J Intell Inf Syst (2007) 28:37–78
DOI 10.1007/s10844-006-0003-2

# Classifying web documents in a hierarchy of categories: a comprehensive study

Michelangelo Ceci · Donato Malerba

Received: 11 July 2005 / Revised: 7 December 2005 /
Accepted: 3 April 2006 / Published online: 19 January 2007

**Abstract** Most of the research on text categorization has focused on classifying text documents into a set of categories with no structural relationships among them (flat classification). However, in many information repositories documents are organized in a hierarchy of categories to support a thematic search by browsing topics of interests. The consideration of the hierarchical relationship among categories opens several additional issues in the development of methods for automated document classification. Questions concern the representation of documents, the learning process, the classification process and the evaluation criteria of experimental results. They are systematically investigated in this paper, whose main contribution is a general hierarchical text categorization framework where the hierarchy of categories is involved in all phases of automated document classification, namely feature selection, learning and classification of a new document. An automated threshold determination method for classification scores is embedded in the proposed framework. It can be applied to any classifier that returns a degree of membership of a document to a category. In this work three learning methods are considered for the construction of document classifiers, namely centroid-based, naïve Bayes and SVM. The proposed framework has been implemented in the system WebClassIII and has been tested on three datasets (Yahoo, DMOZ, RCV1) which present a variety of situations in terms of hierarchical structure. Experimental results are reported and several conclusions are drawn on the comparison of the flat vs. the hierarchical approach as well as on the comparison of different hierarchical classifiers. The paper concludes with a review of related work and a discussion of previous findings vs. our findings.

M. Ceci (✉) · D. Malerba
Dipartimento di Informatica, Universita degli Studi di Bari,
70126 Bari, Italy
e-mail: ceci@di.uniba.it

D. Malerba
e-mail: malerba@di.uniba.it

**Keywords** Text categorization · Hierarchical models · Supervised learning · Feature selection · Performance evaluation · Web content mining

## 1 Introduction

Text classification or text categorization is the process of automatically assigning one or more predefined categories to text documents. A wide range of supervised learning algorithms has been applied to this problem, using a training set of categorized documents to build a classifier that maps arbitrary documents to relevant categories. Most of the learning methods reported in the literature deal with classifying text into a set of categories without structural relationships among them (*flat classification*). More recently, increasing attention has been given to *hierarchical classification* (D'Alessio, Murray, Schiaffino & Kershenbau, 2000; Dumais & Chen, 2000; Koller & Sahami, 1997; McCallum, Rosenfeld, Mitchell & Ng, 1998; Mladenić, 1998b; Ng, Goh & Low, 1997; Ruiz & Srinivasan, 2002; Weigend, Wiener, & Pedersen, 1999), where the pre-defined categories are organized in a hierarchical structure (tree-like structure). From an information retrieval viewpoint, this hierarchical arrangement is essential when the number of categories is high, since thematic search is made easier by browsing topics of interests. Yahoo, Google Directory, Medical Subject Headings (MeSH) in MEDLINE, Open Directory Project (ODP) and Reuters Corpus Volume I (RCV1) (Lewis, Yang, Rose & Li, 2004) are examples of search engines and text databases where documents are arranged in topic hierarchies.

The structural relationship among categories can be taken into account when devising the classification process. 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, which are associated to different levels of the hierarchy and 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. Another motivation is given by the observation that both precision and recall decrease as the number of categories increases (Apté, Damerau & Weiss, 1994; Yang, 1996), due to the increasing effect of term polysemy for large corpora.

On the other hand, taking into account the hierarchy opens additional issues in the development of methods for automated document classification. First, documents can either be associated to the leaves of the hierarchy or to internal nodes. Second, the set of features selected to build a classifier can either be category specific or the same for all categories (corpus-based). Third, the training set associated to each category may or may not include training documents of subcategories. Fourth, the classifier may or may not take into account the hierarchical relation between categories. Fifth, a stopping criterion is required for hierarchical classification of new documents in non-leaf categories. Sixth, performance evaluation criteria should take into account the hierarchy when considering classification errors.

All these issues are systematically investigated in this paper, which presents and evaluates a hierarchical text categorization framework that involves the hierarchy of categories in all phases of text categorization, namely feature extraction, learning and classification of a new document. The framework includes a tree distance-based thresholding algorithm for the classification of documents in internal categories of the hierarchy. It can be applied to any classifier that returns a degree of membership (e.g., distance- or probability-based) of a document to a category. In this work we consider three of the most widely investigated methods for (flat) text classification, namely centroid-based, naïve Bayes (NB) and support vector machines (SVM), and we investigate the performance of these methods on three datasets (Yahoo, DMOZ, RCV1). These datasets present a variety of situations in terms of hierarchical structure: documents can be assigned to any node in the hierarchy; some nodes can have no associated documents and internal nodes can have only one child. The baseline of the empirical evaluation is the flat classification, so that it is possible to analyze the actual contribution of the hierarchy in text classification performance. Another aspect considered in this framework is the construction of feature sets, which can be performed by merging the dictionaries of all subcategories (*hierarchical feature set*) or by taking the union of dictionaries of direct subcategories (*proper feature set*). The pros and cons of hierarchical feature sets are discussed and interactions with learning methods are empirically evaluated.

This paper extends and revises the work by Ceci and Malerba (2003) on hierarchical text classification. The main extensions are: 1) the consideration of hierarchical feature sets in feature selection; 2) the improvement of the naïve Bayes algorithm to avoid problems related to the different document length (Kim, Rim, Yook, & Lim, 2002); 3) the validation of the proposed framework also for a probabilistic SVM-based classifier; 4) 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; 5) a more extensive experimentation. A restriction with respect to previous work is that here we use only hierarchical training sets, which include documents of the subtree rooted in a category (positive examples) and documents of the sibling subtrees (negative examples). *Proper* training sets (see Fig. 1), which include documents of a category (positive examples) and documents of the sibling categories (negative examples), are not considered for two reasons. First, in Ceci and Malerba (2003) we have already showed that hierarchical training sets perform better than proper training sets. Second, when no training document

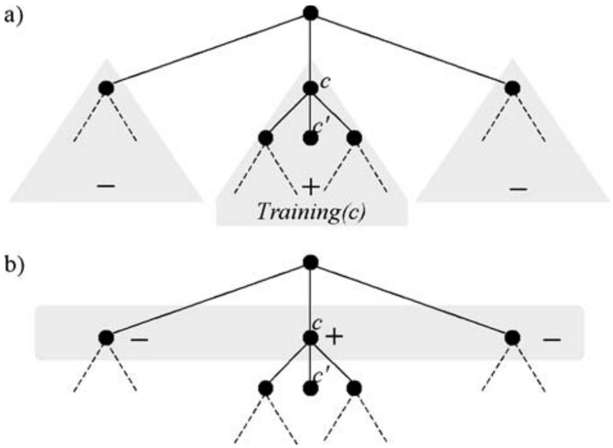


**Fig. 1** **a** Hierarchical training set; **b** proper training set

is associated to internal categories, as in the case of some datasets considered in this work, proper training sets cannot be used, since it would be impossible to build a classifier.

The paper is organized as follows. In the next section, the hierarchical classification framework is introduced in general terms, while in Section 3 the procedure for automated threshold determination is presented in detail. In both sections no reference to specific feature selection methods or learning algorithms is made: they are explained in Sections 4 and 5, respectively. To test alternative hierarchical text classification methods, a new release of the system WebClass (Malerba, Esposito & Ceci, 2002), named WebClassIII, has been implemented.[1] Some experimental results on three distinct datasets are reported and commented in Section 6. Finally, in Section 7 some related papers are discussed and the main differences compared with this work are reported. In the comparison, attention is focused on the method, the experimental setting and the empirical findings.

## 2 Hierarchical document classification: the framework

An important design issue of any hierarchical document classification framework is document representation. One of the frequently used approaches in content-based flat classification is the bag-of-words text representation, where each document is represented as a vector of numbers, each number corresponding to the occurrence of a particular word in the document and no ordering of words or any structure of text is considered. In the seminal work by Apté et al. (1994) two different types of representation are proposed: the same feature vector for all documents or several specialized feature vectors for different categories. The former is obtained by selecting features from a *universal* dictionary, built by examining all documents in the training set, while the latter is obtained by selecting a feature set from several *local* dictionaries, built for each category by examining only documents of that category.

The main advantage of using local feature sets is the large reduction of dimensionality. This is particularly true in flat classification, where the total number of categories is typically quite high, and different categories are characterized by different features. On the other hand, the uniqueness of the feature set permits the application of several statistical and machine learning algorithms (e.g., nearest neighbour or naïve Bayes classifier) defined for multi-class problems.[2] These algorithms are appropriate for single-class categorization and are theoretically founded on the assumption that all documents are points of the same (multidimensional) feature space.

In the context of hierarchical text classification a different, somewhat intermediate, solution can be adopted. Documents of both an internal category $c$ and its subcategories are represented by means of the *same* feature set, in order to build a classifier that assigns documents in $c$ to one of its direct subcategories. However, different internal categories may have different feature sets. In other words, by taking

[1] WebClassIII is available at the website http://www.di.uniba.it/~malerba/software/webclass/.

[2] A learning problem for classification into $r$ categories can be formulated in two different ways: either a *binary classifier* is induced for each category, or a *1-of-r* (or *multi-class*) *classifier* is learned to determine whether a new document belongs to one of the $r$ categories (Sebastiani, 2002).

into account the hierarchy, 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. For instance, documents of the general topic 'Math' can be well represented by general terms like "mathematics," while documents concerning specific topics (e.g., geometry) are better represented by specific terms like "parallelepiped." In the case of hierarchies representing *is*–a relations between categories, this multiple representation of documents means having several abstractions of the same entity (document), each of which is appropriate for a particular decision problem.

In our hierarchical text categorization framework (see Fig. 2), this multiple representation of documents is adopted, and a new document is classified by searching the hierarchy of categories. The search proceeds top-down from the root to the leaves according to a greedy strategy. When the document reaches an internal category *c*, it is represented on the basis of the feature set associated to *c*. The classifier of category *c* returns a score for each direct subcategory. Then, from the subcategories whose score is greater than the corresponding threshold the one with the highest score is chosen. The search proceeds recursively from that subcategory, until no score is greater than the corresponding threshold or a leaf category is reached. The last crossed node in the hierarchy is returned as the candidate category for document classification (*single-category classification*). If the search stops at the root, then the document is considered *unclassified*.

