%)	electron positron
1739 (62%)	experimental results
1478 (52%)	magnetic detector
1190 (42%)	quark
1113 (39%)	talk
715 (25%)	Z0
676 (24%)	anti-p p
551 (19%)	neutrino
463 (16%)	W
458 (16%)	jet

Figure 3: The ten most frequent main key words in the *hep-ex* partition

In these experiments we have used the *hep-ex* partition of the HEP collection, composed by 2,839 abstracts related to experimental High-Energy Physics that are indexed with 1,093 main keywords, with an average number of classes per document slightly above 11. This partition is highly imbalanced: only 84 classes are represented by more than 100 samples and only five classes by more than 1,000. The uneven use is particularly noticeable for the ten most frequent key words: In table 3 the left column shows the number of positive samples of a key word and the percentage over the total of samples in the collection.

4 Experiments and results

Once that the collection and the two approaches studied have been introduced, it is time to show the results obtained. For performing these experiments we have used the TECAT[3] implementation of the adaptive selection algorithm, and the BoosTexter[4] code for running the AdaBoost method.

About the plain text collection, we have only used the abstract field of the meta-data set in the *hep-ex* partition. The keywords have been preprocessed to only consider first-level ones (known in DESY thesaurus as *primary* keywords). Reactions and energy related keywords have not been considered, because they are actually very specialized enti-

[1]The collection is freely available for academic purposes from: `http://sinai.ujaen.es/wiki/index.php/HepCorpus`

[2]`http://cds.cern.ch`

[3]Available at `http://sinai/wiki/index.php/TeCat`

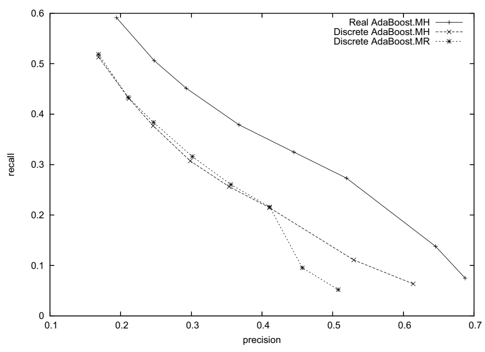
[4]Available at `http://www.cs.princeton.edu/~schapire/BoosTexter/`

ties that should be detected and recognized from the full-text version. Abstracts have been processed as follows:

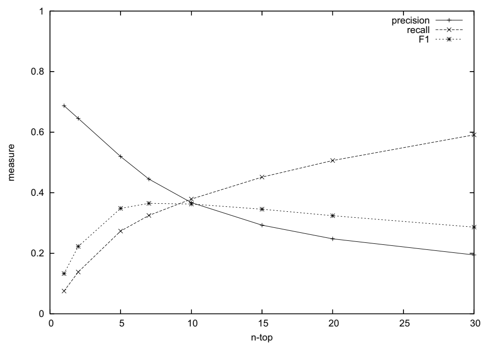
- Punctuation was removed
- Every character was lower-cased
- Stop words were removed
- The Porter stemming algorithm (Porter, 1997) was applied
- Resulting stems were weighted according to the TF.IDF scheme (Salton, Wong, y Yang, 1974)

For the evaluation of experiments, *ten-fold cross validation* (Kohavi, 1995) was used in order to produce statistically relevant results that do not depend on the partitioning of the collection into training, evaluation and test sets. Extensive experiments have shown that this is the best choice to get an accurate estimate. The measures have been computed by macro-averaging per document (that is, they are computed for every document and then averaged). These measures are *precision* and *recall*. The F_1 measure (van Rijsbergen, 1975)) is used as an overall indicator based on the two former ones and is the reference chosen when filtering is applied. Final values are computed using macro-averaging on a per-document basis, rather than the usual micro-averaging over classes. The reason is, again, the high imbalance in the collection. If we average by class, rare classes will influence the result as much as the most frequent ones, which will not provide a good estimate on the performance of the multi-label classifier over documents. Since the goal of this system is to be used for automated classification of individual documents, we considered to be far more useful to concentrate on these measurements for our evaluation of the system. More details about these concepts can be found in (Sebastiani, 2002), (Lewis, 1991) and (Yang y Liu, 1999).

