A New Thresholding Algorithm for Hierarchical Text Classification

Donato Malerba, Michelangelo Ceci, Michele Lapi, Giulio Altini

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

Abstract. This paper describes a method for the automatic classification of a HTML document into a hierarchy of categories. The method, implemented in the system WebClassIII extends and improves the previous version WebClassII in various directions. First, it presents a modified Naive Bayes algorithm in order to avoid problems related to the different document length. Second, it implements a probabilistic SVM-based classifier. Third, it implements a new thresholding algorithm that operates in a bottom-up fashion and takes advantage of the decision taken at lower levels of the hierarchy. Two experiments have been conducted. The first experiment aims to compare WebClassIII with WebClassII. The second experiment aims to show performances of WebClassIII with a deeper hierarchy.

1. Introduction

In recent years, the rapid growth of data made available on the web, has been demanding the use of machine learning techniques in order to extract useful and previously unknown information from data. In this context, particular importance has been given to document classification, which has been extensively used in filtering, retrieval and automatic categorization of web pages.

Typically, in document classification each document is tagged with a class or category label c_j, $j=1, 2, \ldots r$, and is represented as a feature vector $\langle X_1, X_2, \ldots, X_m \rangle$, where each feature represents a word and the value associated to the feature is computed on the basis of the frequency of the word in the document and the frequency of the word in the category (*tf-idf* representation) [16].

Given a training set of documents, the learning problem can be formulated in two different ways: either a *binary classifier* is induced for each category, or a *one-of-r classifier* is learned to determine whether a new document belongs to one of the *r* given categories [16].

More recently, increasing attention has been given to *hierarchical classification* [1] where, differently from classical approaches (called *flat classification approaches*), the pre-defined categories are organized in a hierarchical structure (tree-like structure). Such a structure reflects relations between concepts in the application domain covered by the classification.

While in flat classification a given document is assigned to a category on the basis of the output of one or a set of classifiers, in hierarchical classification, the assignment of a document to a category can be done on the basis of the output of multiple sets of classifiers. These classifiers are associated to different levels of the hierarchy to distribute documents among categories in a top-down way. The advantage of this hierarchical view of the classification process is that the problem is partitioned into smaller subproblems, each of which can be effectively and efficiently managed.

In the literature, several methods for hierarchical text classification have been proposed. They use a variety of statistical learning algorithms such as Centroid-Based [1], Naïve Bayes (NB) [8, 10, 11, 1], Support Vector Machines (SVM) [4], Neural Network (NN) [15, 17], perceptrons [12]. In addition to the learning algorithm they also differ for other two aspects. First, some of them are able to classify new documents only in the leaves of the hierarchy [8, 10, 3, 4, 12, 17] while others can classify new documents both in internal nodes and leaves [17, 11, 1]. Second, methods differ in terms of the feature selection strategy. For example, a separate feature set for each category is used in [8, 11, 4, 12, 15, 1], a unique feature set built from category vocabularies is adopted in [10], a positive and negative feature set per category is used in [3], while Weigend et al. [17] investigate the use of both separate and unique feature sets.

In a previous work by Ceci and Malerba [1], all issues met in the development of methods for hierarchical text classification were systematically investigated. First, documents can either be associated to the leaves of the hierarchy or to internal nodes. Second, the set of features selected for classification can either be category specific or the same for all categories (corpus-based). Third, the training set associated to each category c_j can be *proper* (only documents assigned to c_j) or *hierarchical* (documents assigned to c_j and its descendants). Fourth, the classifier may take into account or not the hierarchical relation between categories. Fifth, some stopping criteria (typically, a threshold on the classification score) are required for hierarchical classification of new documents in non-leaf categories. Sixth, new performance evaluation criteria are required to take into account the different types of classification errors. The method implemented in the system WebClass II is characterized by the fact that the hierarchy of categories is involved in *all* phases of automated document classification, namely feature extraction, learning, and classification of a new document.

