Hierarchical Classification of HTML Documents with WebClassII

Michelangelo Ceci and Donato Malerba

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

Abstract. This paper describes a new method for the classification of a HTML document into a hierarchy of categories. The hierarchy of categories is involved in all phases of automated document classification, namely feature extraction, learning, and classification of a new document. The innovative aspects of this work are the feature selection process, the automated threshold determination for classification scores, and an experimental study on real-word Web documents that can be associated to any node in the hierarchy. Moreover, a new measure for the evaluation of system performances has been introduced in order to compare three different techniques (flat, hierarchical with proper training sets, hierarchical with hierarchical training sets). The method has been implemented in the context of a client-server application, named WebClassII. Results show that for hierarchical techniques it is better to use hierarchical training sets.

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 [2]. Therefore, to facilitate sharing of Web documents among distributed work groups in a large organization, it is important to develop automated document classification tools that assist users in the process of document classification with respect to a given set of document categories.

WebClass [8] 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.

F. Sebastiani (Ed.): ECIR 2003, LNCS 2633, pp. 57-72, 2003.

A simplifying assumption made in the design of WebClass is that document categories are not hierarchically related. This permits the system to build either one unique classifier for all categories (e.g., a decision tree that assigns a document to one of the pre-defined document categories) or to build a classifier for each category independently of the others (*flat classification*). However, in many practical situations categories are organized into a hierarchy like the one developed by Yahoo! (www.yahoo.com). This hierarchical organization 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 to the document classification techniques and we discuss some related work. 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: Issues and Related Work

In flat classification, a unique set of features is extracted from the training documents belonging to several distinct categories [5]. 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 each document. Each representation is useful for the classification of a document at one level of the hierarchy. In this way, general terms 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 either in the formulation of the learning algorithm or in the definition of the training sets. For instance, Almuallim et al. [1] defined a specific decision tree induction

algorithm for the case of hierarchical attributes. Training sets can be specialized for each internal node of the hierarchy by considering only documents of the sub-hierarchy 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, 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. In the case of hierarchical classification, the problem is partitioned into smaller subproblems, each of which can be effectively and efficiently managed.

Some of these aspects have been considered in related works. In particular, in the seminal work by Koller and Sahami [7] the hierarchy of categories is used in every processing step. For the feature extraction step a category dictionary is built for each node in the hierarchy. Feature extraction is based on an information theoretic criterion that eliminates both irrelevant and redundant features. For the learning step, two Bayesian classifiers are compared, namely the naïve Bayes and KDB [13]. A distinct classifier is built for each internal node (i.e., split) of the hierarchy. In the classification step, which proceeds top-down, it is used to decide to which subtree to send the new document. There is no possibility of recovering errors performed by the classifiers associated to the higher levels in the hierarchy. Two limitations of this study are the possibility of associating documents only to the leaves of the hierarchy and the effectiveness of the learning methods only for relatively small vocabularies (<100 features).

McCallum et al. [9] proposed a method based on the naïve Bayes. A unique feature set is defined for all documents by taking the union of all category vocabularies. Features for a given category are selected by means of the mutual information at each internal node of the tree, using the node's immediate children as classes. In the learning step, the hierarchy is considered in the shrinkage technique used to improve the estimate of some probabilities. For the classification step, the authors compare two techniques: exploring all possible paths in the hierarchy and greedily selecting the most probable one as done in [7]. Results show that greedy selection is more error prone but also more computational efficient. As in the previous work, all documents can be assigned only to the leaves of the hierarchy.

Mladenic [10] used the hierarchical structure to decompose a problem into a set of subproblems corresponding to categories (nodes in the hierarchy). For each subproblem, a naïve Bayes classifier is built from a set of positive examples, which is constructed from examples in the corresponding category node and all examples of its subtrees, and a set of negative examples corresponding to all remaining documents. The set of features selected for each category can be different. The classification applies to all the classifiers (nodes) in parallel, using some pruning of unpromising nodes. In particular, a document is passed down to a category only if the posterior probability for that category is higher than a user-defined threshold. Contrary to the previous work, document can be assigned to any node of the hierarchy.

