Automated Classification of Web Documents into a Hierarchy of Categories

Michelangelo Ceci, Floriana Esposito, Michele Lapi, and
Donato Malerba

Dipartimento di Informatica, Universitá degli Studi, via Orabona 4,
70126 Bari, Italy
{ceci, esposito, lapi, malerba}@di.uniba.it

Abstract. In this paper, the problem of classifying a HTML documents into a hierarchy of categories is investigated in the context of cooperative information repository, named WebClassII. The hierarchy of categories is involved in all aspects of automated document classification, namely feature extraction, learning, and classification of a new document. Innovative aspects of this work are: a) an experimental study on actual Web documents which can be associated to any node in the hierarchy; b) the feature selection process; c) the automated selection of thresholds for the score returned by a classifier; d) the comparison of three different techniques (flat, hierarchical with proper training sets, hierarchical with hierarchical training sets); e) the definition of new measures for the evaluation of system performances. Results show that the use of hierarchical training sets improves the hierarchical techniques.

Keyword: web content mining, hierarchical document classification.

1 Introduction

In cooperative information repositories the manual indexing of documents is effective only when all information providers have a thorough comprehension of the underlying shared ontology. However, an experimental study on manual indexing for Boolean information retrieval systems has shown that the degree of overlap in the keywords selected by two similarly trained people to represent the same document is not higher than 30% on average [3]. Therefore, to facilitate document sharing, it is important to develop tools that assist users in the process of document classification.

WebClass [9] is a client-server application that has been designed to support the search activity of a geographically distributed group of people with common interests. It works as an intermediary when users browse the Web through the system and categorize documents by means of one of the classification techniques available. Automated classification of Web pages is performed on the basis of their textual content and may require a preliminary training phase in which document classifiers are built for each document category on the basis of a set of training examples.

A simplifying assumption made in the design of WebClass is that document categories are not hierarchically related. This permits the system to build either a unique classifier for all categories or a classifier for each category independently of the others (*flat classification*). However, in many practical situations categories are organized hierarchically which is essential when the number of categories is quite high, since it supports a thematic search by browsing topics of interests.

In this work we present an upgrade of some techniques implemented in WebClass to the case of Web documents organized in a *hierarchy of categories*, that is, a tree structure whose nodes and leaves are document categories. The upgrading of the techniques involved all aspects of automated document classification, namely:

- the definition of a document representation language (*feature extraction*),
- the construction of document classifiers (*learning*), and
- the *classification* of a new document according to the hierarchy of categories.

The paper is organized as follows. In the next section, we introduce some issues related to upgrading the document classification techniques. In Section 3 we describe a new feature selection process for document classification, while in Section 4 we present the naïve Bayes and the centroid-based classification techniques as well as the automated threshold determination algorithm. Section 5 is devoted to the explanation of the document classification process. All these techniques have been implemented in a new version of the WebClass system, named WebClassII. Finally, some experimental results are reported and commented in Section 6.

2 Hierarchical document classification

In flat classification, a unique set of features is extracted from the training documents belonging to several distinct categories [6]. The uniqueness of the feature set permits the application of several statistical and machine learning algorithms defined for multi-class problems. However, this approach is impractical when the number of categories is high. In the case of hierarchically related categories it would be difficult to select a proper set of features, since documents concerning general topics are well represented by general terms like "mathematics", while documents concerning specific topics (e.g., trigonometry) are better represented by specific terms like "cosine". By taking into account the hierarchy of categories, it is possible to define several representations (sets of features) for a document. Each of them is useful for the classification of a document at one level of the hierarchy. In this way, general and specific terms are not forced to coexist in the same feature set.

As for the learning process, it is possible to consider the hierarchy of categories in the definition of the training sets. In particular, training sets

can be specialized for each internal node of the hierarchy by considering only documents of the subhierarchy rooted in the internal node (*hierarchical training set*). This is an alternative to using all documents for each learning problem like in flat classification.

In the classification process, considering the problem hierarchically reduces the number of decisions that each classifier has to take and increases its accuracy. Indeed, the problem is partitioned into smaller subproblems, each of which can be effectively and efficiently managed on the contrary, in flat classification with r categories the classifier has to choose one of r. This can be difficult in the case of large values of r and may lead to inaccurate decisions.