This paper follows and extends this initial work on hierarchical text classification along several directions. The first extension concerns the feature selection phase, which considers the dictionaries of *all* subcategories of a class c_j and not only those of its children. The second extension concerns the learning algorithms. In particular, a modified Naive Bayes algorithm is considered in order to avoid problems related to the different document length [7]. Moreover, a probabilistic SVM-based classifier is also considered in addition to centroid-based and naïve Bayes classifiers. The third extension concerns the document classification process. The original top-down strategy implemented in the system *WebClassII* [1] to determine the thresholds of the classifiers has been replaced by a new automated threshold selection algorithm that operates according to a bottom-up strategy, thus taking full advantage of the decisions made at lower levels of the hierarchy. New experimental results are reported, which show better performances of the extended system WebClass III for the classification of Web pages according to a hierarchy of categories.

The paper is organized as follows. In the next section, we illustrate the learning strategies available in *WebClassIII.* In Section 3, the new automated threshold determination algorithm is presented. Experimental results are reported and commented in Section 4. Finally, in Section 5 some conclusions are drawn.

2 Learning

The document representation adopted in WebClass III is based on the standard *bag-of-words* approach, that is, a document is represented by a vector of features, each of which corresponds to a word[1] occurring in at least a document of the training set.[2] Differently from other systems, in WebClass III a document can be represented by several feature vectors, one for each classifier used in the hierarchical classification procedure. In this way, the most appropriate representation can be chosen for the same document at different levels in the hierarchy, with more emphasis on general terms at higher levels while very specific terms are used at lower levels. The selection of feature sets can be formalized as follows. Let *c* be a category and *c'* one of its children in the hierarchy of categories, that is, $c' \in DirectSubCategories(c)$. Let *d* be a training document (after the tokenizing, filtering and stemming steps) from *c'*, *w* a feature extracted from *d* and $TF_d(w)$ the relative frequency of *w* in *d*. Then, the following statistics can be computed:

- the maximum value of $TF_d(w)$ (term frequency of a word *w* in a document *d*) 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, $PF_{c'}(w) = \frac{occurrences_{c'}(w)}{|Training(c')|}$

  where *Training(c')* is a hierarchical training set for documents of category *c'*.
- the *category frequency* $CF_c(w)$, that is, the number of categories $c'' \in DirectSubCategories(c)$ in which the word *w* occurs.

It is noteworthy that only documents considered as positive examples of *c'* are used to compute both $TF_{c'}(w)$, and $PF_{c'}(w)$ and $CF_c(w)$.

For each category *c'*, a category dictionary $Dict_{c'} = [(w_1, v_1), (w_2, v_2), \ldots, (w_k, v_k)]$ is computed, such that $\forall i \in [1 \ldots k]$ w_i is a feature extracted from some document *d* in the hierarchy rooted in *c'*, $v_i = TF(w_i) \times PF^2_{c'}(w_i) \times \frac{1}{CF_c(w_i)}$ and $k \leq n_{dict}$ (n_{dict} is the maximum number of features selected for a category dictionary). By taking words that maximize the product TF-PF^2-ICF, where ICF stands for "inverse CF", we reward common words used in documents of category *c'*, but we penalize words

[1] Actually, original words are normalized to an approximation of the morphological root, called *stem*, according to Porter's stemming algorithm [14].

[2] Words occurring in a stopword list taken from Glimpse (glimpse.cs.arizona.edu) are discarded.

common to both *c′* and its sibling categories. For this measure, good results were reported in the case flat (i.e., non-hierarchical) classification [9].

The feature set $FeatSet_c$ associated to a category *c* is defined on the basis of the category dictionaries $Dict_{c'}$ for all its subcategories *c′*. More precisely, in WebClassII the *level feature set* $FeatSet_c$ is defined as the union of the dictionaries of all children of *c*:

$$FeatSet_c = \bigcup_{c' \in DirectSubCategories(c)} Dict_{c'}$$

while in WebclassIII, the *hierarchical feature set* is defined as the union of the dictionaries of all subcategories (similarly to [11] where, in addition, weights are used to give less importance to subcategories that are further down in the Hierarchy):

$$HierFeatSet_c = \bigcup_{c' \in SubCategories(c)} Dict_{c'}$$

The rationale for this choice is based on our previous observation that the flat approach always outperformes the hierarchical approach [1]. While the differences in precision are small for categories at higher levels, they become bigger for categories at leaves, and they are quite evident for deep hierarchies. This phenomenon can be explained by the fact that when high level classifiers do take into account only general terms (such as “math” and “mathemat”) typically extracted from documents of high level categories (e.g., Mathematics), they do have some difficulties to correctly route along the right path those documents belonging to leaf categories (e.g., Lattices), because of the rarer occurrence of general terms. As we will show in the experimental results, the usage of a hierarchical feature set prevents this problem from upraising.