BoosTexter offers three possible versions of the algorithm: *Real AdaBoost.MH*, which uses real values for the classification status value (CSV) of every weak learner and *Hamming loss* as bound error, *Discrete AdaBoost.MH* with discrete values as class decisions (CSV), and *Discrete AdaBoost.MR* with discrete CSV and designed for ranking correct classes on the top. In figure 4a we can notice how they differ in performance when



(a) AdaBoost versions



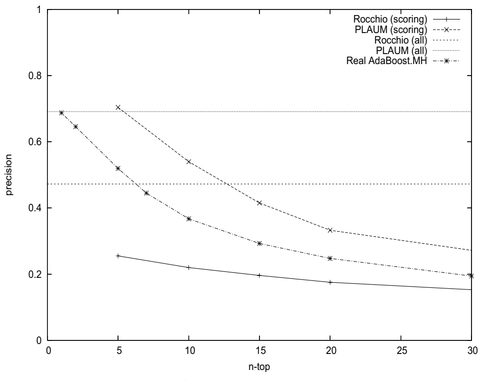
(b) Effects of scoring

Figure 4: Behaviour of the AdaBoost algorithm on the *hep-ex* partition

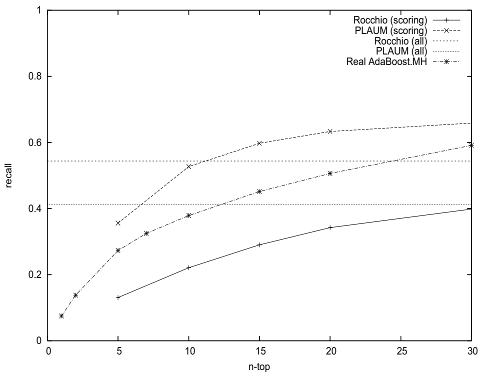
considering scoring (each point is obtained considering a given number of classes per document), that is, when ranking all classes by their classification status value and selecting the ones with the highest rank. From this graph the main conclusion is that the Real AdaBoost.MH algorithm widely outperforms discrete versions. In figure 4b the effect of assigning a given number of classes to the document is drawn. When 10 classes are considered, all measures meet at the same value, and it is also evident the obvious behaviour for precision and recall depending on the number of classes assigned.

On the other hand, TECAT has been run with four possible configurations where two different algorithms (Rocchio and PLAUM) have been studied in order to show how scoring affects performance. The configurations are as follows:

- *Rocchio (all).* The linear Rocchio algo-



(a) Precision

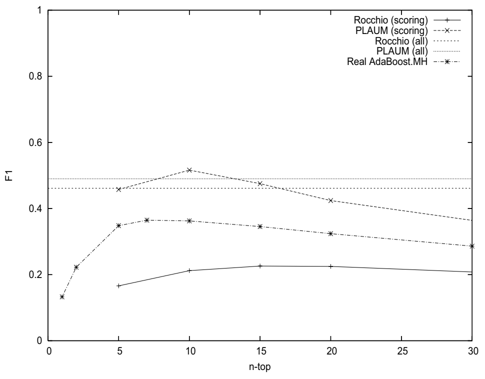


(b) Recall

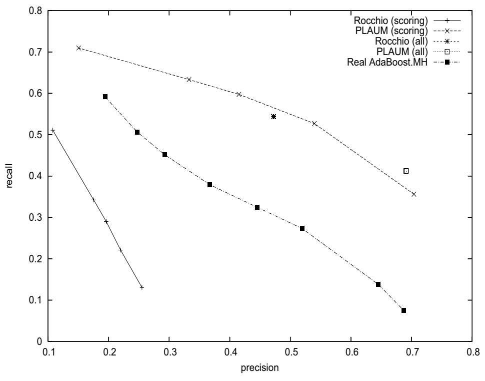
Figure 5: Results using scoring for studied methods

rithm is used as base classifier (Lewis et al., 1996) with *S-cut* thresholding (Yang, 2001)

- *Rocchio (scoring)*. The assignment is performed taking the *n-top* classes. We have tried with a variable number of them (n) and found that 10 achieves a good compromise between precision and recall. In fact, the average number of classes per document in the *hep-ex* partition is 11,06.
- *PLAUM (all)*. This perceptron is used as base classifier, in this configuration all positive values returned by the classifier are assigned, so no selection by rank is done.
- *PLAUM (scoring)*. Finally, as for Rocchio, here we select top classes by ranking their classification status values. Here the system knows beforehand how many classes are assigned to every document.