During the classification process the document is represented at decreasing levels of abstraction, since selected features tend to be more specific for lower level categories. These different representations of a document make the classification scores incomparable across different nodes in the hierarchy (e.g., in the case of naïve Bayes classifiers, posterior probabilities are defined on different probability spaces) and prevent the correct application of an exhaustive search strategy instead of the proposed greedy strategy.

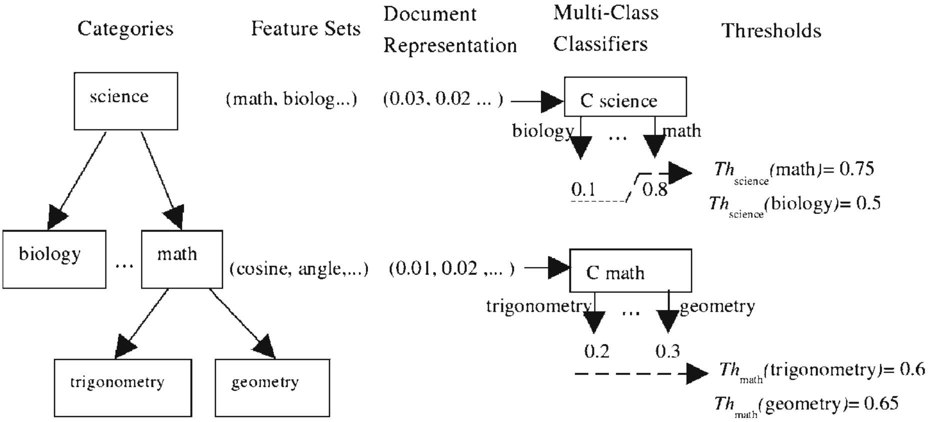


**Fig. 2** Classification of a new document. On the basis of the scores returned by the first classifier (associated to the category *science*) the document is passed down to *math*. The scores returned by the second classifier (associated to the category *math*), are not high enough to pass down the document to either *trigonometry* or *geometry*. Therefore, the document is classified in the *math* category

A special case is represented by categories with a unique direct subcategory. A probabilistic classifier would assign a unit probability to all documents that reach a category $c$ with a single subcategory $c'$, since no alternative to $c'$ is given. In this case, the thresholding procedure cannot work properly: if the threshold is less than one, all documents that reach $c$ would be passed down to $c'$, thus committing some errors for those documents that actually belong to $c$; otherwise, no document would be passed down to $c'$, thus committing some errors for those documents that should be actually classified in a subcategory of $c$. To avoid this problem, a dummy sibling category of $c'$ is introduced during the learning process. The training documents associated to the dummy subcategory are only those associated to $c$. The effect is that documents of $c$ can be considered as negative examples for all subcategories of $c$ itself. Therefore, the prior probabilities of all direct subcategories of $c$ do not sum to 1.0, since the possibility that the document belongs to no subcategory should be taken into account. While the dummy category is used during the learning process, it plays no role during the classification process, since scores associated to the dummy category are not considered. The assignment of the document to $c$ is based only on the thresholds, whose bottom-up automated determination permits the consideration of the final effect of local decisions taken by the classifier associated to $c$.

## 3 Automated threshold determination

As pointed out in the previous section, a classifier (either centroid-based, or naïve 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 category $c'$ among the direct subcategories of $c$ 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:

1. The document to be classified deals with a general rather than a specific topic, or
2. The document to be classified belongs to a specific category that is not present in the hierarchy and it 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, it is necessary to compute the thresholds that represent the "minimal score" (returned by the classifier), such that a document can be considered to belong to a direct subcategory. More formally, let $\gamma_{c\to c'}(d)$ denote the score[3] returned by the classifier associated to the internal category $c$, 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 the "minimal score" such that a document assigned to $c$ can be considered to belong to $c'$.

[3] As explained in the next sections, $\gamma_{c\to c'}(d) = P_c\left(c'|d\right)$ in the case of the naïve Bayes Classifier and the SVM probabilistic classifier, and $\gamma_{c\to c'}(d) = Sim_c\left(c', d\right)$ in the case of the centroid-based classifier.

The algorithm for automated threshold determination 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 notations are introduced:

1. $Hierarchy(c)$ is the hierarchy of categories rooted in $c$;
2. $DirectSubCategories(c)$ is the set of direct subcategories of $c$ in $Hierarchy(c)$;
3. $Training(c)$ is the set of positive examples in the hierarchical training set of category $c$;
4. $Training(c/c')= Training(c) - Training(c')$ is the set of positive examples in $Training(c)$ but not in $Training(c')$ for some $c' \in DirectSubCategories(c)$;
5. $\gamma_c(c') = \lfloor \gamma_{c \to c'}(d) | d \in Training(c') \rfloor$ is the list of values taken by the classifier for all documents of category $c'$ (or a subcategory);
6. $\gamma_c(\neg c') = \lfloor \gamma_{c \to c'}(d) | d \in Training(c/c') \rfloor$ is the list of values taken by the classifier for each document in $c$ or a subcategory $c''$ not in $Hierarchy(c')$;
7. $V = \gamma_c(c') \cup \gamma_c(\neg c')$ sorted in ascending order.

The algorithm (see Algorithm 1) is recursive and takes as input the category $c$ and a set of thresholds already computed for some siblings of $c$ and their descendants (at the first invocation, $c$ is the root category and *thresholdSet* is empty). It returns the union of the input set *thresholdSet* with the set of thresholds computed for all descendants of $c$. In particular, if $c'$ is a direct subcategory of $c$, the threshold $Th_c(c')$ associated to $c'$ 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 1** (Tree distance)
Let *Categories* be the set of all categories in a given *Hierarchy*. The tree distance $\delta_{Hierarchy}$ is a function $\delta_{Hierarchy} : Categories \to \mathbb{R}$ that associates two categories $c_1, c_2 \in Categories$ with a real value such that the following conditions are fulfilled:

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

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 (Esposito, Malerba, Tamma & Bock, 2000). 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.

The computation proceeds bottom-up, from leaves to the root. In a previous work (Ceci & Malerba, 2003) a top-down approach was proposed, which suffered from two limitations:

- It is conservative in the sense that it tends to classify documents in higher categories;
- When a threshold is defined it is impossible to take into account the possibly wrong decisions made by classifiers at lower levels of the hierarchy.

**Algorithm 1** Automated threshold definition algorithm for a category *c*.

```
find_thresholds(c,thresholdSet)
if $notleaf(c)$ then
  for all $c' \in DirectSubCategories(c)$ do
    $thresholdSet \leftarrow find\_thresholds(c', thresholdSet)$; {recursive bottom-up threshold determination}
    $compute\_and\_sort(V, c, c')$;
    $Th_c(c') \leftarrow 0$; $bestError \leftarrow \infty$;
    for all $k = 0, ..., |V|$ do
      {choose a possible threshold}
      if $k = 0$ then
        $threshold \leftarrow V[1] - \varepsilon$; {$\varepsilon > 0$}
      else if $k = |V|$ then
        $threshold \leftarrow V[k]$;
      else
        $threshold \leftarrow (V[k] + V[k+1])/2$;
      end if
      $error \leftarrow 0$; {compute tree distance-based errors}
      for all $v \in \gamma_c(c')$ do
        $let\ d \in Training(c') \quad such\ that \quad v = \gamma_{c \to c'}(d)$
        if $v > threshold$ then
          $error\ + =\delta_{Hierarchy(c')}(class(d), classify(d, thresholdSet, Hierarchy(c')))$
        else
          $error+ = \delta_{Hierarchy(c')}(class(d), c)$;
        end if
      end for
      for all $v \in \gamma_c(\neg c')$ do
        $let\ d \in Training(c/c') \quad such\ that \quad v = \gamma_{c \to c'}(d)$
        if $v > threshold$ then
          $error\ + =\delta_{Hierarchy(c)}(class(d), classify(d, thresholdSet, Hierarchy(c')))$
        else
          $error+ = 0$;
        end if
      end for
      if $error < bestError$ then
        $Th_c(c') \leftarrow threshold$; $bestError \leftarrow error$;
      end if
    end for
    $thresholdSet \leftarrow thresholdSet \cup \{\langle c', Th_c(c') \rangle\}$;
  end for
end if
return $thresholdSet$
```

Another difference is that in our previous work thresholds were determined by maximizing the *FScore* of the hierarchical classification on training documents. Although this approach gives promising results, it presents the limitation that the distance between "target" and "assigned" categories in the hierarchy is not considered when a misclassification error occurs.

The proposed algorithm may not find the set of thresholds that minimize the global error (globally optimal set), since the threshold associated to $c'$ is not determined by taking into account the thresholds associated to all other categories that are not subcategories of $c$. In fact, the problem of finding the absolute minimum in $\mathbb{R}^m$ (where $m$ is the number of thresholds, i.e., the number of nodes minus one) for a given function requires the exploration of the entire search space when no specific properties of the function (e.g., continuity, differentiability and monotonicity) can be defined. Since the error function Algorithm 1 minimizes does not show any particular property that permits to reduce the complexity of the searching problem we are forced to resort to a heuristic search based on a sequence of local reductions of errors (one for each branch of the tree). Global optimization by means of evolutionary algorithms or simulated annealing is postponed for future research.

## 4 The feature selection process

In this section the feature selection algorithm is briefly overviewed in order to better understand experimental results. We remark that feature selection is not the main topic of this paper, since the effect of feature selection in hierarchical categorization has already been investigated by Mladenić and Grobelnik (2003). In the proposed framework, the feature set is unique for each internal category and is automatically determined by means of a set of positive and negative training examples. More specifically, in WebClassIII, 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 *stopwords*, such as articles, adverbs, prepositions and other frequent words taken from Glimpse,[4] 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 (Porter, 1980).

Although these preprocessing steps reduce the number of extracted tokens, the feature set can still be large even in the case of small document collections. In many learning algorithms reduction of the set of features is essential for both complexity and accuracy. In particular, centroid-based methods compute the distance of a document from a centroid on the basis of all features used to describe the documents. If the attribution of a document to a category depends on only a few of the many available features, then the documents that are truly "close" to the centroid may

[4] glimpse.cs.arizona.edu.

well be far apart. Galavotti, Sebastiani and Simi (2000) and Ruiz and Srinivasan (2002) have independently proved that the Rocchio classifier, which is a particular centroid-based classifier, benefits from feature selection. It has also been proved that naïve Bayesian classifiers benefit from irrelevant feature removal (Mladenić, 1998a). The situation is different in the case of SVM classifiers, which work well with high dimensional feature spaces and eliminate the need for feature selection (Joachims, 1998b). In this work, where these three different learning methods are considered, feature selection is always performed for the purpose of having a fair comparison.