WebClassII embeds 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).

WebClass III also includes another way opportunity:

3. By estimating the posterior probability for that category according to an SVM probabilistic classifier.

Moreover, the original naïve Bayes method has been modified in order to correctly handle documents of different length. For the sake of completeness, a brief description of the learning methods is reported in the next subsections.

2.1 Naive Bayes Classifier

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 $c_i \in DirectSubCategories(c)$ 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]):

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

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

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

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

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

which is non-null even when $PF(w,c)=0$, that is when w does not occur in training documents of category c. 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.

Despite of the fact that this formalization has been used in WebClassII and in most of naive Bayes text classifiers proposed in the literature, it presents problems when one wants to interpret the score for each class as an estimate of uncertainty. If for some word w, the value of $P_c(w|c_i)$ differs by one order of magnitude between different classes c_i, then the final probabilities will differ by as many orders of magnitude as there are words in the document. As a consequence, scores for the winning class tend to be close to 1.0 while scores for the losing classes tend toward 0.0. These extreme values are an artefact of the independence assumption. Class-conditional word probabilities would be much more similar across classes if word dependencies were taken into account [2]. This problem is particularly damaging in WebClassII/III where scores are used in the construction of the hierarchical classifier (see Section 4). An additional problem in the above formalization is strictly related to the probability estimation in formula 1, which regards all documents belonging to c_i as one huge document. In other words, this estimation method does not take into account the fact that there may be important differences among term occurrences from documents with different lengths [7] and estimation could be affected by significant length discrepancy among documents belonging to the same class [16].

In this paper we adopt a normalization of the value $TF(w,d)$ in formula 1 in order to avoid these problems. In particular, we normalize TF according to the following formula:

$$NormalizedTF(w,d) = \frac{TF(w,d)}{\|TF(\bullet,d)\|_2} \quad \textbf{(5)}$$

where $\|TF(\bullet,d)\|_2 = \sqrt{\sum_{w' \text{ in } d} TF(w',d)^2}$. By substituting $TF(w,d)$ with $NormalizedTF_c(w,d)$ in (1), we have:

$$P_c(c_i \mid d) = \frac{P_c(c_i) \cdot \prod_{w \in FeatSet_c} P_c(w \mid c_i)^{NormalizedTF(w,d)}}{\sum_{c' \in DirectSubCategories(c)} P_c(c') \cdot \prod_{w \in FeatSet_c} P_c(w \mid c')^{NormalizedTF(w,d)}} \quad \textbf{(6)}$$

2.2 Centroid-Based Classifier

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, 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:

$$P_c(w, c_i) = \frac{\sum_{d \in Training(c_i)} TF_d(w)}{|Training(c_i)|} \qquad Sim_c(c_i, d) = \frac{\sum_{w \in FeatSet_c} P_c(w, c_i) \times TF_d(w)}{\sqrt{\sum_{w \in FeatSet_c} P_c(w, c_i)^2 \times \sum_{w \in FeatSet_c} TF_d(w)^2}} \quad \textbf{(7)}$$

The cosine correlation returns a particularly meaningful value when vectors are highly dimensional and features define orthogonal directions. In WebClassIII both conditions are satisfied, though orthogonality refers to the group of features extracted from each category dictionary rather than to the individual features.

2.3 SVM-probabilistic Classifier

SVM are based on the Structural Risk Minimization (SRM) principle. The idea of SRM is to find a hypothesis that reflects the well-known trade-off between the training error and the complexity of the space. A SVM learns from the training set to find a decision surface (hyperplane) in the vector space of documents that 'best' separates the data points (i.e. documents) into *two* classes.