In the work by D'Alessio et al. [3] documents are associated only to leaf categories of the hierarchy. Two sets of features are associated to each category, one is positive (features extracted from documents of the category) while the other is negative (features extracted from documents of sibling categories in the hierarchy). In addition

to contributing to feature extraction, the training set is also used to estimate feature weights and a set of thresholds, one for each category. Classification in a given category is based on a weighted sum of feature occurrences that should be greater than the category threshold. Both single and multiple classifications are possible for each testing document. The classification of a document proceeds top-down either through a single path (one-of-M classification) or through multiple-paths (binary classification). An innovative contribution of this work is the possibility of restructuring an initial hierarchy or building a new one from scratch.

Dumais and Chen [4] use the hierarchical structure for two purposes. First, to train several Support Vector Machines (SVM's), one for each intermediate node. The sets of positive and negative examples are constructed from documents of categories at the same level, and different feature sets are built, one for each category.[1] Second, to classify documents by combining scores from SVM's at different levels. Several combination rules are compared, some requiring a category threshold to be exceeded to pass a test document down to descendant categories. Multiple classification of a document is allowed for leaf categories, while the assignment of a document to intermediate categories is not considered. An empirical comparison based on a large heterogeneous collection of pages from LookSmart's web directory showed small advantages in accuracy for hierarchical models over flat models.

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 [10]. 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 [8].

The third difference 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 innovative contribution 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).

[1] Note that differently from [3], where each category is associated with two sets of features, positive and negative, here positive and negative examples are used to select a unique feature set for each category.

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, also called *bag-of-words*, no ordering of words or any structure of text is used. The feature set is unique for each category and 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. Only relevant tokens are used in the feature set. Before selecting relevant features, standard text pre-processing methods are used to:

1. Remove words with high frequency, or *stopwords*, such as articles and adverbs. Stopwords have been taken from Glimpse (glimpse.cs.arizona.edu), a tool used to index files by means of words.
2. Determine equivalent stems (*stemming*), such as 'topolog' in the words 'topology' and 'topological', by means of Porter's algorithm for English texts [11].

Many approaches have been proposed in the literature on information retrieval for the identification of relevant words to be used as index terms of documents [12]. Most of them simply score words according to some measure and select the best firsts. However, techniques proposed for information retrieval purposes are not always appropriate for the task of document classification. Indeed, we are not interested in words characterizing each single document, but we look for words that distinguish a document category from other categories. Generally speaking, the set of words required for classification purposes is much smaller than the set of words required for indexing purposes.

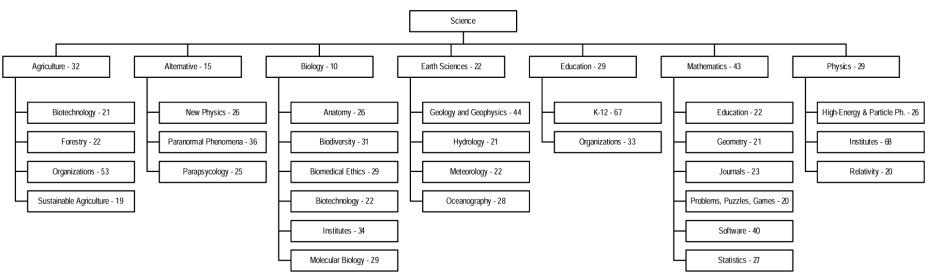

Fig. 1. A segment of Yahoo! ontology considered in this work. Numbers refer to the number of training documents selected for each category. Documents were downloaded on 12th March, 2002

For two-class problems, Mladenic [10] 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. [8] developed a feature selection procedure based on an extension of the well-known TF-IDF measure. They showed that this feature selection technique compares favorably with respect to average mutual information and odds-ratio. 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 all training documents d of category c',

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

- the *page frequency*, that is, the percentage of documents of category c' in which the feature w occurs,

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

where *Training(c')* can be either a proper training set or a hierarchical training set for documents of category c'. We remark that only documents considered as positive examples of c' are used to compute both $TF_{c'}(w)$ and $PF_{c'}(w)$.