Several feature selection measures have been reported in the literature. They can be classified on the basis of four dependency tuples between a term $w$ and a category $c_i$ (Zheng, Wu & Srihari, 2004):

1. $(w, c_i)$: $w$ and $c_i$ co-occurs,
2. $(w, \neg c_i)$: $w$ occurs without $c_i$;
3. $(\neg w, c_i)$: $c_i$ occurs without $w$;
4. $(\neg w, \neg c_i)$: neither $w$ nor $c_i$ occur.

The first two tuples concern the *presence* of a term, while the last two are related to its *absence*. The first and the last tuples represent the *positive dependency* between $w$ and $c_i$, while the other two represent the *negative dependency*. Although all feature selection measures try to capture the intuition that the best terms for $c_i$ are the ones that are distributed most differently in the sets of positive and negative examples of $c_i$,[5] they consider different dependency tuples. For instance, Correlation Coefficient (Ng et al., 1997) considers all the four tuples, Mutual Information (Yang & Pedersen, 1997) considers the first three, while Odds ratio (Mladenić, 1998b) is based only on the first two. The variety of results reported in the literature does not allow us to state what dependencies should be involved in the definition of a good feature selection measure. As observed by Mladenić and Grobelnik (2003) “the most important characteristics of a good feature scoring measure for text are: favoring common features and considering domain and algorithm characteristics.”

Following this indication, in this work we focus our interest on the first two tuples, since the classifiers that will be presented in the next sections increase their confidence on classification on the basis of present terms rather than absent ones. Moreover, we adopt a global approach (a set of terms is chosen for classification under all categories (Sebastiani, 2002)), which seems best suited for multi-class classifiers. In the design of the feature selection measure reported in this work we take into account another important factor: the observation unit for all classifiers is *the document*, hence the “common features” referred to by Mladenic and Grobelnik, should not only be “frequent for a category,” but also *shared by most of documents of the same category*. A term that occurs frequently in very few documents of a category can be frequent for the category, but can hardly be considered a common feature. Surprisingly, a closer look at the feature selection measures reported in the literature reveals that most of them consider a term (and not a document) as the observation unit. By looking at the formulas of the most widely investigated feature selection

[5] A notable exception is the frequency of a term in a document collection, where only positive examples are considered.

measure reported in Mladenić and Grobelnik (1999) at Table 1, we find that the ingredients of various formulas are:

1. $P(w)$, the prior probability that the term $w$ occurs
2. $P(c_i)$, the prior probability of the $i$th class or category
3. $P(c_i|w)$, the conditional probability of the $i$th class value given that $w$ occurs
4. $P(w|c_i)$, the conditional probability of $w$ given the $i$th class value
5. $TF(w)$, the term frequency.

None of them actually refer to the document as the observation unit. For instance, the *absolute frequency of a term in a document*, *TF(w,d)*, which is used in the naïve Bayes classifier (see Section 5.1), is not considered. In the centroid-based classification, where it is important to select a set of features that increase the intra-class document similarity and decrease the inter-class document similarity, *the distribution of a term across training documents of the same category* is important, but it does not appear in the list above.

For multi-class problems, as those considered in the framework proposed in this paper, Malerba et al. (2002) developed a feature selection procedure that does take into account these observations. In this work, we develop an extension to the case of hierarchical training sets.

Let $c$ be a category and $c' \in DirectSubCategories(c)$. 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 $TF_d(w)$ on all training documents $d$ of category $c'$,

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

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

$$DF_{c'}(w) = \frac{|\{d \in Training(c')| \quad w \ occurs \ in \ d\}|}{|Training(c')|}$$

- The *category frequency* $CF_c(w)$, that is, the number of subcategories $c'' \in DirectSubCategories(c)$ such that $w$ occurs in a document $d \in Training(c'')$.

We observe that only documents considered as positive examples of $c'$ are used to compute both $TF_{c'}(w)$ and $DF_{c'}(w)$, while the estimation of $CF_c(w)$ also takes into account documents considered as negative examples of $c'$.

For each category $c'$, a list of pairs $\langle w_i, v_i \rangle$ is computed, such that $w_i$ is a term extracted from some document $d \in Training(c')$ and

$$v_i = TF_{c'}(w_i) \times DF^2_{c'}(w_i) \times \frac{1}{CF_c(w_i)}$$

By taking words that maximize the product $maxTF \times DF^2 \times ICF$, where ICF stands for "inverse CF," we reward common words used in documents of category $c'$, and we penalize words common to both $c'$ and its sibling categories. The category dictionary of $c'$, $Dict_{c'}$, is the set of the best $n_{dict}$ terms with respect to $v_i$, where $n_{dict}$ is a user defined parameter.

The measure $maxTF \times DF^2 \times ICF$ returns high scores for features that appear (possibly frequently) in many relevant documents and in documents of few alternative categories. In contrast with the correlation coefficient, it does not suffer from problems of unreliability for low frequency terms, so we are not forced to remove rare features as done by Ruiz and Srinivasan (2002) in their study on hierarchical text categorization. Moreover, it is not influenced by the marginal probability of terms as in the case of mutual information (Yang & Pedersen, 1997), which makes the score incomparable across terms of widely differing frequency.

The feature set associated to a category $c$ is defined on the basis of the dictionaries of its subcategories.[6] More precisely, the *p*roper feature set $FeatSet_c$ is defined as the union of the dictionaries of all direct subcategories of $c$ (see Fig. 3):

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

It contains features that appear frequently in many documents of one of the subcategories, but that 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 for classifying 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") for upper level categories, and a set of specific features (like "topolog") for lower level categories.

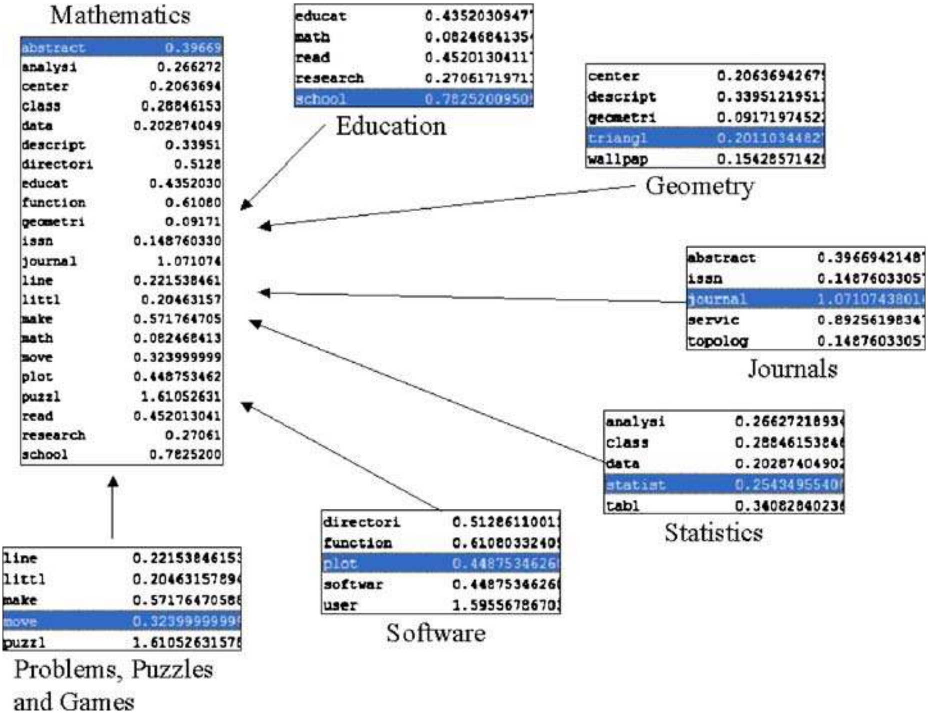
An alternative proposal is the *hierarchical feature set*, which is defined as the union of the dictionaries of *all* subcategories (similar to Mladenić (1998b) where, in addition, weights are used to give less importance to subcategories that are further down in the hierarchy):

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

The rationale behind the hierarchical feature set is that, if classifiers at the top level do take into account only general terms (such as "math") typically extracted from documents of general topics (e.g., Mathematics), they might have some difficulties correctly routing along the right path those documents belonging to leaf categories (e.g., Geometry), because of the rarer occurrence of general terms.

Once the set of features has been determined for an internal category $c$, training documents in $Training(c)$ can be represented as feature vectors, where each feature value is the frequency of a word.

[6] McCallum et al. (1998) use the term *hierarchical feature selection* to denote the selection of an equal number of features at each internal node of the tree, using the node's immediate children as the classes.



| | |
|---|---|
| abstract | 0.39669 |
| analysi | 0.266272 |
| center | 0.2063694 |
| class | 0.28846153 |
| data | 0.202874049 |
| descript | 0.33951 |
| directori | 0.5128 |
| educat | 0.4352030 |
| function | 0.61080 |
| geometri | 0.09171 |
| issn | 0.148760330 |
| journal | 1.071074 |
| line | 0.221538461 |
| littl | 0.20463157 |
| make | 0.571764705 |
| math | 0.082468413 |
| move | 0.323999999 |
| plot | 0.448753462 |
| puzzl | 1.61052631 |
| read | 0.452013041 |
| research | 0.27061 |
| school | 0.7825200 |

| | |
|---|---|
| educat | 0.4352030947 |
| math | 0.0824684135 |
| read | 0.4520130411 |
| research | 0.2706171971 |
| school | 0.7825200950 |

| | |
|---|---|
| center | 0.2063694267 |
| descript | 0.3395121951 |
| geometri | 0.0917197452 |
| triangl | 0.2011034482 |
| wallpap | 0.1542857142 |

| | |
|---|---|
| abstract | 0.3966942148 |
| issn | 0.1487603305 |
| journal | 1.0710743801 |
| servic | 0.8925619834 |
| topolog | 0.1487603305 |

| | |
|---|---|
| analysi | 0.2662721893 |
| class | 0.2884615384 |
| data | 0.2028740490 |
| statist | 0.2543495540 |
| tabl | 0.3408284023 |

| | |
|---|---|
| directori | 0.5128611001 |
| function | 0.6108033240 |
| plot | 0.4487534626 |
| softwar | 0.4487534626 |
| user | 1.5955678670 |

| | |
|---|---|
| line | 0.2215384615 |
| littl | 0.2046315789 |
| make | 0.5717647058 |
| move | 0.3239999999 |
| puzzl | 1.6105263157 |

**Fig. 3** Category dictionaries extracted by WebClassIII for all subcategories of "Mathematics" in an experiment on Yahoo dataset ($n_{dict} = 5$) and proper feature set selected for "Mathematics"

## 5 The learning process

In the hierarchical text categorization framework, the definition of the same feature set to represent documents of a category $c$ and all its subcategories permits the application of a multi-class learning algorithm to induce a classifier that categorizes a document (temporarily) assigned to $c$ as belonging to a subcategory $c'$ of $c$. In this work we consider three different learning approaches:

1. Naïve Bayes (Mitchell, 1997) modified in order to correctly handle documents of different length;
2. A centroid-based method (Han & Karypis, 2000), where each centroid (or class prototype) is the center of a cluster of documents of the same category;
3. SMO, which is an optimized algorithm for training SVM on very large data sets (Platt, 1998).