The SVM embedded in WebClassIII is a modified version of Platt's Sequential Minimal Optimization classifier (SMO) [13]. Modification is necessary, since the classifier learned for each internal node of the hierarchy is of the kind one-of-*r* (multi-class classification). More precisely, a binary classifier is learned for each couple of classes and afterwards, the probability $P_c(c_i|d)$ is computed by means of a probabilistic pair-wise coupling classification [6]. Once again, the decision taken by the classifier for each training document is associated with a (probabilistic) score, which is processed by the automated thresholding algorithm as explained in Section 3.

In the proposed method, documents are represented by means of the classical *tfidf* vector representation [16].

3 Computation of thresholds

In WebClassIII, a classifier (either centroid-based, or naive Bayesian or SVM-based) is learned for each internal category *c* of the hierarchy. This classifier is used to decide, during the classification of a new document, which class $c_i \in DirectSubCategories(c)$ is the most appropriate to receive the document. In general, however, a document should not be necessarily passed down to a subcategory of *c*. This makes sense in the case that:

a) the document to be classified deals with a general topic rather than a specific topic, or
b) the document to be classified belongs to a specific category that is not present in the hierarchy and makes more sense to classify the document in the "general category" rather than in a wrong category.

To support the classification of documents also in the internal categories of the hierarchy, WebClassIII (and its predecessor WebClass II) computes the thresholds that represent the "minimal belief" (returned by the classifier) such that a document can be considered belonging to a direct subcategory. More formally, let $\gamma_{c\to c'}(d) = P_c(c' \mid d)$ in the case of naive Bayes Classifier and SVM probabilistic classifier, and $\gamma_{c\to c'}(d) = Sim_c(c', d)$ in the case of the centroid-based classifier. $\gamma_{c\to c'}(d)$ denotes the score returned by the classifier associated to the internal node *c* when the decision of classifying the document *d* in the subcategory *c*' is made. Thresholds are used to decide if a new testing document is characterized by a score that justifies the assignment of such a document to c'. Formally, a new document *d,* temporary assigned to a category *c,* will be passed down to a category *c'* if $\gamma_{c\to c'}(d) > Th_c(c')$, where $Th_c(c')$ represents "minimal belief" such that a document assigned to *c* can be considered belonging to *c'*.

In WebClassII, thresholds are determined by means of an algorithm that computes scores top-down by maximizing the *FScore* of the hierarchical classification on training documents. Although this approach gives promising results, it presents three limitations:

- it is conservative in the sense that it tends to classify documents in higher categories [1]
- thresholds are computed according to a top-down strategy, so that when a threshold is defined, it is not possible to take into account the possibly wrong decisions taken by classifiers at lower levels of the hierarchy.
- the distance between "target" and "assigned" categories in the hierarchy is not considered when a misclassification error occurs.

In WebClassIII, a different thresholding algorithm is implemented. It is based on a bottom-up strategy and tries to minimize a measure that is based on a *tree distance*.

Before describing the algorithm and the used measure, some useful notations are introduced:

1. $\gamma_c(c_i) = \lfloor \gamma_{c\to c_i}(d) \mid d \in Training(c_i) \rfloor$ that is, the list of values taken by the classifier for all documents of category c_i (or a subcategory).

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

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

3. $V = \gamma_c(c_i) \cup \gamma_c(\neg c_i)$ sorted in ascending order;

The algorithm (see Figure 1) takes in input the categories *c* and *c'* where *c'* is a child of *c* and returns the threshold associated to *c'* ($Th_c(c')$). $Th_c(c')$ 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 expected error is minimized. This error is estimated on the basis of the distance between two nodes in a tree-structure:

Definition (*tree distance*)
Let *Categories* be the set of all the categories, the tree distance $\delta_{Hierarchy}$ is a function $\delta_{Hierarchy} : Categories^2 \to \Re$ that associates two categories $c_1,c_2 \in Hierarchy$ with a real value such that the following conditions are fulfilled:

I. $\forall c_1,c_2 \in Categories \quad 0=\delta_{Hierarchy}(c_1,c_1) \leq \delta_{Hierarchy}(c_1,c_2) = \delta_{Hierarchy}(c_2,c_1)$
II. $\forall c_1,c_2 \in Categories$: $\delta_{Hierarchy}(c_1,c_2)=0 \Rightarrow c_1= c_2$
III. $\forall c_1,c_2,c_3,c_4 \in Categories : \delta_{Hierarchy}(c_1,c_2) + \delta_{Hierarchy}(c_3,c_4) \leq$
max $\{\delta_{Hierarchy}(c_1,c_3)+ \delta_{Hierarchy}(c_2,c_4), \delta_{Hierarchy}(c_1,c_4) + \delta_{Hierarchy}(c_2,c_3)\}$ ■