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. Indeed, Web page authors usually "hide" a number of occurrences of "keywords" in the HTML code, in order to force search engines to rank that page in the first positions of the returned lists of Web references. By sorting each category dictionary according to the product $TF_{c'}(w) \times PF_{c'}(w)^2$, the effect of this phenomenon is kept under control.[2] Moreover, common words used in 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", "suggestion", "time" and "people") and should be considered as *quasi-stopwords*. In order to move quasi-stopwords down in the sorted dictionary, the value $TF_{c'}(w) \times PF_{c'}(w)^2$ 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'. We remark that the estimation of $ICF_c(w)$ also takes into account documents considered as negative examples of c'.

[2] 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.

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 $\forall i \in [1 \ldots k]$ w_i is a feature extracted from some document d in c', $v_i = TF(w_i) \times PF_{c'}^2(w_i) \times \frac{1}{CF_c(w_i)}$ 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' (see Fig. 2). 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 (like "math" and "mathemat") for upper level categories, and a set of specific features (like "topolog") for lower level categories.

Once the set of features has been determined, training documents can be represented as feature vectors, where each feature value is the frequency of a word.

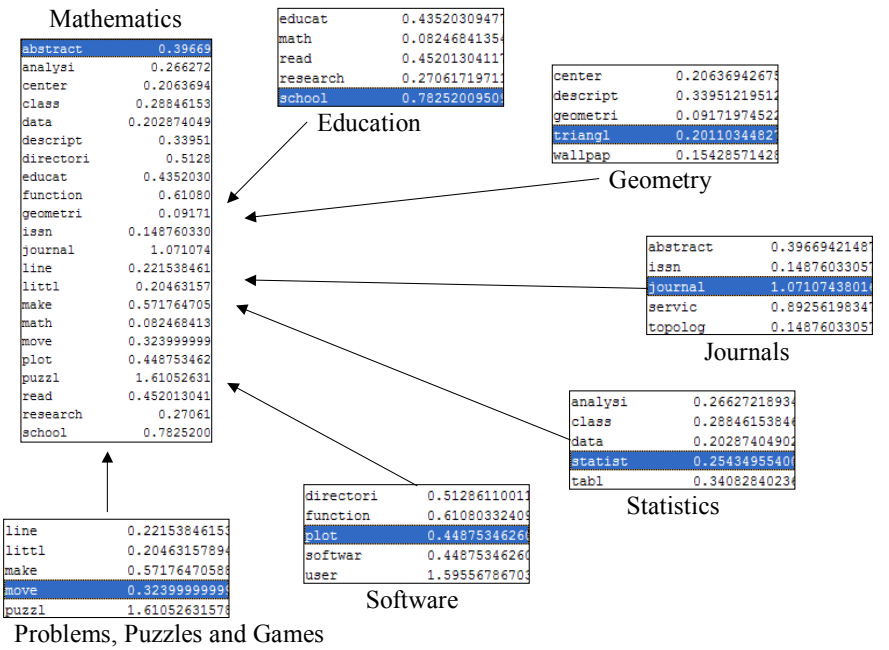

Fig. 2. Category dictionaries extracted by WebClassII for all subcategories of "Mathematics" (n_{dict}=5) and feature set selected for "Mathematics". Some features, like "geometri", are extracted from Web documents in German. Indeed, the collection of Web pages referenced by the Yahoo! Web directory "Science" also includes several non-English documents

4 The Learning Process

Currently, WebClassII has two alternative ways of assigning a Web page to a category:

1. By computing the similarity between the document and the *centroid* of that category.
2. By estimating the Bayesian posterior probability for that category (naïve Bayes).

Therefore, a training phase is necessary either to compute the centroids of the categories or to estimate the posterior probability distributions.