Some of these aspects have been considered in several works [8], [10], [11], [4], [5]. Our work differs from previous studies in several respects. First, documents can be associated to both internal and leaf nodes of the hierarchy. Surprisingly, this aspect is considered only in [11]. However, differently from Mladenic's work, we consider actual Web documents referenced in the Yahoo! ontology, and not only the items which briefly describe them in the Yahoo! Web directories (see Fig. 1). This is the situation that we expect to have in cooperative web repositories indexed by a hierarchy of categories.

A second difference is in the feature selection process for each internal category. It is based on an upgrade of the technique implemented and tested in WebClass, named TF-PF^2-ICF. Indeed, a comparison with other two well-known feature selection measures showed better results in the case of flat classification [9].

The third innovative contribution is in the development of a technique for the automated selection of thresholds both for posterior probabilities (in the case of naïve Bayes classifiers) and for similarity measures (in the case of centroid-based classifiers). The thresholds are used to determine whether a document has to be passed down to the one of the child categories during the top-down classification process.

The fourth difference is the comparison of system performances on two types of training sets definable for a given category: i) a *proper* training set, which includes documents of the category (positive examples) and documents of the sibling categories (negative examples), and ii) a *hierarchical training set*, which includes documents of the subtree rooted in the category (positive examples) and documents of the sibling subtrees (negative examples).

Finally, we define new measures for the evaluation of the system performances so to capture some aspects related to the "semantic" closeness of the predicted category from the actual one.

3 The feature selection process

In WebClassII each document is represented as a numerical feature vector, where each feature corresponds to the occurrence of a particular word in

the document. In this representation, no ordering of words or any structure of text is used. The feature set representation is unique for each category. It is automatically determined by means of a set of positive and negative training examples. All training documents are initially tokenized, and the set of tokens (words) is filtered in order to remove HTML tags, punctuation marks, numbers and tokens of less than three characters. Standard text preprocessing methods are used to:

1. remove words with high frequency, or *stopwords*, such as articles,
2. determine equivalent stems (*stemming*) by means of Porter's algorithm for English texts [12].

Only relevant tokens are considered in the feature set. Many techniques have been proposed in the literature on information retrieval for the identification of relevant words to be used as index terms of documents [13]. However, they are not appropriate for the task of document classification, where words are selected for discrimination purposes and not indexing. For two-class problems, Mladenic [11] compared scoring measures based on the *Odds ratio* and those based on *information gain*, leading her to favor the former. For multi-class problems, as in the case of WebClassII, Malerba et al. [9] developed a feature selection procedure based on an extension of the well-known TF-IDF measure. Here we briefly present its extension to the case of hierarchical training sets.

Let c be a category and c' one of its children in the hierarchy of categories, that is, $c' \in SubCategories(c)$. Let d be a training document from c', w a feature extracted from d (after the tokenizing, filtering and stemming steps) and $TF_d(w)$ the relative frequency of w in d. Then, the following statistics can be computed:

- the maximum value of $TF_d(w)$ on training documents d of category c',

$$TF_{c'}(w) = \max_{d \in Training(c')} TF_d(w) \tag{1}$$

- the percentage of documents in c' where w occurs (*page frequency*)

$$PF_{c'}(w) = \frac{occ_{c'}(w)}{|Training(c')|} \tag{2}$$

where $Training(c')$ is the training set for category c' (see section 6).

The union of feature sets extracted from Web pages of c' defines an "empirical" *category dictionary* used by documents on the topics specified by that category. By sorting the dictionary with respect to $TF_{c'}(w)$, words occurring frequently only in one long HTML page might be favored. By sorting each category dictionary according to the product $TF_{c'}(w) \cdot PF_{c'}(w)^2$, the effect of this phenomenon is kept under control.[1] Moreover, common words used in

[1] The plain $PF_{c'}(w)$ factor was also used, but was found to reduce performance slightly. For small sets of training documents, the term $PF_{c'}(w)$ might not be small enough to reduce the effect of very frequent words in single documents.