Therefore, the classification of a new document in a category $c'$ is obtained as follows:

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

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

The three learning algorithms are briefly described in the next subsections, while in the last subsection the learning complexity of the hierarchical text categorization framework is formally evaluated.

### 5.1 Naïve Bayes classifier

Let $d$ be a document temporarily assigned to a category $c$. We intend to classify $d$ into one of the subcategories of $c$. The Bayes optimal classification can be achieved by assigning $d$ to the category $c' \in DirectSubCategories(c)$ that maximizes the posterior probability $P_c(c_i|d)$.

In the literature, several Bayesian models have been proposed for text categorization. The *naïve Bayes classifier* is the simplest of these models, in that it assumes that all the features used to describe the document are independent of each other, given the context of the class (class conditional feature independence). Although this assumption is clearly false in text categorization, naïve Bayes classifiers perform surprisingly well. The explanation is that classification depends only on the magnitude of the conditional probabilities (e.g., the sign of the difference of conditional probabilities in the binary case), therefore, even when the approximation of conditional probability is poor, the classification accuracy remains high (Domingos & Pazzani, 1997).

In the text categorization literature, two different models based on the naïve Bayes assumption have been proposed: the *multivariate Bernoulli model* and the *multinomial model* (McCallum & Nigam, 1998). The former specifies that a document is represented by a vector of binary attributes indicating which terms occur and do not occur in the document. The "event" is the document, and both the presence and the absence of a term contribute to the estimation of the posterior probability, which is modelled as a multivariate Bernoulli. In the context of hierarchical text categorization it has been used by Koller and Sahami (1997). The multinomial model specifies that a document is represented by the set of term occurrences in the document. In this case the "event" is the term and the number of occurrences of each term affects the posterior probability, which is based on a multinomial model. In hierarchical text categorization this model has been used by Mladenić (1998b). A review of naïve Bayes classifiers and their usage in information retrieval is reported in Lewis (1998), where the Bernoulli model is named the *binary independence model*.

McCallum and Nigam (1998) have shown that, in a number of different text categorization problems, the multinomial model is capable of categorizing documents more accurately than the multivariate Bernoulli model. Eyheramendy, Lewis and Madigan (2003) have considered three alternatives to the multinomial model, all of which similarly incorporate term frequencies. The authors have empirically shown that the multinomial model often outperforms these alternatives. Therefore, in this work we consider the naïve Bayes classifier based on the multinomial model. This choice is also coherent with the feature selection process, where only the presence (and not the absence) of a feature is considered, and the number of occurrences of a term is an important factor in feature selection.

In its general formalization the multinomial model accommodates very naturally with the document length. The posterior probability $P_c(c_i|d)$ can be defined as the sum over posterior probabilities of documents of different length (Joachims, 1997):

$$P_c(c_i|d) = \sum_{l=1}^{\infty} P_c(c_i|d,l) P_c(l|d) \tag{1}$$

where $P_c(l|d) = 1$ for the length $l_d$ of document $d$ and is zero otherwise. In other terms, $P_c(c_i|d) = P_c(c_i|d, l_d)$. By applying Bayes' theorem to $P_c(c_i|d)$ we have:

$$P_c(c_i|d) = \frac{P_c(d|c_i, l_d) P_c(c_i|l_d)}{\sum\limits_{c' \in DirectSubCategories(c)} P_c(d|c', l_d) P_c(c'|l_d)} \tag{2}$$

$P_c(c_i|l_d)$ is the prior probability that a document of length $l_d$ is in class $c_i$. By assuming that *the category of a document does not depend on its length*, we can write $P_c(c_i|l_d) = P_c(c_i)$. The prior probability $P_c(c_i)$ is estimated as the fraction of training documents of $c$ assigned to class $c_i$:

$$P_c(c_i) = \frac{|Training(c_i)|}{\sum\limits_{c' \in DirectSubCategories(c)} |Training(c')|} \tag{3}$$

The estimation of the likelihood $P_c(d|c_i, l_d)$ is based on the multinomial model:

$$P_c(d|c_i, l_d) = \frac{l_d!}{\prod\limits_{w \in FeatSet_c} TF(w,d)!} \prod_{w \in FeatSet} P_c(w|c_i, l_d) \tag{4}$$

where $TF(w, d)$ denotes the absolute frequency of $w$ in $d$.

The first term depends only on the document $d$ and multiplies both the numerator and the denominator of formula (2), hence it can be dropped. The subsequent terms are the probabilities of observing a term $w$ of the feature set in documents of length $l_d$ and of class $c_i$. Unfortunately, the estimation of this conditional probability is quite difficult, since we should consider only documents of length $l_d$ in the training set. Therefore, a further simplifying assumption is usually made, that the occurrence of a term is only dependent on the membership class of a document (Joachims, 1997). By combining this assumption with the original feature independence assumption we have:

$$P_c(d|c_i, l_d) \propto \prod_{w \in FeatSet_c} P_c(w|c_i)^{TF(w,d)} \tag{5}$$

In conclusion, under the assumption that each term in $d$ occurs independently of other terms, as well as independently of the text length, it is possible to estimate the posterior probability as follows:

$$P_c(c_i|d) = \frac{P_c(c_i) \prod\limits_{w \in FeatSet_c} P_c(w|c_i)^{TF(w,d)}}{\sum\limits_{c' \in DirectSubCategories(c)} P_c(c') \prod\limits_{w \in FeatSet_c} P_c(w|c')^{TF(w,d)}} \tag{6}$$

To make our probability estimate of $P_c(w|c_i)$ more robust with respect to infrequently used terms, we use a smoothing method to modify the estimates that would have been obtained by simple event counting. Smoothing, whose main effect is that of assigning a small, non-null probability to unobserved events, is important in naïve Bayes classifiers, since probability estimates are multiplied. If only one of them were zero at the numerator, the posterior probability in Eq. 6 would be zero, regardless of the values of the other estimates. In this work smoothing is based on Laplace's law of succession, that is:

$$P_c(w|c_i) = \frac{1 + PF(w, c_i)}{|FeatSet_c| + \sum\limits_{w' \in FeatSet_c} PF(w', c_i)} \tag{7}$$

where $PF(w, c)$ denotes the absolute frequency $w$ in documents of category $c$. An alternative to the Laplace estimator is Witten–Bell smoothing, that has been used in the work by Craven and his colleagues on text categorization (Craven et al., 2000).

The main weakness of this naïve Bayesian classifier is that 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. Consequently, scores for the winning class tend to be close to 1.0, while scores for the losing classes tend toward 0.0. For instance, Bennett (2000) shows this phenomenon on two classes (Earn and Corn) of the well-known Reuters 21578 dataset. 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 (Craven et al., 2000). An additional problem in the above formulation is strictly related to the probability estimation in formula (5), 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 (Kim et al., 2002) and estimation could be affected by significant length discrepancy among documents belonging to the same class (Sebastiani, 2002). As observed by Eyheramendy et al. (2003), “directly incorporating document length into the multinomial model has little effect due to the extreme probability estimates produced by the naïve Bayes-type models. One possibility would be to correct for the bias before introducing length.”

In our proposal we adopt a normalization of the value $TF(w, d)$ in formula (6) in order to avoid these problems. In particular, we normalize $TF$ according to the following formula:

$$NormalizedTF(w, d) = \frac{TF\,(w, d)}{\|TF(\bullet, d)\|_2} \tag{8}$$

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 Eq. 6, we have:

$$P_c\,(c_i|d) = \frac{P_c\,(c_i) \cdot \prod\limits_{w \in FeatSet_c} P_c\,(w|c_i)^{NormalizedTF(w,d)}}{\sum\limits_{c' \in DirectSubCategories(c)} P_c\,(c') \cdot \prod\limits_{w \in FeatSet_c} P_c\,(w|c')^{NormalizedTF(w,d)}} \tag{9}$$

We observe that this normalization does not change the assignment of a document to a class: it only contributes to smoothing the values of the posterior probabilities and to making the thresholding algorithm more effective, since choosing a threshold when probability values are all 0 or 1 would not help in hierarchical text classification. A similar normalization, but to $L_1$-norm, has been proposed by Shen and Jiang (2003).

### 5.2 Centroid-based classifier

Linear classifiers are a family of learning algorithms that learn a feature weight vector (or prototype)

$$\vec{c}_i = \left\langle w_{i1}, w_{i2}, \ldots, w_{i|FeatSet_c|} \right\rangle$$

for every category $c_i$. In our framework, where a document $d$ temporarily assigned to a category $c$ has to be possibly assigned to a category $c_i \in DirectSubCategories(c)$, the dimensionality of the feature weight vector of $c_i$ corresponds to the size of $FeatSet_c$. The score returned by a linear classifier for a document $d$ and a category $c_i$ is the dot product between the feature vector describing $d$ and $\vec{c}_i$ (hence the *linearity* of the classifier). Generally, the dot product (or equivalently, both the document and the class vectors) is normalized to the unit as follows:

$$\frac{\vec{d} \cdot \vec{c}_i}{\left\|\vec{d}\right\|_2 \left\|\vec{c}_i\right\|_2}$$

This normalization represents the cosine of the angle spanned by the two vectors $d$ and $\vec{c}_i$. It is a similarity measure (also known as *cosine similarity*), therefore, the higher the value, the more similar the document $d$ and the category prototype $\vec{c}_i$.