Let d be a document temporarily assigned to 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 [6]):

$$P_c(c_i \mid d) = \frac{P_c(c_i) \times \prod_{w \in FeatSet_c} P_c(w \mid c_i)^{TF(w,d)}}{\sum_{c' \in SubCategories(c)} P_c(c') \times \prod_{w \in FeatSet_c} P_c(w \mid c')^{TF(w,d)}} \quad (1)$$

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

$$P_c(c_i) = \frac{|Training(c_i)|}{\sum_{c' \in SubCategories(c)} |Training(c')|} = \frac{|Training(c_i)|}{|Training(c)|} \quad (2)$$

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

$$P_c(w \mid c_i) = \frac{1 + PF(w, c_i)}{|FeatSet_c| + \sum_{w' \in FeatSet_c} PF(w', c_i)} \quad (3)$$

In the above formulas, $TF(w,d)$ 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 likelihood $P_c(w \mid c_i)$ could be estimated according to the relative frequency, that is:

$$P_c(w \mid c_i) = \frac{PF(w, c_i)}{\sum_{w' \in FeatSet_c} PF(w', c_i)} \quad (4)$$

However, Laplace probability estimate is preferred because it is non-null even when $PF(w,c)=0$, that is when w does not occur in training documents of category c. Indeed, in Bayesian statistical inference, the assignment of a probability of zero to some parameter values of a probability distribution is considered a strong assumption, because it means it is impossible for the parameter to take on those values.

The *centroid* of a category is defined as an 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, the centroid most similar to the document description has to be found. The similarity measure considered is the *cosine correlation*, which computes the angle spanned by two vectors (the document and the centroid). The mathematical formulations of both the centroid and the similarity measure are the following:

$$P_c(w, c_i) = \frac{\sum_{d \in Training\ (c_i)} TF_d(w)}{|Training\ (c_i)|} \qquad Sim_c(c_i, d) = \frac{\sum_{w \in FeatSet_c} P_c(w, c_i) \times TF_d(w)}{\sqrt{\sum_{w \in FeatSet_c} P_c(w, c_i)^2 \times \sum_{w \in FeatSet_c} TF_d(w)^2}} \quad (5)$$

The cosine correlation returns a particularly meaningful value when vectors are highly dimensional and features define orthogonal directions. In WebClassII both conditions are satisfied, though orthogonality refers to the group of features extracted from each category dictionary rather than to the individual 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. Although this problem can be operatively dealt with by adding a "reject" category, where all such documents might be collected, from the viewpoint of document classification this is not always correct. This reject category represents "the rest of the world" and the computation of a posterior probability, as well as the computation of a centroid, would make little sense.

By assuming that documents to be rejected have either a low posterior probability for all categories or a small similarity to the centroids of all categories, the problem can be reformulated in a different way, namely, how to define a threshold for the value taken by a naïve Bayes or centroid-based classifier γ. In WebClassII the value of this threshold $Th_c(c_i)$ is algorithmically determined as follows. Let $\gamma_{c \to c'}(d) = P_c(c' \mid d)$ or $\gamma_{c \to c'}(d) = Sim_c(c', d)$ depending on the classifier. In other words, $\gamma_{c \to c'}(d)$ denotes the value returned by the classifier associated to the internal node *c* when the decision of classifying the document *d* in the subcategory *c'* is made. Then, we define the following threshold:

$$TruePosTh_c(c') = \min_{d \in Training(c')} \gamma_{c \to c'}(d)$$

Assumed that all training documents of c' have been (temporarily) assigned to *c* during the classification process, it is certainly desirable to set $Th_c(c_i)$ to a value lower than $TruePosTh_c(c')$, so that all training documents of c' can be correctly passed down from *c* to c'. Specifically, $TruePosTh_c(c')$ is the minimum threshold that allows all training documents of category c' (temporarily) assigned to *c* to be passed down to the category c'. On the other hand, it is desirable to set $Th_c(c_i)$ to a value greater than or equal to the two following thresholds:

$$FalsePosTh_c(c') = \max_{c'' \in SubCategories(c), c'' \neq c', d \in Training(c'')} \gamma_{c \to c'}(d)$$

$$ParentTh_c(c') = \max_{c' \in SubCategories(c), d \in Training(c), d \notin Training(c')} \gamma_{c \to c'}(d)$$

where the first is the maximum threshold that causes a misclassification of documents belonging to siblings of c', while the second is the maximum threshold that causes a misclassification of documents of the parent category c. In other words, $Th_c(c_i)$ should be set to a value lower than $TruePosTh_c(c')$ but not too low, since it might cause the misclassification in c' of training documents belonging either to siblings of c' or to c.

Obviously, if $\max(FalsePosTh_c(c'), ParentTh_c(c')) < TruePosTh_c(c')$ we can set $Th_c(c') = \max(FalsePosTh_c(c'), ParentTh_c(c'))$ since no misclassification is committed on the training examples of c'. This is the case in which the classifier separates well all examples of c' (and its descendants, in the case of hierarchical training sets) from those of its parent and sibling categories.

In general, the above inequality does not hold, in which case the threshold is determined empirically by maximizing the *Fscore*[3]. The procedure followed by WebClassII when the above inequality does not hold is reported in Fig. 3. Input data are the following:

1. $\gamma_c(c_i) = [\gamma_{c \to c_i}(d) | d \in Training(c_i)]$ that is, the list of values taken by the classifier for all documents of category c_i (or a subcategory, in the case of hierarchical training sets);

$$\gamma_c(\neg c_i) = \left[\gamma_{c \to c_i}(d) \,\middle|\, d \in \left(Training(c) \cup \bigcup_{c_j \in SubCategories(c)} Training(c_j) \right) - Training(c_i) \right]$$

   that is, the list of values taken by the classifier for each document in c or a subcategory of c different from c_i;

2. $V = \gamma_c(c_i) \cup \gamma_c(\neg c_i)$ sorted in ascending order;
3. β, that is, the parameter used in the computation of the *Fscore*.

The only output parameter of the procedure is $Th_c(c_i)$. It is determined by examining the sorted list V of classification scores and by selecting the middle point between two values in V such that the *Fscore* is maximized.

5 The Classification Process

The classification of a new document is performed by searching the hierarchy of categories. The system starts from the root and selects the nodes to be expanded such

[3] *Fscore* is a measure that synthesizes two important parameters used in information retrieval, namely *recall* and *precision*. It depends on a parameter β:

$$Fscore_\beta = \frac{(1+\beta^2) \cdot precision \cdot recall}{\beta^2 \cdot precision + recall}$$

In this work, we follow the common practice of setting β to 1, which gives equal importance to precision and recall.

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 (either internal or leaf) are considered for the final selection. The winner is the explored category with the highest score. If the document is assigned to the root, then it is considered rejected.

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. Since selected features are expected to be more specific for lower levels categories, the document is represented at decreasing levels of abstraction during the classification process. This automated representation change is highly desirable in hierarchical classification.

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.

The dataset is analyzed by means of a 5-fold cross-validation, that is, the dataset is first divided into five *folds* of near-equal size, and then, for every fold, WebClassII is trained on the remaining folds and tested on it. The system performance is evaluated by averaging some performance measures (see below) on the five cross-validation folds.

maxFscore := 0; $Th_c(c_i) := 0$;

$\forall k$:= 1, 2, …, |V|-1

begin

positive :=0; *negative* :=0; $middle := \frac{(V[k]+V[k+1])}{2}$;

$\forall v \in \gamma_c(c_i)$ if (*v*>*middle*) then *positive* := *positive*+1;