In a tree distance, the dissimilarity between two categories is reproduced as the sum of the weights of all edges of the (unique) path connecting the two categories in the hierarchy [5]. When a unit weight is associated to each edge (as in WebClassIII) the dissimilarity is the length of the path. Intuitively, the automated thresholding algorithm tries to compute thresholds by minimizing the distance between the true class of a document and the class returned by the hierarchical classifier.

4 Experimental results

In this section we will discuss experimental results obtained by running WebClassIII on two different collections of Web documents. Before proceeding with the analysis of the obtained results, we introduce the measures used to evaluate the classification performance. We begin considering the micro-weighted-average (µWA) of both precision and recall measures. 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* [16]. In order to obtain a combined measure of precision and recall, we use F-Score (with $\beta = 1$) [16].

Intuitively, if a classification method misclassifies a document into a category "close to" the correct one, this is considered better than another method that misclassifies the documents into a totally unrelated category. Therefore, we define other three evaluation measures for a category *c*, namely *misclassification error, generalization error* and *specialization error* [1]. In addition, the *unknown ratio*, that

```
find_thresholds(c,c',thresholdSet) $\rightarrow$ thresholdSet'  {
    if not leaf(c') then
        $\forall$ c'' $\in$ SubClasses(c')
            thresholdSet$\leftarrow$ find_thresholds(c',c'',thresholdSet);
    compute_and _sort(V,c,c') ;
    $Th_c$(c') $\leftarrow$0;
    bestError $\leftarrow \infty$;
    $\forall$ k = 0,..., |V|  {
        if  k=0 then middle $\leftarrow$ V[1]-ε;                                // ε>0
            elseif  k=|V| then  middle $\leftarrow$ V[k];
                else  middle $\leftarrow$ (V[k]+V[k+1])/2;
        error $\leftarrow$ 0;
        $\forall$ v $\in \gamma_c(c')$ {
          let $d \in$ Training(c') s.t. v=$\gamma_{c\rightarrow c'}(d)$
          if v>middle then
              error += $\delta_{Hierarchy(c')}(class(d), classify(d, thresholdSet, Hierarchy(c')))$
            else
              error+=$\delta_{Hierarchy(c')}(class(d), c)$ ;
        }
        $\forall$ v $\in \gamma_c(\neg c')$ {
        let $d \in \left(Training(c) \cup \bigcup_{c_j \in DirectSubCategories(c)} Training(c_j)\right) - Training(c_i)$ s.t. v=$\gamma_{c\rightarrow c'}(d)$
        if v>middle then
            error += $\delta_{Hierarchy(c)}(class(d), classify(d, thresholdSet, Hierarchy(c')))$;
          else
            error+=0;
        }
         if error<bestError then
          $Th_c$(c') $\leftarrow$ middle;
    }
      return thresholdSet$\cup${<c', $Th_c$(c') >};
}
```

Fig. 1. Automated threshold definition for a category c_i.

measures the percentage of rejected documents belonging to a category c among all documents belonging to c has been computed.

All classifiers are trained according to the following techniques:

1. flat, that is, by considering all subcategories together and neglecting their relations;
2. hierarchical with level feature sets;
3. hierarchical with hierarchical feature sets.

Datasets have been analysed 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, the system is trained on the remaining folds and tested on it. The system performance is evaluated by averaging the performance measures discussed above on the five cross-validation folds.

4.1 WebClassII vs WebClassIII: two levels Science

The first experiment aims to compare the performances of WebClassII and WebClassIII on the basis of the measures defined in the previous section. This evaluation follows the experiments conducted in [1] on a set of Web documents extracted by a subset of the Yahoo! science ontology http://dir.yahoo.com/Science. Details are available in [1]. Experiments are conducted by varying the size of the feature set from 5 to 50.