The most well-known linear classifier is an adaptation to text categorization of Rocchio's formula, originally proposed for relevance feedback in the context of information retrieval (Rocchio, 1971). The learning method, denoted as the Rocchio method, computes the weights of $\vec{c}_i$ as follows:

$$w_{ij} = \beta \sum_{d \in Training(c_i)} \frac{d_j}{|Training(c_i)|} - \gamma \sum_{d \in Training(c/c_i)} \frac{d_j}{|Training(c/c_i)|}$$

where $d_j$ denotes the *j*th component of the document vector, *Training($c_i$)* is the set of positive documents of category $c_i$, and *Training(c/$c_i$)* in our framework is the set of negative examples for $c_i$. The control parameters $\beta$ and $\gamma$ define the relative impact of positive and negative examples in the definition of the class prototype. Dumais, Platt, Heckerman and Sahami (1998), Joachims (1997), Han and Karypis (2000), Lertnattee and Theeramunkong (2004) set $\beta$ to 1 and $\gamma$ to 0, so that the prototype of a class coincides with the *centroid* of its positive training examples. In this work we follow this mainstream and compute the classification score as the cosine similarity between the document vector and the centroid of a class. The main difference is that document vectors contain the term frequencies, that is $d_j = TF_d(w_j)$, while all mentioned works operate on *tfidf* representations, that is, the weight associated to the *j*-th feature is the product of the term frequency of the term $w_j$ in $d$, $TF_d(w_j)$, and the logarithm of the inverse document frequency, *IDF($w_j$)*. The document frequency is defined as the percentage of documents in the collection where the term $w_j$ occurs.[7] The *tfidf* representation embodies the intuition that

- The more often a term occurs in a document, the more it is representative of its content, and
- The more documents a term occurs in, the less discriminating it is (Sebastiani, 2002).

The second intuition is appropriate for document indexing, that is, the task of information retrieval for which Salton and Buckley (1988) defined the *tfidf* representation. However, for text categorization tasks, the usage of the IDF factor seems counterintuitive. In the feature selection phase, the most discriminant features are selected, such that they correspond to terms that occur frequently in documents of the same category. The IDF factor would penalize mainly the best discriminative features, while it would weight more those terms that occur frequently in a single document. A confirmation of our observation is indirectly given by Debole and Sebastiani (2003), who suggest replacing the IDF factor with the value taken by the feature selection measure. Therefore, in this work the weight associated to the *j*th feature of a document is based exclusively on the TF factor. In this case, the value associated to the feature $w$ for the *centroid* of the category $c_i$ is defined as follows:

$$P_c(w, c_i) = \frac{\sum_{d \in Training(c_i)} TF_d(w)}{|Training(c_i)|} \tag{10}$$

and the mathematical formulation of the *cosine correlation* is the following:

$$Sim_c(c_i, d) = \frac{\sum_{w \in FeatSet_c} P_c(w, c_i) \times TF_d(w)}{\sqrt{\sum_{w \in FeatSet_c} P_c(w, c_i)^2 \times \sum_{w \in FeatSet_c} TF_d(w)^2}} \tag{11}$$

It is noteworthy that the cosine correlation returns a particularly meaningful value when vectors are highly dimensional and features define orthogonal directions. As pointed out in Section 4 our feature selection algorithm guarantees a kind of orthogonality property, which applies to the group of features extracted from each category dictionary rather than to the individual features. Therefore, the procedure adopted for feature selection seems to be coherent with this classifier as well.

We conclude by highlighting another difference with respect to related papers by Joachims (1997), Han and Karypis (2000) and Lertnattee and Theeramunkong (2004), where all features are used in their experiments.[8] In this work, features are preliminarily filtered and only those deemed most discriminant actually contribute to the classification. This seems to improve the accuracy of Rocchio classifiers (Ruiz

[7] The document frequency used in the *tfidf* representation should not be confused with $DF_{c'}(w)$ defined in Section 4, which depends on the category $c'$.

[8] Joachims (1997) actually filters out all features that occur less then three times in the training documents.

& Srinivasan, 2002), which can achieve quite competitive performance if properly trained (Schapire, Singer & Singhal, 1998).

### 5.3 SVM-probabilistic classifier

Recently, a new learning technique has emerged and become quite popular in text categorization because of its good performance and its theoretical foundations in the computational learning theory: support vector machines (SVMs), proposed by Vapnik (1995). Given a set of positive and negative examples $\{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots, (\vec{x}_N, y_N)\}$, where $\vec{x}_i \in \mathbb{R}^m$ ($\vec{x}_i$ is a document vector) and $y_i \in$\{-1,+1\}, an SVM identifies the hyperplane in $\mathbb{R}^m$ that linearly separates positive and negative examples with the maximum margin (*optimal separating hyperplane*). In general, the hyperplane can be constructed as the linear combination of all training examples, however, only some examples, called *support vectors*, do actually contribute to the optimal separating hyperplane. The coefficients of the linear combination are determined by solving a large-scale quadratic programming (QP) problem, for which efficient algorithms that find the global optimum exist.

The linear separability appears to be a strong limitation, however, as experimentally observed by Joachims (1998b), most text categorization problems are linearly separable. In any case, SVMs can be generalized to non-linearly separable training data by mapping the data into another *feature space* $F$ via a non-linear map:

$$\Phi : \mathbb{R}^m \quad \rightarrow \quad F$$

and then performing the above linear algorithm in $F$. Yang and Liu (1999) reported that they have tested the linear and non-linear models offered by the SVM$^{light}$ system (Joachims, 1998a), and obtained “a slightly better result with the linear SVM than with the non-linear models.” Therefore, in our experiments we will use only linear models.

SVMs are based on the *Structural Risk Minimization* principle: a function that can classify training data accurately and which belongs to a set of functions with the lowest capacity (particularly in the VC-dimension) (Vapnik, 1995) will generalize best, regardless of the dimensionality of the feature space $m$. Therefore, SVMs can generalize well even in large feature space, such as those used in text categorization. In the case of the separating hyperplane, minimizing the VC-dimension corresponds to maximizing the margin.

The SVM embedded in WebClassIII is a modified version of the Sequential Minimal Optimization classifier (SMO) (Platt, 1998). The method developed by Platt is very fast and is based on the idea of breaking a large QP problem down into a series of smaller QP problems that can be solved analytically. The same system has been used by Dumais et al. (1998) in an empirical comparison of five different learning algorithms for text categorization. Since SVMs are defined for two-classes problems, we modified Platt’s original method to learn a *one–of–r* (i.e., multi-class) classifier for each internal node of the hierarchy. 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 (Hastie & Tibshirani, 1998). 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.

### 5.4 Learning complexity

In the hierarchical text categorization framework, the original learning problem is partitioned into smaller subproblems, each of which can be efficiently managed. This leads to an efficiency gain, with respect to the flat classifier, that depends on the number of classes associated to each learned classifier.

More formally, let

- *f(number of classes, number of training documents, number of features)* be the learning complexity of a generic classification algorithm,
- $r$ be the total number of classes
- $n$ be the number of training documents
- $a$ be the number of features
- $d$ be the depth of the hierarchy
- $k$ be the number of children of a generic internal node (for simplicity, in this analysis we suppose that $k$ is constant).

Then the complexity of a flat classifier is: $f(r, n, a)$.

The complexity of the hierarchical framework is:

- $f(k, n, a)$ for the first level;
- $k \cdot f(k, n/k, a)$ for the second level, in the worst case that all documents are classified in lower categories[9]
- $k^2 \cdot f(k, n/k^2, a)$ for the third level.

By generalizing, the complexity of the hierarchical framework is:

$$\sum_{i=1}^{d} k^i f\left(k, \frac{n}{k^i}, a\right) \quad .$$

In the case of both naïve Bayes and centroid-based classifiers, the complexity of the learning phase is linear in the number of training documents, in the number of features and in the number of classes (Han & Karypis, 2000; Mitchell, 1997). In such a case the time complexity of a flat classifier is O($n{\cdot}a{\cdot}r$), while in the case of a hierarchical framework, it is:

$$O\left(\sum_{i=1}^{d} k^i \cdot \left(\left(\frac{n}{k^i} \cdot k \cdot a\right)\right) = O\left(\sum_{i=1}^{d} (n \cdot k \cdot a)\right) = O(d \cdot n \cdot k \cdot a)$$

Both are linear in the number of training examples and in the number of features. The difference is that the complexity of a flat classifier is linear in the number of classes, while the complexity of the hierarchical framework is linear in the product of the number of children of each node and the depth of the tree. In the extreme case of $k = 1$, the two approaches have the same complexity O($n{\cdot}a{\cdot}r$). The main difference occurs in the case of a balanced hierarchy where $d{=}log_k r$ and the complexity becomes O($n \cdot a \cdot log_k r$).

In the case of the SVM classifier, the complexity is linear in the number of training documents, features and classes (Platt, 1998). However, the SMO has been modified

[9] We assume the uniform distribution of documents among direct subcategories.

to deal with multi-class problems and to estimate the probability $P_c(c_i|d)$. This probability is computed by means of a probabilistic pair-wise coupling classification (Hastie & Tibshirani, 1998). This modification makes the algorithm linear in the number of examples and cubic in the number of classes. Therefore, the time complexity of a flat classifier is O($n{\cdot}a{\cdot}r^3$), while in the case of the hierarchical framework it is:

$$O\left(\sum_{i=1}^{d} k^i \cdot \left(\frac{n}{k^i} \cdot k^3 \cdot a\right)\right) = O\left(\sum_{i=1}^{d} \left(n \cdot k^3 \cdot a\right)\right) = O\left(d \cdot n \cdot k^3 \cdot a\right)$$

This time, also in the extreme case of $k = 1$, the hierarchical classifier is computationally more efficient than the flat classifier and the complexity is O($n{\cdot}a{\cdot}r$). In the case of a balanced hierarchy where $d{=}log_k r$, the complexity of the hierarchical framework is O($n \cdot a \cdot log_k r$).

This analysis can be refined by taking into account that the value of $a$ (i.e., number of features) may change level by level. More precisely:

$a = n_{dict} \cdot r$ in the flat classifier,
$a = n_{dict} \cdot k$ in the hierarchical framework with a proper feature set,
$a < n_{dict} \cdot r$ in the hierarchical framework with a hierarchical feature set.

In fact, in the case of a hierarchical framework with a hierarchical feature set, the number of features depends on the level of the hierarchy to which the classifier is associated. For the first level $a = n_{dict} \cdot r$, in the second level $a = n_{dict} \cdot (r - k)$, in the third level $a = n_{dict} \cdot (r - k - k^2)$ and so on.

## 6 Experiments

In this section we seek answers to the following questions with empirical evidence:

- Does the hierarchical classifier built with the proposed framework improve the performance when compared to a flat classifier?
- Does the proposed framework minimize the (tree) distance between the correct class and the returned one when the document is not correctly classified?
- Does the proposed framework actually improve the computational efficiency of the learning algorithms?
- What feature selection strategy is the most promising for hierarchical categorization?
- Which classifier has the best performance within the proposed framework?

Before describing the results, we illustrate the three corpora used for this study and the performance evaluation measures considered.