$\forall v \in \gamma_c(\neg c_i)$ if (*v*>*middle*) then *negative*:= *negative*+1;

$$precision(c_i) := \frac{positive}{positive + negative}; recall(c_i) := \frac{positive}{|\gamma_c(c_i)|};$$

$$Fscore_\beta := \frac{(1+\beta^2) \times precision(c_i) \times recall(c_i)}{\beta^2 \times precision(c_i) + recall(c_i)};$$

if ($Fscore_\beta > maxFscore$) then

begin maxFscore := $Fscore_\beta$; $Th_c(c_i)$= middle; end

end

Fig. 3. Automated threshold definition for a category c_i. Here $|V|$ and $|\gamma_c(c_i)|$ denote the cardinality of V and the number of training documents of category c_i, respectively

Several feature sets of different size have been extracted for each internal category in order to investigate the effect of this factor on the system performance. In particular, the feature set size ranges from 5 to 50 features per category. Features are extracted using hierarchical training sets. In principle, this guarantees that it is possible to associate a feature set also to those categories that have no proper training documents, although this is not the case of this experimentation.

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, this is, by assigning only documents of category *c* to *Training(c)*;
3. hierarchical with hierarchical training sets, this is, by assigning documents of either category *c* or one of its subcategories to *Training(c)*.

The baseline for the comparison is represented by the flat technique.

To evaluate both flat and hierarchical classification techniques, we begin considering the macro-weighted-average (MWA) of both precision and recall measures [14]. As shown in Fig. 3, the precision for a category *c*, denoted as *precision(c)*, measures the percentage of correct assignments among all the documents assigned to *c*, while the measure *recall(c)* gives the percentage of correct assignments in *c* among all the documents that should be assigned to *c*. For the whole category space, say $\{c_1, \ldots, c_m\}$, the MWA of precision and recall are defined as follows:

$$precision = \frac{\sum_{i=1}^{m} precision(c_i)w_i}{m} \qquad recall = \frac{\sum_{i=1}^{m} recall(c_i)w_i}{m}$$

where the weight w_i is the percentage of documents of category c_i.

Results reported in Fig. 4 and 5 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.

In fig. 6 the percentage of "rejected" documents is reported. All training and test documents belong to a category of the hierarchy; therefore, it would be desirable to have very low percentages of rejected documents. However, only the naïve Bayes method with hierarchical training sets have percentages below 30%. This means that the high thresholds automatically determined for the root classifier prevent most of test documents from passing down the hierarchy of categories during the classification process. For lower thresholds the rejection rate would decrease, but the system would be more prone to errors.

Intuitively, if a classification method misclassifies documents into categories similar to the correct categories, it is considered better than another method that misclassifies the documents into totally unrelated categories. Therefore, we define other three evaluation measures for a category c, namely:

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

For each category c, the sum of the recall, the generalization error, the specialization error and the misclassification error equals one.

The MWA of the misclassification error is reported in Fig. 7. Bayesian methods, which showed the lowest percentages of rejected documents, are also 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. 8 and 9 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. This is due to the "conservative" thresholds determined by the algorithm in Fig. 3. On the contrary, all methods have a low specialization error rate.

A finer analysis of the experimental results can be performed by considering some evaluation measures level by level in the hierarchy. The average precision and recall for the seven classes at the first level are reported in Fig. 10 and 11, while statistics concerning the categories at the second level are reported in Fig. 12 and 13. At both levels, the precision of the flat technique is generally the best. At first level, the recall is better for all the hierarchical techniques, while at the second level only the naïve Bayes with hierarchical training sets has a better recall than the flat technique. Interestingly, at the first level, the methods using proper training sets perform better (in precision and recall) than the methods using the hierarchical training sets, while this result is reversed at the second level. In other words, categories at lower levels take major advantages from hierarchical training sets.

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. In any case, the learning time of hierarchical techniques is lower than the flat technique, independently of the method.