In Figure 2, results concerning the centroid-based classifier are reported. It is noteworthy that, although in terms of precision WebClassIII does not significantly improve the performances of WebClassII, this is not true in the case of Recall, where WebClassIII with level Feature Set reaches comparably good results with respect to WebClass II/III – flat approach. The abundance of features in the hierarchical feature sets for high-evel categories negatively affects the performance of the classifier. As expected, WebClassIII drastically reduces the generalization error. In Figure 3 results about Naïve-Bayes classifier are reported. Results show that WebClassIII outperforms WebClassII in terms of precision, but not in terms of recall. Anyway, the *FScore* shows that WebClassIII is in general beneficial (also considering generalization, specialization and misclassification errors).

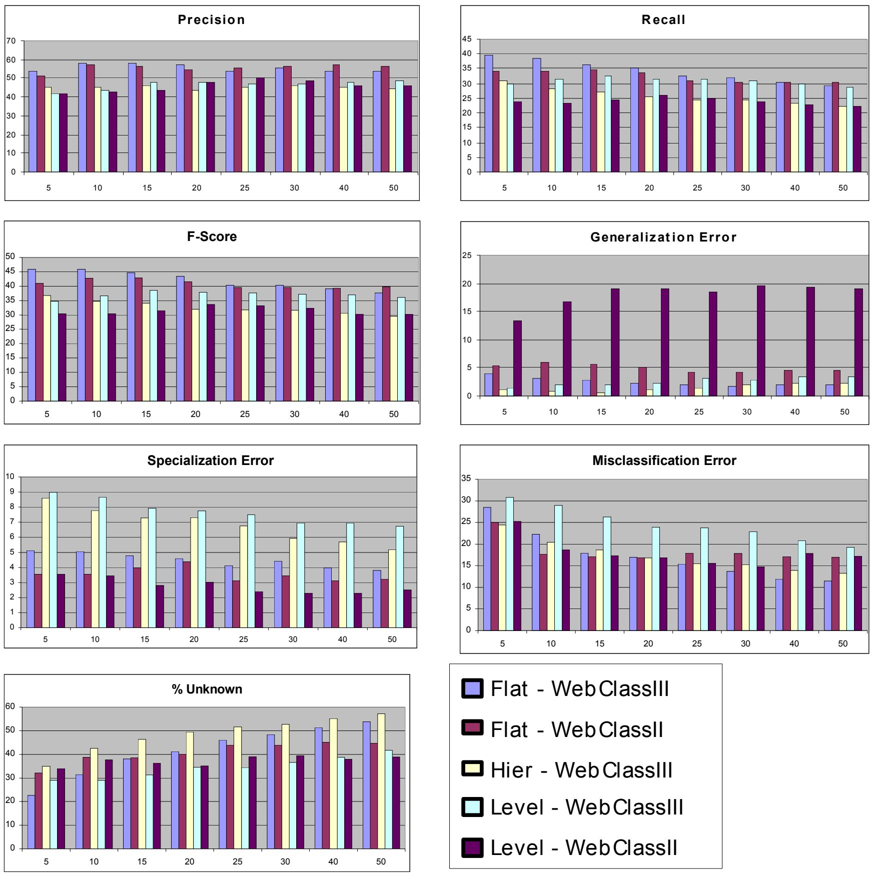

Fig. 2. WebClassII vs WebClassIII: centroid-based algorithm

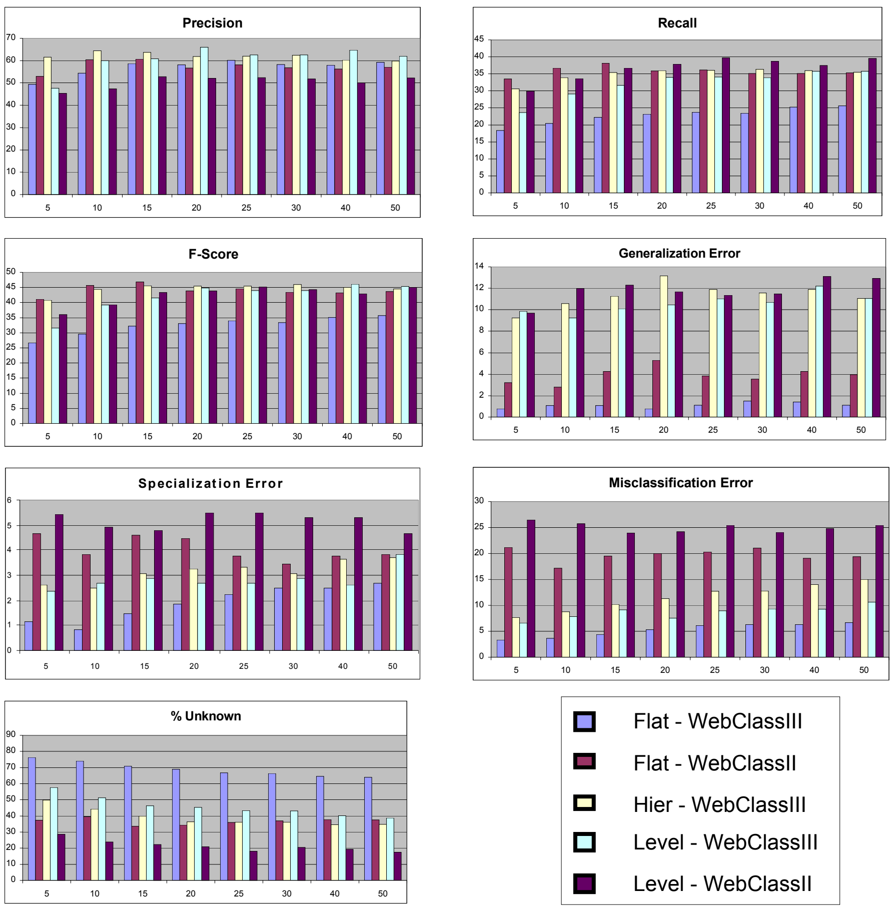

Fig. 3. WebClassII vs WebClassIII: Naive Bayes

4.2 WebClassIII: Three levels Science (15th July 2003 Science-Yahoo dataset)

In this experiment we analyze the performance of WebClassIII on a different set of Web documents, using centroid-based, Support Vector Machines and naïve Bayes classifiers. Once again, the data source used in this experimental study is Yahoo! science ontology (Documents have been extracted on the 15th July 2003). The main difference w.r.t. the previous dataset is that this is organized in a three-levels hierarchy. We extracted 907 actual Web documents referenced at the top three levels of the Web directory http://dir.yahoo.com/Science. There are 6 categories at the first level, 27 categories at the second level and 35 categories at the third level. A document assigned to the root of the hierarchy is considered "rejected" since its content is not related to any of the 68 subcategories.

Several feature sets (both level and hierarchical) of different size have been extracted for each internal category in order to investigate the effect of this factor on