### 6.1 Datasets

The three corpora chosen for this study are the recently published benchmark dataset Reuters Corpus Volume I (RCV1) (Lewis et al., 2004) and two collections of HTML documents (WebClass is specifically designed to classify HTML pages) referenced either in the Yahoo! Search Directory[10] or in a web directory developed in the Open Directory Project (ODP).[11] The three corpora differ considerably in the training set size, in the hierarchical structure of categories, as well as in the procedure adopted for the classification of documents. For the sake of completeness, a brief description of the document collections is reported in the following.

### *6.1.1 Reuters Corpus Volume 1*

Reuters Corpus Volume I (RCV1) is a benchmark dataset widely used in text categorization and in document retrieval. It consists of over 800,000 newswire stories, collected by the Reuters news and information agency. The stories have been manually coded using three orthogonal category sets. Therefore, category codes from three sets (Topics, Industries, and Regions) are assigned to stories:

- Topic codes capture the major subject of a story.
- Industry codes are assigned on the basis of the types of business discussed in the story.
- Region codes include both geographic locations and economic/political groupings.

In our study, similar to other authors (Zhang, Jin, Yang & Hauptmann, 2003), we use topic codes for categorization.

The main characteristic that makes RCV1 particularly suitable in our study is the adopted coding policy. In particular, topics are organized hierarchically. The hierarchy of topics consists of a set of 104 categories organized in a four-level hierarchy.

We pre-processed documents as proposed by Lewis et al. and, in addition, we considered only documents associated to a single category. This selection is due to the fact that in this study we are interested in investigating single category assignment (feature selection method, learning algorithms, categorization framework and performance evaluation functions are all based on the assumption that a document can be assigned to one category at the most). The removal of documents associated with multiple classes has also been adopted by other authors on different datasets in the evaluation of single-label corpora (Schapire & Singer, 2000).

We separate the training set and the testing set using the same split adopted by Lewis et al. In particular, documents published from August 20, 1996 to August 31, 1996 (document IDs 2286 to 26150) are included in the training set, while documents published from September 1, 1996 to August 19, 1997 (document IDs 26151 to 810596) are considered for testing. The result is a split of the 804,414 documents into 23,149 training documents and 781,265 test documents. After multiple-label document removal, we have 150,765 documents, (4,517 training documents and 146,248 testing documents).

[10] http://dir.yahoo.com/.

[11] www.dmoz.org.

### *6.1.2 Yahoo dataset*

The second data set used in this experimental study is obtained from the documents referenced in the Yahoo! Search Directory.[12] We extracted all 907 actual Web documents referenced at the top three levels of the Web directory http://dir.yahoo.com/Science. Empty documents and documents containing only scripts were removed.

There are six 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.

The dataset is analyzed by means of a five-fold cross-validation, that is, the dataset is first divided into five *folds* of near-equal size, and then, for every fold, the learner 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.

### *6.1.3 Dmoz dataset*

The third data set used in this experimental study is obtained from the documents referenced by the Open Directory Project (ODP).[13] We extracted all actual Web documents referenced at the top five levels of the Web directory rooted in the branch Health\Conditions_and_Diseases\. Empty documents and documents containing only scripts were removed.

The dataset contains 5,612 documents in 221 categories organized in a five level hierarchy as follows:

- In the first level there are 21 categories and 340 documents.
- In the second level there are 81 categories and 1,514 documents.
- In the third level there are 85 categories and 2,604 documents.
- In the fourth level there are 32 categories and 1,099 documents.
- In the fifth level there are two categories and 55 documents.

The dataset is analyzed by means of a five-fold cross-validation. The system performance is evaluated by averaging performance measures on the five cross-validation folds.

Both Yahoo and Dmoz datasets were used in order to evaluate the performances of the system in the presence of "noisy" documents and in the presence of documents with no clearly predefined structure. This is not the case of the corpus RCV1, whose documents respect a well-defined XML structure.

## 6.2 Evaluation measures

Performances of the system have been evaluated on the basis of several measures. The first measure is the standard *accuracy* defined in machine learning to evaluate

[12] Documents were downloaded on the 15th of July 2003. The dataset is electronically available at http://www.di.uniba.it/%7ececi/micFiles/yahoo_science_docs.zip.

[13] Documents were extracted in April 2004. The dataset is electronically available at http://www.di.uniba.it/%7ececi/micFiles/dmoz_health_conditions_and_diseases_docs.zip.

the performances of *1-of-r* classifiers. It represents the number of testing documents correctly classified over all testing documents. It is noteworthy that in the *1-of-r* classifiers context, this "narrowly" defined accuracy is indeed equivalent to the standard recall and is not equivalent to the standard definition of accuracy in text categorization literature that is given for classifiers based on binary decisions. In this case it is the proportion of correct assignments among the binary decisions over all category/document pairs. The standard text categorization accuracy measure is well-defined for documents with multiple categories; the narrowly defined accuracy is not (Yang & Liu, 1999). In our analysis, we use the narrowly defined accuracy because, as observed by Sebastiani (2002), in single-label text categorization, precision and recall are not independent of each other and in this case either precision or recall (machine learning accuracy) can be used as a measure of effectiveness.

Furthermore, we define other four additional evaluation measures, in order to provide a more detailed evaluation of results. Intuitively, if a text categorization 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. The four evaluation measures we consider are:

1. The *misclassification error*, which computes the percentage of documents misclassified into a category not related to the correct category in the hierarchy.
2. The *generalization error*, which computes the percentage of documents misclassified into a supercategory of the correct category;
3. The *specialization error*, which computes the percentage of documents misclassified into a subcategory of the correct one;
4. The *unknown ratio*, that measures the percentage of rejected documents.

The sum of the accuracy, the generalization error, the specialization error, the misclassification error and the unknown ratio equals one.

### 6.3 Flat vs hierarchical classifiers

The first question we investigate is the effectiveness of the hierarchical categorization framework with respect to flat classification. For a fair comparison, the thresholding algorithm has been used both for hierarchical and flat classification. In this way, both algorithms are able to "reject" documents.

For evaluation purposes, several feature sets (proper or hierarchical) of different size have been extracted for each internal category in order to investigate the effect of this factor on the system performance. The feature set size ranges from 5 to 60 features per category in the case of RCV1 and Yahoo dataset, while it ranges from 5 to 40 in the case of Dmoz dataset. Collected statistics concern the three classifiers.

Figure 4 shows the accuracy of different classifiers for the three datasets.[14] *Among flat classifiers, SVM performs the best across the three datasets*. It is also noteworthy that in all datasets the SVM or centroid-based classifiers, built according to the flat approach, are more accurate than the corresponding two hierarchical

[14] Experimental results for flat SVM are available only up to 20 features per category, since the high computational complexity of the method prevented the generation of about $10^7$ classifiers with more than 4,500 features in reasonable time (see Section 5.4 for considerations on computational complexity).

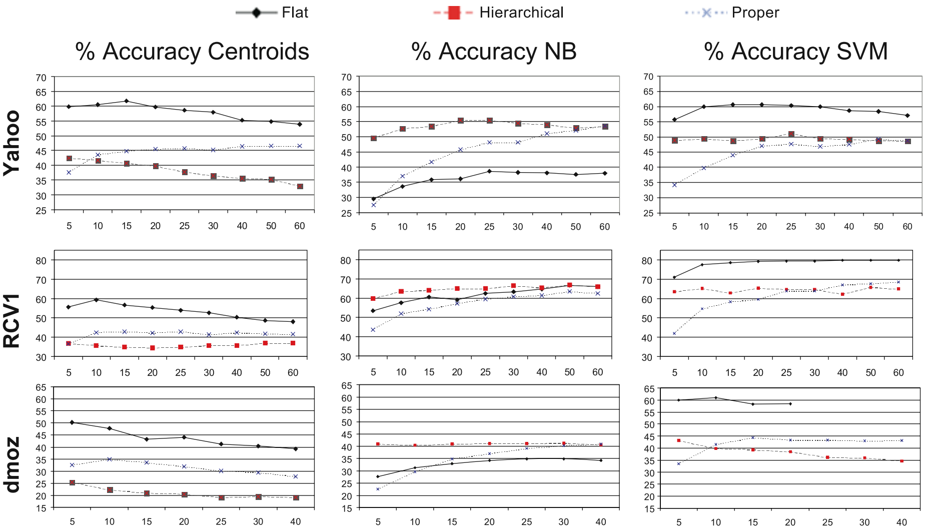


**Fig. 4** Accuracy for the three datasets: flat vs hierarchical with hierarchical feature set vs hierarchical with proper feature set

classifiers, built on proper or hierarchical feature sets. The situation is different for NB classifiers, which do benefit of the hierarchical framework in all three datasets. This is particularly evident for NB classifiers built from hierarchical feature sets.

The variance in performance of the three methods can be explained by the effect of the number of categories in the naïve Bayesian classifier. In the flat approach, where the classifier has to discriminate between the whole set of categories, prior probabilities are very small and the numerator in Eq. 9, that is, the score computed by the NB classifier, is consequently small. In this situation is difficult to determine appropriate thresholds and the rate of unclassified documents remains high because of the conservative thresholds found by the thresholding algorithm. The high number of categories, which is a problem for the flat NB classifier, does not affect centroid-based classifiers and SVMs, since they are not based on the computation of a prior probability. On the contrary, the hierarchical approach seems to cause problems in distance-based classifiers (centroids and SVMs), since training documents of high-level categories form partially overlapping clouds of points in the multi-dimensional space and the selection of appropriate thresholds becomes hard. The method that most of all suffers from this problem for high-level categories is the centroid-based, since boundaries for $r$ categories is based on as many centroids, while SVMs can use $s \gg r$ support vectors. Therefore, our second conclusion is that *there is an interaction, in terms of accuracy, between the hierarchical framework and the type of classifier*.