7 Conclusions

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 graph search technique that explores all possible paths 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 on the Yahoo! documents show mainly that:

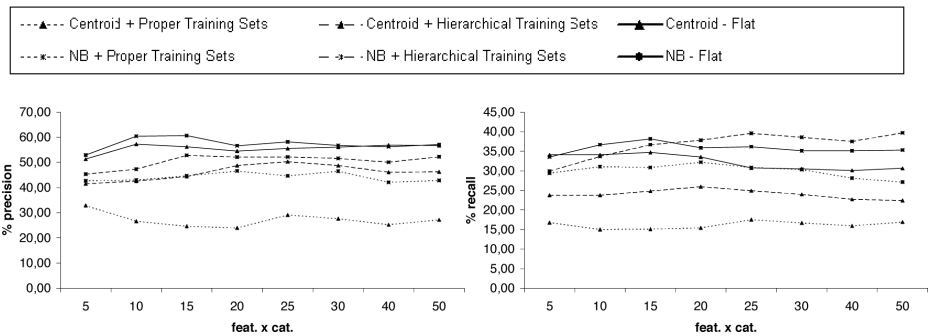

Fig. 4. MWA of accuracy

Fig. 5. MWA of recall

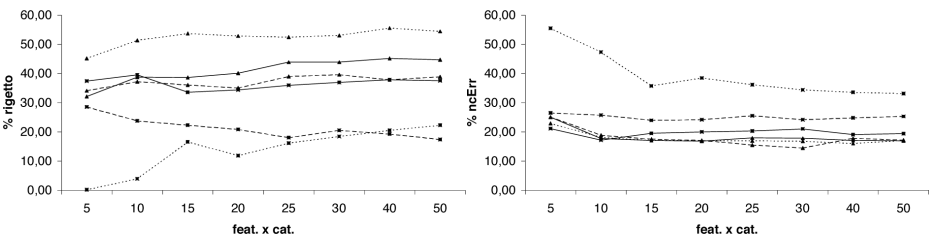

Fig. 6. Rejection rate

Fig. 7. MWA of misclassification error

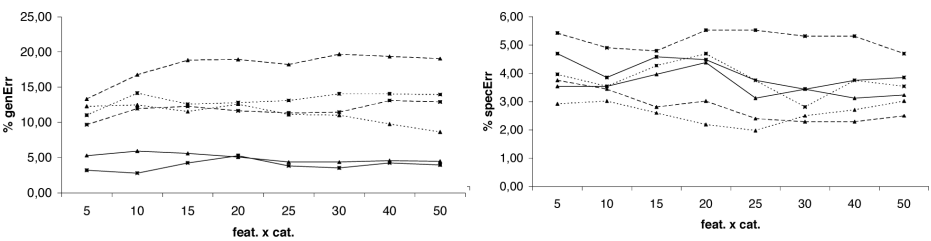

Fig. 8. MWA of generalization error

Fig. 9. MWA of specialization error

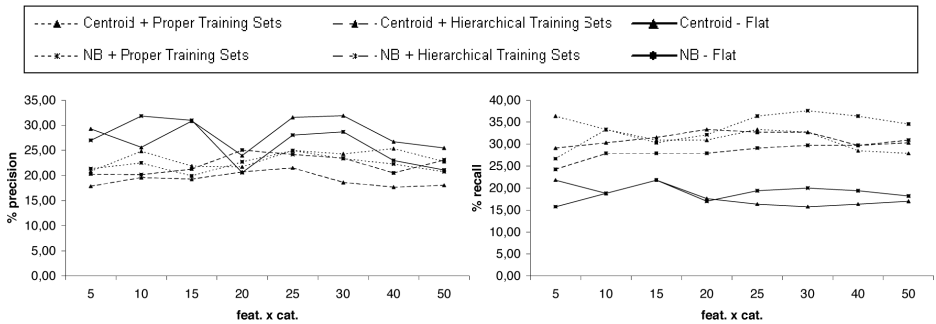

Fig. 10. MWA of precision for level 1

Fig. 11. MWA of recall for level 1

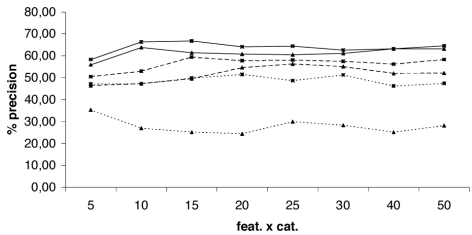

Fig. 12. MWA of precision for level 2

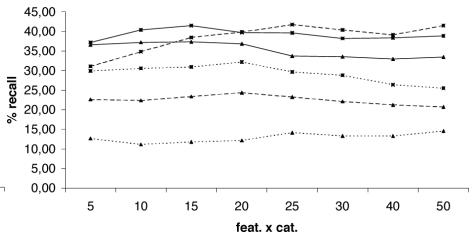

Fig. 13. MWA of recall for level 2

1. For increasing feature sets size, the best performances are observed for the naïve Bayes classifier trained according to either the flat approach or hierarchical training sets. Our results are comparable to those reported in [9], 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.
2. Bayesian methods show the lowest percentages of rejected documents but also the highest of misclassification errors. In any case, the relatively small increase of the misclassification rate is counterbalanced by the relatively large decrease of the rejection rate.
3. The lower the level of categories in the hierarchy, the greater the advantage of hierarchical training sets.

The level-by-level analysis performed in this work helped to better understand the different behaviour of the hierarchical and flat approaches. As future work we intend to investigate the relation between sibling categories in the performance evaluation as well as the application a multistrategy approach where different criteria and techniques are used at different levels of the hierarchy.

References

[1] Almuallim H., Akiba Y., & Kaneda S.: 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) (1996) 703-708

[2] Cleverdon C.: Optimizing convenient online access to bibliographic databases. Information Services and Use. 4 (1984) 37-47