(a)



(b)

Figure 6: (a) results over F1 measure when using scoring, and (b) overall view of different approaches

Results obtained were very clear: TECAT outperforms Boostexter on any situation. In figures 5a and 5b values of precision and recall respectively reached by all the studied approaches are represented. Straight lines are values obtained for those algorithms that do not consider scoring (so the assignment of the class is decided according to a threshold on the CSV). Though highest values on these measures are reached by scoring strategies, the overall performance measured on the F1 value tells us that non-scoring strategies are better in general (see figure 6a), except in the case of the PLAUM algorithm with a fixed number of classes equal to 10.

5 Conclusions and open issues

From results above, we can summarize some empirical conclusions:

- Scoring is not a desirable approach for multi-label text categorization of HEP related corpora. We can see how the Rocchio algorithm obtains a much better precision and recall when each classifier takes the decision of whether to assign the associated class or not, i.e. when we compute a different Scut on each class.
- In the case of PLAUM, only scoring with a number of 10 classes is a better choice than non-scoring assignment. But this implies the study of the best number of classes, so that number may not be that good for other collections. In general, we would suggest also in this case the use of selection by thresholding (CSVs over 0 in the case of PLAUM).
- AdaBoost is able to perform multi-label classification on the HEP collection, but its performance has been reported to be significantly worse than that offered by the adaptive strategy.

Among the possible lines that we should follow in the future, we have identified two interesting topics:

- Set a different cut (threshold) for each class using AdaBoost may improve the multi-label classification. This can be done not as part of the AdaBoost algorithm, since an approach like the Scut threshold can only be computed using a validation set.
- Adaptive selection of base classifiers are not better than AdaBoost at the same level, that is, the adaptive selection is an strategy for binary classifiers integration focused on multi-label categorization. Therefore, it is possible to integrate AdaBoost as base classifier into the adaptive selection algorithm and to study if this improves current results.

Both approaches are similar in that both algorithms integrate base classifiers (known as *weak learners* in boosting). The behaviour of AdaBoost with more accurate classifiers, like SVM (Li, Wang, y Sung,), should be studied. Anyhow, the adaptive selection is a very interesting and cheap strategy for classifiers integration oriented to solve multi-label assignments.

Bibliografía

INSPEC Thesaurus. http://www.iee.org.uk/publish/inspec/.

Bruno Pouliquen, Ralf Steinberger, y Camelia Ignat. 2003. Automatic Annotation of Multilingual Text Collections with a Conceptual Thesaurus. En Amalia Todirascu, editor, *Proceedings of the workshop 'Ontologies and Information Extraction' at the EuroLan Summer School 'The Semantic Web and Language Technology'(EUROLAN'2003)*, página 8 pages, Bucharest (Romania).

Daelemans, Walter, Jakub Zavrel, Ko van der Sloot, y Antal van den Bosch. 2004. Timbl: Tilburg memory based learner, version 5.1, reference guide. Informe Técnico 04–02, ILK Research Group, Tilburg University.

Dallman, D. y J. Y. Le Meur. 1999. Automatic keywording of High Energy Physics. En *4th International Conference on Grey Literature : New Frontiers in Grey Literature Washington, DC, USA*, Oct.

DESY. 1996. Guide for the use of the high energy physics index keywords 1996. http://www-library.desy.de/guide1.html.

Genkin, Alexander, David D. Lewis, y David Madigan. 2004. Large-scale bayesian logistic regression for text categorization. Informe técnico, Center for Discrete Mathematics and Theoretical Computer Science.

Hersh, William, Chris Buckley, T. J. Leone, y David Hickam. 1994. Ohsumed: an interactive retrieval evaluation and new large test collection for research. En *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, páginas 192–201. Springer-Verlag New York, Inc.