documents of c' will appear in the first entries of the corresponding category dictionary. Some of these words are actually specific to that category, while others are simply common English words (e.g., "information", "unique", and "people") and should be considered as *quasi-stopwords*. In order to move quasi-stopwords down in the sorted dictionary, the value $TF_{c'}(w) \cdot PF_{c'}(w)^2$ is multiplied by a factor $ICF_c(w) = 1/CF_c(w)$, where $CF_c(w)$ (*category frequency*) is the number of categories $c" \in SubCategories(c)$ in which the word w occurs. In this way, the relevant features that discriminate documents of c' from documents of its sibling categories can be found in the first entries of the category dictionary for c'. If n_{dict} is the maximum number of features selected for a category dictionary, then $Dict_{c'} = [(w_1, v_1), (w_2, v_2), \ldots, (w_k, v_k)]$ such that $\forall i \in [1 \cdots k]$ w_i is a feature extracted from some document d in c', $v_i = TF(w_i) \cdot PF^2_{c'}(w_i) \cdot 1/CF_c(w_i)$ and $k \leq n_{dict}$ (with k=n_{dict} when at least n_{dict} features can be extracted from training documents of category c'). The feature set (FeatSet$_c$) associated to a category c is the union of Dict$_{c'}$ for all its subcategories c'. It contains features that appear frequently in many documents of one of the subcategories but seldom occur in documents of the other subcategories (*orthogonality of category features*). In other words, selected features decrease the intra-category dissimilarity and increase the inter-category dissimilarity. Therefore, they are useful to classify a document (temporarily) assigned to c as belonging to a subcategory of c itself. It is noteworthy that this approach returns a set of quite general features for upper level categories, and a set of specific features for lower level categories.

Once the feature set has been determined, training documents can be represented as numerical feature vectors whose values are frequencies.

4 The learning process

WebClassII has two ways of assigning a Web document d to a category c_i:

1. By computing the similarity between d and the *centroid* of c_i.
2. By estimating the posterior probability for c_i given d (*naïve Bayes*).

Therefore, a training phase is necessary either to compute the centroids of the categories or to estimate the posterior probability distributions. Let us suppose that d has been temporarily assigned to a category c. We intend to classify d into one of the subcategories of c. According to the Bayesian theory, the optimal classification of d assigns d to the category $c_i \in SubCategories(c)$ maximizing the posterior probability $P_c(c_i|d)$. Under the assumption that each word in d occurs independently of other words, as well as independently of the text length, it is possible to estimate the posterior probability as follows (adapted from [7]):

$$P_c(c_i|d) = \frac{P_c(c_i) \cdot \prod_{w \in FeatSet_c} P_c(w|c_i)^{TF(w,d)}}{\sum_{c' \in SubCategories(c)} P_c(c') \cdot \prod_{w \in FeatSet_c} P_c(w|c')^{TF(w,d)}} \quad (3)$$

where the prior probability $P_c(c_i)$ is estimated as follows:

$$P_c(c_i) = \frac{|Training(c_i)|}{\sum_{c' \in SubCategories(c)} |Training(c')|} = \frac{|Training(c_i)|}{|Training(c)|} \quad (4)$$

and the likelihood $P_c(w|c_i)$ is estimated according to Laplace's law of succession:

$$P_c(w|c_i) = \frac{1 + PF(w, c_i)}{|FeatSet_c| + \sum_{w' \in FeatSet_c} PF(w', c_i)} \quad (5)$$

In (3)-(5), $TF(w, d)$ and $PF(w, c)$ denote the absolute frequency of w in d and the absolute frequency of w in documents of category c, respectively.

The *centroid* of a category is defined as a feature vector whose components are computed by averaging on the corresponding feature values of all training documents of the category. In order to classify a document d, the centroid most similar to the description of d is sought. Similarity is measured by means of *cosine correlation*, which computes the angle spanned by two feature vectors (d and the centroid):

$$P_c(w, c_i) = \frac{\sum_{d \in Training(c_i)} TF_d(w)}{|Training(c_i)|} \quad (6)$$

$$Sim_c(c_i, d) = \frac{\sum_{w \in FeatSet_c} P_c(w, c_i) \cdot TF_d(w)}{\sqrt{\sum_{w \in FeatSet_c} P_c(w, c_i)^2 \cdot \sum_{w \in FeatSet_c} TF_d(w)^2}} \quad (7)$$

The cosine correlation is particularly meaningful when features define orthogonal directions and vectors are highly dimensional. Both conditions are satisfied in WebClassII, though orthogonality refers to the group of features extracted from each category dictionary rather than to the single features.