It is noteworthy that the flat classifiers and the hierarchical classifiers with hierarchical feature sets are positively correlated, that is, they both increase or decrease with the number of features. This can be explained by the observation that, for high level categories, where the majority of errors tend to concentrate, the hierarchical classifiers with hierarchical feature sets are trained with approximately the same set of features of flat classifiers. Therefore, if the increase in the number of features negatively affects the accuracy of the flat classifiers, the same result is observed

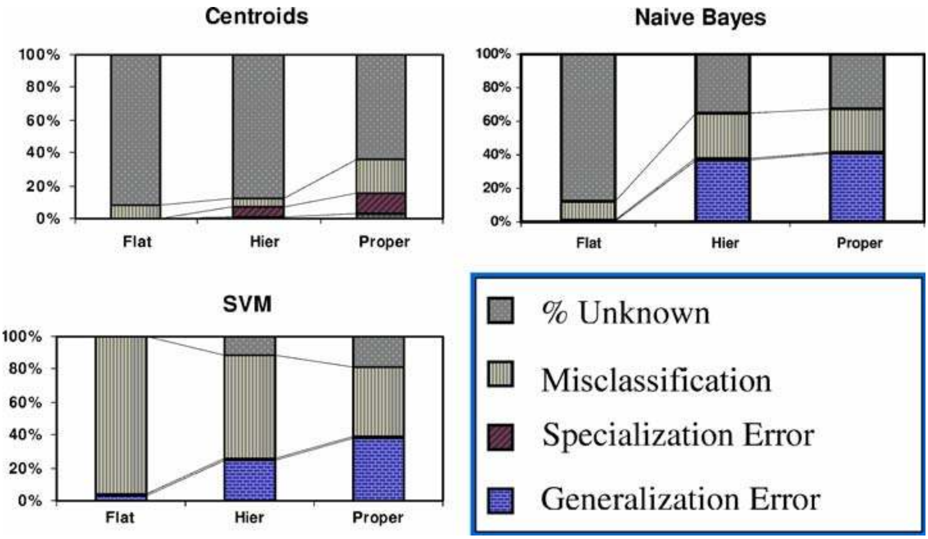


**Fig. 5** Distribution of errors for Reuters dataset ($n_{dict}$=60)

for hierarchical classifiers. For opposite reasons, the trend of flat classifiers is less correlated to that of hierarchical classifiers with proper feature sets.

From a closer analysis of the percentage of errors (reject, misclassification, generalization and specialization) performed by the various classifiers (see Figs. 5, 6 and 7), we observe that the flat classifiers commit more rejection and misclassification errors (in percentage) than the corresponding hierarchical classifiers. Therefore, with reference to the second question, we conclude that, *even though SVM or centroid-*

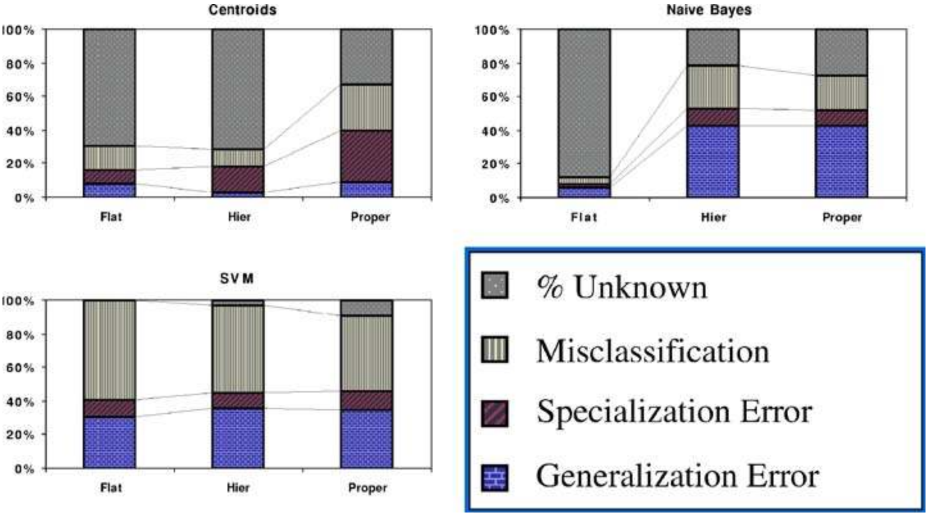


**Fig. 6** Distribution of errors for Yahoo dataset ($n_{dict}$=60)

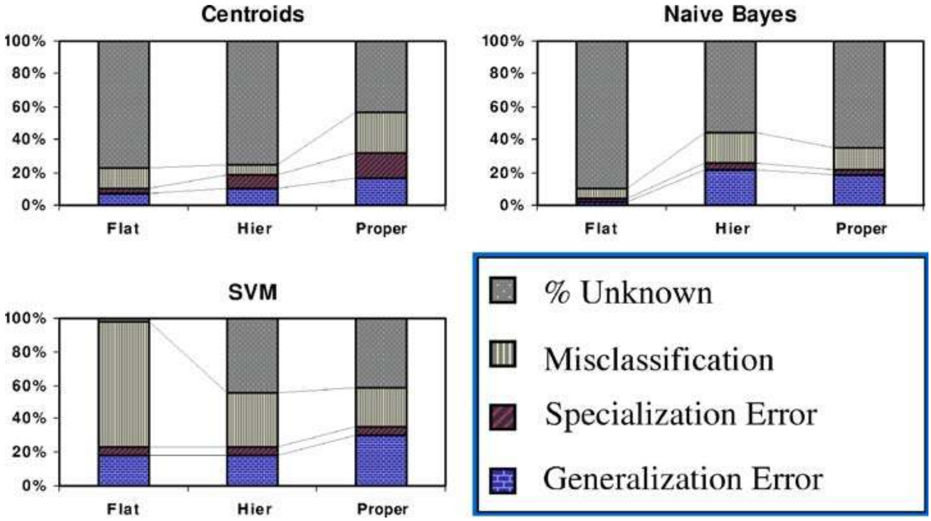


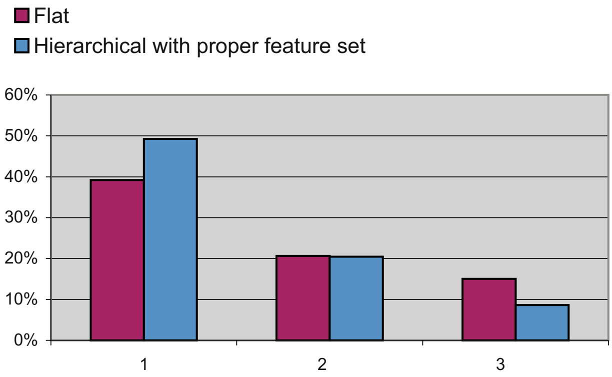
**Fig. 7** Distribution of errors for Dmoz dataset ($n_{dict}$=20)

*based flat classifiers are more accurate than the corresponding hierarchical classifiers, they tend to commit "more serious" errors.*

This difference in error type is particularly significant for NB classifiers. Figure 8 shows the distribution of misclassification, specialization and generalization errors with respect to the (tree) distance of the wrong category from the correct one. The represented statistics refer to the Dmoz dataset, which is the most complex in terms of number of categories and depth of the hierarchy. In general, errors are distributed quite "close" to the correct category, also thanks to the automated threshold definition algorithm that minimizes the sum of tree distances between the correct and the predicted categories. Nevertheless, results are better for the hierarchical classifier, since the distribution is more skewed towards low distance values.

To answer the question on the actual improvement of the computational efficiency of the learning algorithms, we collected statistics on the running time (see Fig. 9).

**Fig. 8** Distribution of errors. Percentage of misclassification, specialization and generalization errors classified at distance 1, 2 and 3 from the correct class. Statistics for larger distances are not shown. Results are obtained on the Dmoz dataset, with Naïve Bayes classifier, feature set size = 20



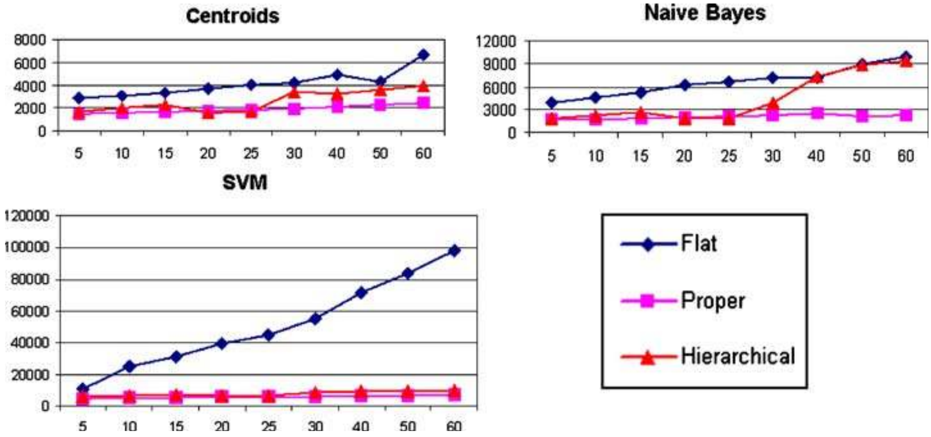


**Fig. 9** Learning running times on the RCV1 dataset. Results are expressed in seconds varying the number of selected features. Results show the comparison between the flat technique, hierarchical with a proper feature set and hierarchical with a hierarchical feature set. WebClassIII has been executed on a Pentium 4 PC 1.4 GHz running a Windows 2000 operating system

*Results are substantially in favor of the hierarchical framework*. The difference is particularly evident in the case of the SVM classifier. This confirms the analysis of complexity reported in Section 5.4.

The results also show the better performances of the hierarchical framework with a proper feature set, with respect to the hierarchical framework with a hierarchical feature set. This also confirms the formal analysis of complexity reported in Section 5.4 and, in particular, the role of the number of features, which grows proportionally to the total number of classes in the case of a hierarchical feature set.

### 6.4 Comparing hierarchical classifiers

In the previous section we answered questions on the pros and cons of hierarchical classifiers when compared to flat classifiers. In this section, we investigate aspects specifically related to the hierarchical classifiers, namely, which is the best strategy for feature selection and what is the best classifier to use in combination with the hierarchical categorization framework.

From the results shown in Fig. 4 we observe that while the accuracy of hierarchical classifiers built on proper feature sets remains constant or increases with the addition of new features, the inverse trend is shown in the case of hierarchical feature sets. This is caused by the fact that the size of feature sets may remarkably increase for high-level categories up to loosing the property of ‘orthogonality.’ This phenomenon is particularly evident in the case of the centroid-based classifiers. As regards SMVs and NB, the hierarchical approach performs better than the proper approach for smaller feature sets, while there is an asymptotic convergence for larger feature sets. This can be explained by the fact that, with a limited number of features, the lower categories are not represented and it is necessary to use a hierarchical feature set. By increasing the number of features, the deeper categories are better represented and the benefits of a hierarchical approach vanish.

For the comparison of hierarchical classifiers, we limit our study to proper feature sets. Once again, several feature sets of different sizes have been extracted for each internal category, in order to study the effect of this factor on the classifier performance. Sizes range from 5 to 60 features per category in the case of the RCV1 and the Yahoo dataset, while it ranges from 5 to 40 in the case of Dmoz dataset. Collected statistics concern centroid-based, NB and SVM classifiers.