[3] D'Alessio S., Murray K., Schiaffino R., & Kershenbau A.: The effect of using hierarchical classifiers in text categorization. Proc. of the 6th Int. Conf. on "Recherche d'Information Assistée par Ordinateur" (RIAO) (2000) 302-313

[4] Dumais S.& Chen H.: Hierarchical classification of Web document. Proc. of the 23rd ACM Int. Conf. on Research and Development in Information Retrieval (SIGIR'00) (2000) 256-263

[5] Esposito F., Malerba D., Di Pace L., & Leo P.: A Machine Learning Approach to Web Mining. In E. Lamma & P. Mello (Eds.). AI*IA 99: Advances in Artificial Intelligence, Lecture Notes in Artificial Intelligence, Vol. 1792, Berlin: Springer (2000) 190-201

[6] Joachims T.: A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. Proc. of the 14th Int. Conf. on Machine Learning (1997) 143-151

[7] Koller D. & Sahami M.: Hierarchically classifying documents using very few words. Proc. of the 14th Int. Conf. on Machine Learning ICML'97 (1997) 170-178

[8] Malerba D., Esposito F., & Ceci M.: Mining HTML Pages to Support Document Sharing in a Cooperative System. In R. Unland, A. Chaudri, D. Chabane & W. Lindner (Eds.): XML-Based Data Management and Multimedia Engineering - EDBT 2002 Workshops, Lecture Notes in Computer Science, Vol. 2490, Berlin:Springer (2002)

[9] McCallum A., Rosenfeld R., Mitchell T.M., Ng A.Y.: Improving text classification by shrinkage in a hierarchy of classes. Proc. of the 15th Int. Conf. on Machine Learning (ICML'98) (1998) 359-367

[10] Mladenic D.: Machine learning on non-homogeneus, distribuited text data, PhD Thesis, University of Ljubjana (1998)

[11] Porter M. F.: An algorithm for suffix stripping. Program, 14(3) (1980) 130-137

[12] Salton G.: Automatic text processing: The transformation, analysis, and retrieval of information by computer. Reading, MA: Addison-Wesley (1989)

[13] Sahami M.: Learning limited dependence Bayesian classifiers. Proc. of the 2nd Int. Conference on Knowledge Discovery in Databases (KDD'96) (1996) 335-338

[14] Sebastiani F.: Machine Learning in Automated Text Categorization. ACM Computing Surveys 34 (2002) 1-47