The above formulation of both the naïve Bayes and the centroid-based classifiers assigns a document d to the most probable or the most similar class, independently of the absolute value of the posterior probability or similarity measure. However, we should expect that WebClassII users try to classify documents not related to any category in the hierarchy. In this case we expect a "reject" of the document. By assuming that documents to be rejected have either a low posterior probability for all categories or a low similarity to the centroids of all categories, the problem can be formulated as defining a threshold for the value taken by a naïve Bayes or centroid-based classifier [2].

5 The classification process

The classification of a new document is performed by means of a general graph search of the hierarchy of categories. The system starts from the root

and selects the nodes to be expanded such that the score returned by the classifier is higher than the threshold determined by the system. At the end of the process, all explored categories are considered for the final selection. The winner is the explored category with the highest score. It is noteworthy that the application of a classifier is always preceded by a change in the document representation according to the set of selected features. This corresponds to abstracting the same document at increasing levels of specificity.

6 Experimental results

In this section we study the performance of WebClassII on a set of Web documents. The data source used in this experimental study is Yahoo! ontology. We extracted 1026 actual Web documents referenced at the top two levels of the Web directory http://dir.yahoo.com/Science. There are 7 categories at the first level and 28 categories at the second level (see Fig. 1). A document assigned to the root of the hierarchy is considered "rejected" since its content is not related to any of the 35 subcategories. All the results are averages of five cross-validation folds. The size of the feature set ranges from 5 to 50 features per category. Features are extracted using hierarchical training sets. Collected statistics concern centroid-based and naïve Bayes classifiers trained according to one of the following three techniques:

1. *flat*, that is, by considering all subcategories together and neglecting their relations;
2. *hierarchical with proper training sets*, that is, by assigning only documents of category c to *Training(c)*;
3. *hierarchical with hierarchical training sets*, that is, by assigning documents of either category c or one of its subcategories to *Training(c)*.

To evaluate both flat and hierarchical classification techniques, we begin by considering the macro-weighted-average of both precision and recall measures [14]. They are computed as the normalized weighted sum of the corresponding microaverage measure. Weights are the percentages of documents per category, while the normalization factor is the number of categories.

Results reported in Fig. 2 (a) and (b) show that the flat technique always outperforms the hierarchical technique with respect to precision, while the naïve Bayes with hierarchical training sets has the best recall for increasing feature set size. Moreover, the centroid-based classifiers almost always have a lower accuracy and recall than the corresponding naïve Bayes classifiers, independently of the adopted technique (flat or hierarchical) and of the feature set size. Another interesting point is that the use of hierarchical training sets improves both performance measures independently of the method and the feature set size (see lines with suffix $\_SubCat$).

In fig. 2 (c) the percentage of "rejected" documents is reported. All training and test documents belong to a category of the hierarchy; therefore, it

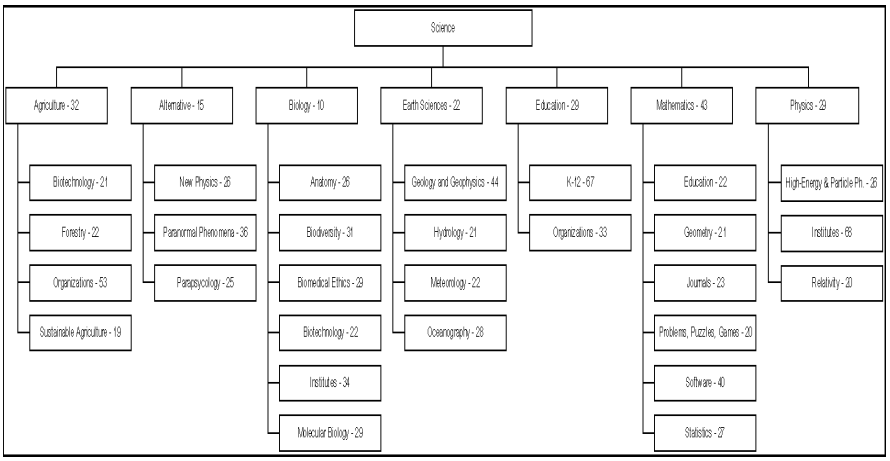


Fig. 1. A segment of Yahoo! ontology considered in this work. Numbers refer to the number of training documents for each category

would be desirable to have very low percentages of rejected documents, as in the case of naïve Bayes method with hierarchical training sets ($< 30\%$).