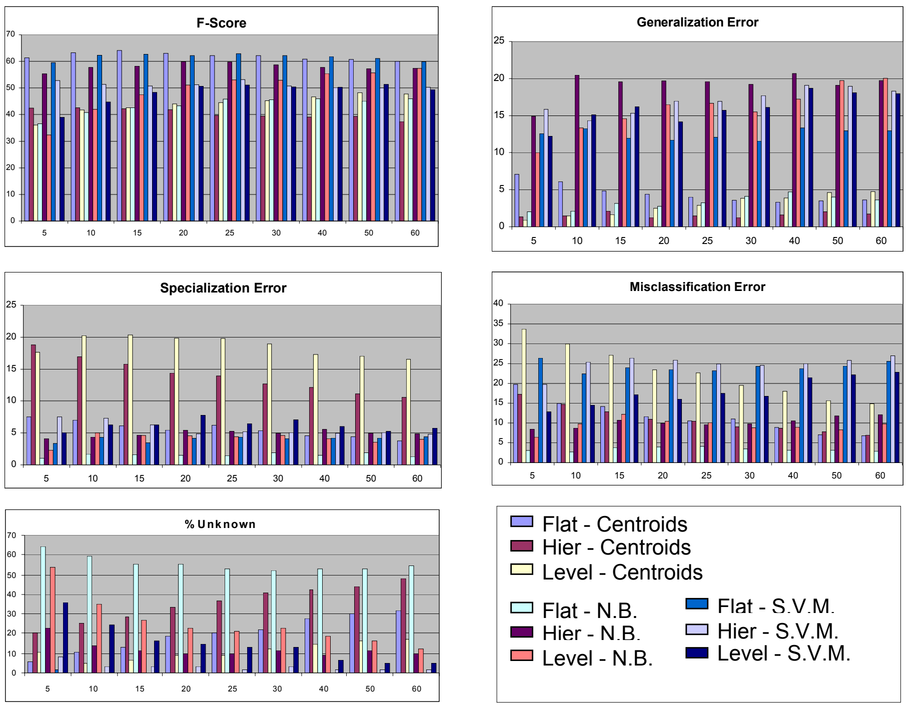

Fig. 4. WebClassIII results on 15th July 2003 Science-Yahoo dataset

the system performance. In particular, the feature set size ranges from 5 to 60 features per category and dictionaries are extracted using hierarchical training sets.

The baseline for the comparison is represented by the flat technique. Results (Figure 4) show that, in general, for what concerns *FScore* on hierarchical classification, Naive Bayes outperforms SVM. This result is a bit surprising and would be interesting to deeply investigate this aspect using different datasets. The worst performance is represented by Centroids. Results also show that hierarchical classification outperforms flat classification only in the case of Naive Bayes. This is probably due to the limited number of categories. Finally, the use of the Hierarchical Feature Set is generally beneficial especially with a low number of features.

5 Conclusions

In this paper, we presented some further extensions of a previous study on hierarchical text categorization. To investigate the effectiveness of a new automated thresholding algorithm, the performance of the an SVM-based classifier, the effect of document length normalization for the naïve Bayes classifier, and the efficacy of a hierarchical feature set approach, a new release of the WebClass system has been produced. Results show that this new release performs generally better than the previous one and that the use of the Hierarchical Feature Set is generally beneficial.

For future work, we plan to extend experiments to other datasets, including Reuters Corpus Volume1 in order to deeply investigate the behaviour of the proposed approach.

References

1. Ceci M., Malerba D. Web-pages Classification into a Hierarchy of Categories, in *Proceedings of the BCS-IRSG 25th European Conference on Information Retrieval Research* (ECIR ‘03), Pisa, Italy - April 14-16, 2003.
2. M Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, S. Slattery: Learning to construct knowledge bases from the World Wide Web. *Artificial Intelligence*, vol. 118, pp. 69 – 113. (2000).
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)* (2000) 256-263
5. F. Esposito, D. Malerba, V. Tamma, & H.-H. Bock (2000). Classical resemblance measures. Chapter 8.1 in in H.-H. Bock and E. Diday (Eds.), Analysis of Symbolic Data. Exploratory methods for extracting statistical information from complex data, Series: Studies in Classification, Data Analysis, and Knowledge Organization, vol. 15, Springer-Verlag:Berlin, 139-152
6. T. Hastie, R. Tibshirani: Classification by Pairwise Coupling in *Proceedings of the 1997 conference on Advances in neural information processing systems* (1998) 507-513
7. S.B. Kim, H.C. Rim, D. Yook, H. Lim: Effective Methods for Improving Naive Bayes Text Classifier. *In 7th International Conference on Artificial Intelligence*. Lecture Notes in Computer Science, vol. 2417. (2002).
8. 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
9. 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)
10. 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
11. Mladenic D.: Machine learning on non-homogeneus, distribuited text data, PhD Thesis, University of Ljubjana (1998)
12. Ng H.T., Goh W.B., Low K.L.: Feature Selection, Perception Learning, and a Usability Case Study for Text Categorization. Proc. Of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (1997) 67-73
13. J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advances in kernel methods - support vector learning*. MIT Press, (1998).
14. Porter M. F.: An algorithm for suffix stripping. Program, 14(3) (1980) 130-137.
15. Ruiz M.E., Srinivasan P.: Hierarchical Text Categorization Using Neural Networks. Information Retrieval Vol. 5 Num. 1 (2002) 87-118
16. Sebastiani F.: Machine Learning in Automated Text Categorization. ACM Computing Surveys 34 (2002) 1-47
17. Weigend A.S., Wiener E.D. and Pedersen J.O.: Exploiting Hierarchy in Text Categorization. Information Retrieval Vol. 1 Num. 3 (1999) 193-216