Joachims, Thorsten. 1998. Text categorization with support vector machines: learning with many relevant features. En Claire Nédellec y Céline Rouveirol, editores, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, numero 1398, páginas 137–142, Chemnitz, DE. Springer Verlag, Heidelberg, DE.

Kohavi, Ron. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. En *Proc. of the Fourteenth International Joint Conference on Artificial Intelligence*, páginas 1137–1145. Morgan Kaufmann, San Mateo, CA.

Lewis, D. D. 1991. Evaluating Text Categorization. En *Proceedings of Speech and Natural Language Workshop*, páginas 312–318. Morgan Kaufmann.

Lewis, D. D., Robert E. Schapire, James P. Callan, y Ron Papka. 1996. Training algorithms for linear text classifiers. En Hans-Peter Frei Donna Harman Peter Schäuble, y Ross Wilkinson, editores, *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, páginas 298–306, Zürich, CH. ACM Press, New York, US.

Li, Xuchun, Lei Wang, y Eric Sung. Adaboost with svm based component classifiers. under review of IEEE Transactions on System, Man and Cybernetics, part B.

Montejo-Ráez, A. 2002. Towards conceptual indexing using automatic assignment of descriptors. Workshop in Personalization Techniques in Electronic Publishing on the Web: Trends and Perspectives, Málaga, Spain, May.

Montejo-Ráez, A. y D. Dallman. 2001. Experiences in automatic keywording of particle physics literature. *High Energy Physics Libraries Webzine*, (issue 5), November. URL: http://library.cern.ch/HEPLW/5/papers/3/.

Montejo-Ráez, A y R. Steinberger. 2004. Why keywording matters. *High Energy Physics Libraries Webzine*, (Issue 10), December.

Montejo-Ráez, A., R. Steinberger, y L. A. Ureña-López. 2004. Adaptive selection of base classifiers in one-against-all learning for large multi-labeled collections. (3230):1–12.

Morik, Katharina, Peter Brockhausen, y Thorsten Joachims. 1999. Combining statistical learning with a knowledge-based approach - a case study in intensive care monitoring. En *Proc. 16th International Conf. on Machine Learning*, páginas 268–277. Morgan Kaufmann, San Francisco, CA.

Porter, M. F., 1997. *An algorithm for suffix stripping*, páginas 313–316. Morgan Kaufmann Publishers Inc.

Salton, Gerard, A. Wong, y C. S. Yang. 1974. A Vector Space Model for Automatic Indexing. Technical Report TR74-218, Cornell University, Computer Science Department, Julio.

Schapire, Robert E. y Yoram Singer. 2000. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168.

Sebastiani, Fabrizio. 2002. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47.

van Rijsbergen, C. J. 1975. *Information Retrieval*. London: Butterworths. http://www.dcs.gla.ac.uk/Keith/Preface.html.

Vieduts-Stokolo, Natasha. 1987. Concept recognition in an automatic text-processing system for the life sciences.

Weiss, S. M., C. Apt'e, D. Damerau, D. E. Johnson, F. J. Oles, T. Goetz, y T. Hampp. 1999. Maximizing text-mining performance. *IEEE Intelligent Systems*, 14(4):63–69.

Y., Li, Zaragoza H., Herbrich R., Shawe-Taylor J., y Kandola J. 2002. The perceptron algorithm with uneven margins. En *Proceedings of the International Conference of Machine Learning (ICML'2002).*

Yang, Yiming. 2001. A study on thresholding strategies for text categorization. En W. Bruce Croft David J. Harper Donald H. Kraft, y Justin Zobel, editores, *Proceedings of SIGIR-01, 24th ACM International Conference on Research and Development in Information Retrieval*, páginas 137–145, New Orleans, US. ACM Press, New York, US. Describes RCut, Scut, etc.

Yang, Yiming y Xin Liu. 1999. A re-examination of text categorization methods. En Marti A. Hearst Fredric Gey, y Richard Tong, editores, *Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval*, páginas 42–49, Berkeley, US. ACM Press, New York, US.