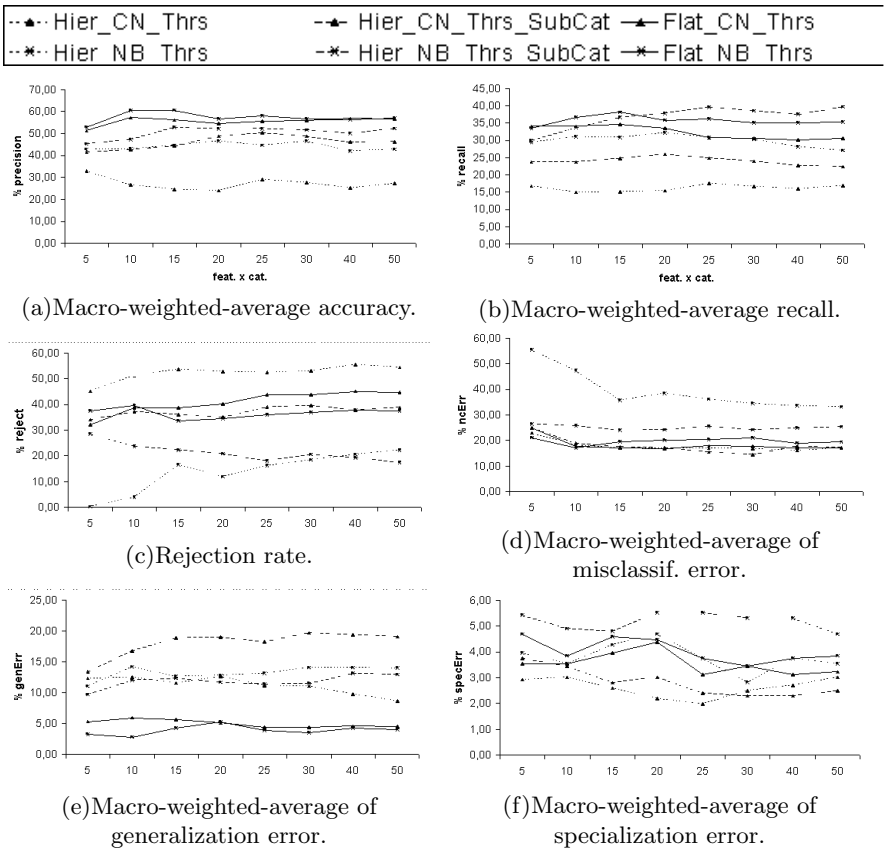
Intuitively, misclassifications of a document into categories similar to the correct one should be preferred to misclassifications into totally unrelated categories. Therefore, other three evaluation measures can be defined:

1. the *generalization error*, which computes the percentage of documents in c misclassified into a supercategory c';
2. the *specialization error*, which computes the percentage of documents in c misclassified into a subcategory c';
3. the *misclassification error*, which computes the percentage of documents in c misclassified into a category c' not related to c in the hierarchy.

The macro-weighted-average of the misclassification error is reported in Fig. 2 (d). Bayesian methods are the most prone to misclassification errors. However, in the case of hierarchical training sets, the increase of misclassification rate with respect to the centroid-based approach is about 7%, while the decrease of the rejection rate is above 23% with 50 features per category.

The graphs in fig. 2 (e) and (f) show the generalization and specialization errors. The flat approaches have the lowest generalization error, since they simply ignore the relations among categories. The centroid-based method with hierarchical training sets tend to overgeneralize. On the contrary, all methods have a low specialization error rate.

Finally, it is noteworthy that the centroid-based method is more computational efficient of the naïve Bayes only in the flat approach, while they have almost equal learning time in the hierarchical approaches. Also, the learning time of hierarchical techniques is lower than the flat technique, independently of the learning method.



(a)Macro-weighted-average accuracy.

(b)Macro-weighted-average recall.

(c)Rejection rate.

(d)Macro-weighted-average of misclassif. error.

(e)Macro-weighted-average of generalization error.

(f)Macro-weighted-average of specialization error.

Fig. 2. Results achieved

7 Conclusion

In this paper, the problem of automatically classifying HTML documents into a hierarchy of categories has been investigated in the context of a client-server application developed to support Web document sharing in a group of users with common interests. We studied the use of the hierarchy of categories both in the feature extraction and in the construction of the classifiers and in the classification process. As to feature extraction, a novel technique for the selection of relevant features from training pages has been presented. For the learning step, two classifiers have been considered and a thresholding algorithm has been proposed in the case of a reject class. For the classification, a generalized graph search technique has been considered. Experiments have been performed on a set of Web documents indexed in the Yahoo! ontology. Three techniques have been compared: i) flat classification; ii) hierarchical classification with proper training sets and iii) hierarchical classification with

hierarchical training sets. Results show that for hierarchical techniques it is better to use hierarchical training sets. Furthermore, the naïve Bayes classifier with hierarchical training sets has high overall recall and efficient learning process. Our results are comparable to those reported in [10], which is the only other work where Web documents indexed by Yahoo! ontology are actually used. However, in our study we have obtained an overall recall of 39.58% using only 875 features, while McCallum et al.'s method required about 13,000 features.

As future work, we intend to investigate a multistrategy approach, where different criteria and techniques are used at different levels of the ontology.

References

1. Almuallim H., Akiba Y., and Kaneda S.(1996) An efficient algorithm for finding optimal gain-ratio multiple-split tests on hierarchical attributes in decision tree learning. Proc. of the Nat. Conf. on Artificial Intelligence (AAAI'96), 703-708
2. Ceci M., Malerba D. (2003) Web-pages Classification into a Hierarchy of Categories, in Proceedings of the BCS-IRSG 25^{th} European Conference on Information Retrieval Research (ECIR '03)
3. C. Cleverdon (1984) Optimizing convenient online access to bibliographic databases. Information Services and Use, **4**, 37-47
4. D'Alessio S., Murray K., Schiaffino R., and Kershenbau A.(2000) The effect of using hierarchical classifiers in text categorization, Proc. of the 6th Int. Conf. on "Recherche d'Information Assiste par Ordinateur" (RIAO), 302-313
5. Dumais S. and Chen H.(2000) Hierarchical classification of Web document. Proc. of the 23^{rd} ACM Int. Conf. on Research and Development in Information Retrieval (SIGIR'00), 256-263
6. Esposito F., Malerba D., Di Pace L., and Leo P.(2000) A Machine Learning Approach to Web Mining, In E. Lamma and P. Mello (Eds.), AI*IA 99: Advances in Artificial Intelligence, Lecture Notes in Artificial Intelligence, **1792**, 190-201
7. Joachims T.(1997) A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. Proc. of the 14^{th} Int. Conf. on Machine Learning, 143-151
8. Koller D. and Sahami M.(1997) Hierarchically classifying documents using very few words. Proc. of the 14^{th} Int. Conf. on Machine Learning ICML'97, 170-178
9. Malerba D., Esposito F., and Ceci M.(2002) Mining HTML Pages to Support Document Sharing in a Cooperative System. In R. Unland, A. Chaudri, D. Chabane and W. Lindner (Eds.) XML-Based Data Management and Multimedia Engineering - EDBT 2002 Workshops, Lecture Notes in Computer Science, **2490**
10. McCallum A., Rosenfeld R., Mitchell T.M., Ng A.Y.(1998) Improving text classification by shrinkage in a hierarchy of classes. Proc. of the 15^{th} Int. Conf. on Machine Learning (ICML'98), 359-367
11. Mladenic D.(1998) Machine learning on non-homogeneus, distribuited text data, PhD Thesis, University of Ljubjana
12. Porter M. F.(1980) An algorithm for suffix stripping. Program, **14(3)**, 130-137
13. Salton G.(1989) Automatic text processing: The transformation, analysis, and retrieval of information by computer. Reading, MA: Addison-Wesley
14. Sebastiani F. (2002) Machine Learning in Automated Text Categorization. ACM Computing Surveys **34**, 1-47