Figures 10, 11 and 12 show the performances of different classifiers for each document collection and for different sizes of the proper feature sets. For the RCV1 and the Dmoz datasets, which are characterized by a complex hierarchy (both in the number of categories and in the depth of the tree structure), the best results in terms of accuracy are obtained by the SVM classifier, while for the Yahoo dataset, the naïve Bayes classifier performs best for sufficiently large feature sets. The centroid-based

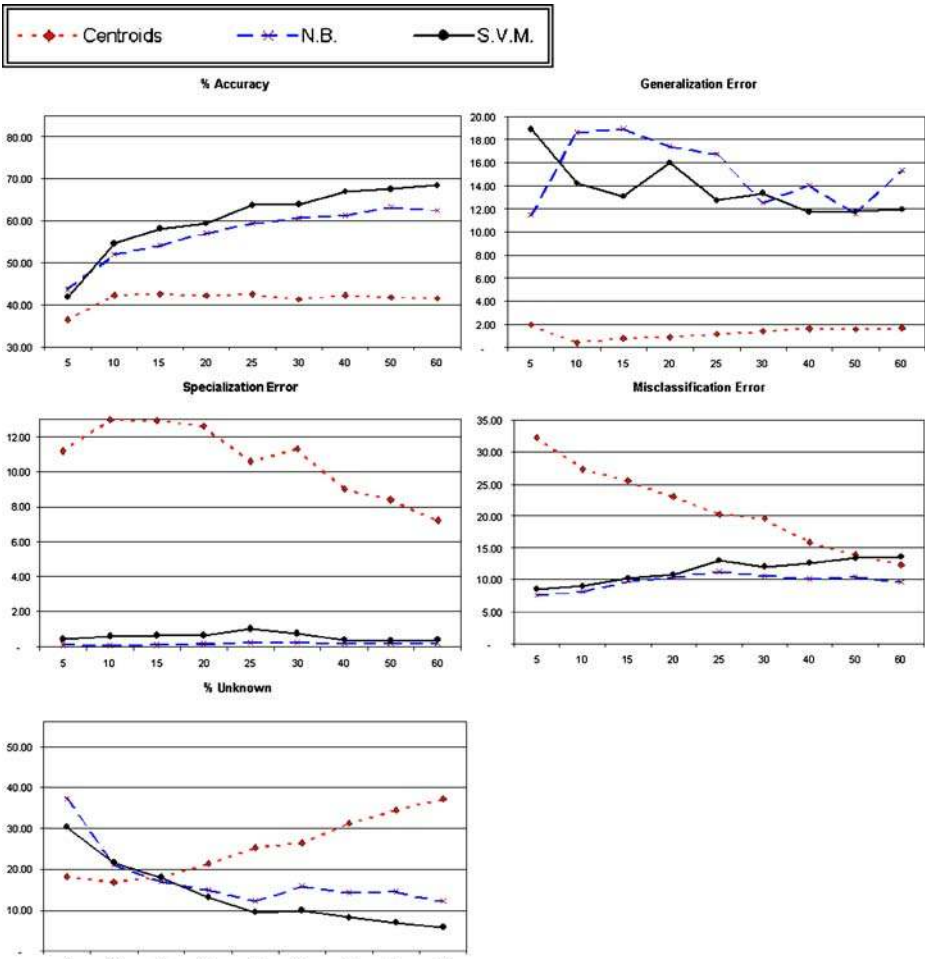


**Fig. 10** Classifier comparison on the RCV1 collection. Features are extracted using proper feature sets

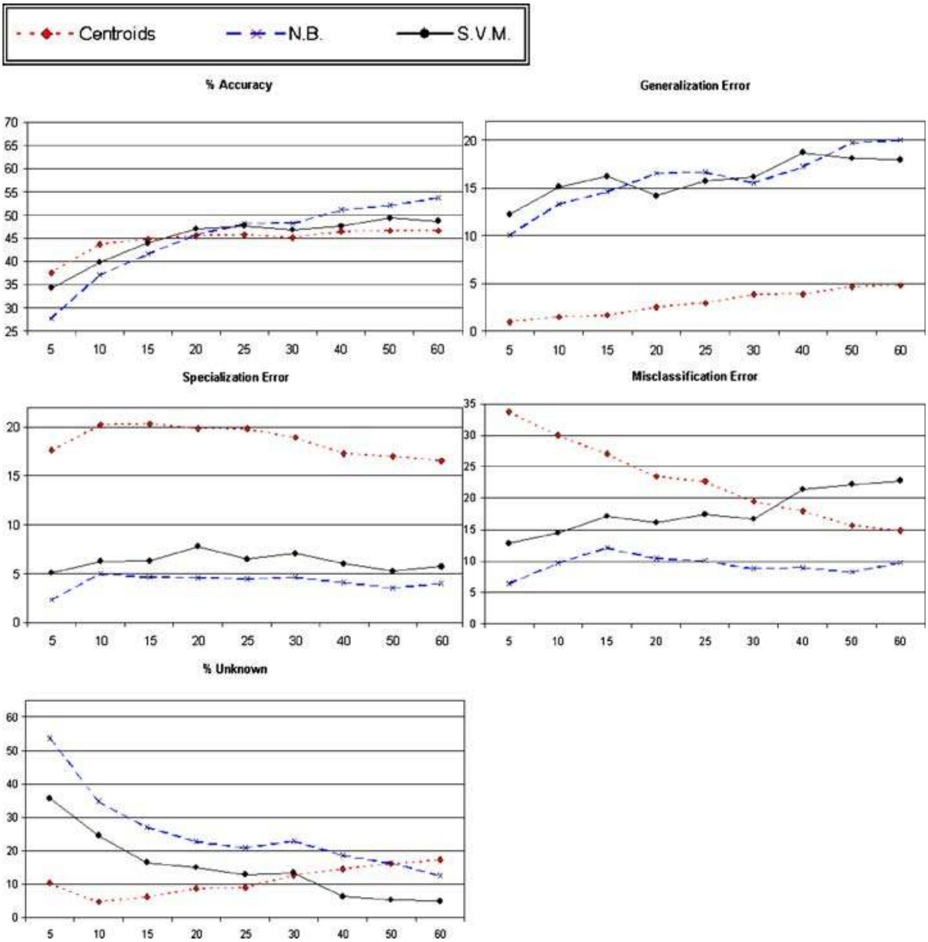


**Fig. 11** Classifier comparison on the Yahoo collection. Features are extracted using proper feature sets

classifier shows the worst performance, particularly when the size of the feature set increases.

Looking at the errors committed in detail, it is interesting to note that:

- *NB and SVM show the same trend, which is different from the trend of centroids.* For example, while for SVM and NB the specialization error is low and the generalization error tends to be quite high, the situation is reversed for centroids.
- *Increasing the number of the features, the percentage of misclassifications for NB and SVM increases, while the percentage of "rejected" documents (unknown error) decreases. This behavior is reversed for centroids.*

The different behaviour can be explained by the fact that the thresholding algorithm tends to be generally conservative (i.e., high thresholds and few documents passed down) for SVM and NB, while in the case of centroids the thresholds become

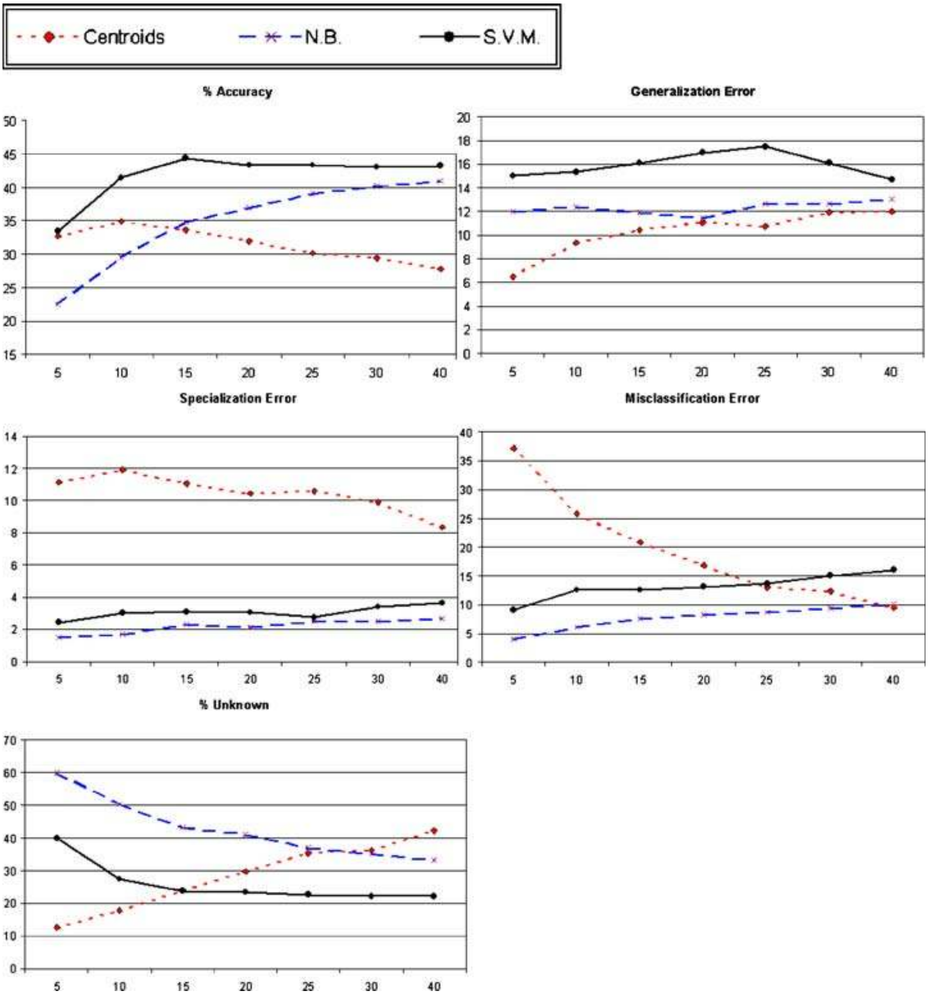


**Fig. 12** Classifier comparison on the Dmoz dataset. Features are extracted using a proper feature set

more selective only for larger feature sets. Indeed, the scores $\gamma_{c \to c'}(d)$ computed by centroid-based classifiers are unevenly distributed at the extremes of the unit interval when only a few features determine the result of the classification. In this situation of binary-like classification, the thresholding algorithm cannot work properly. On the contrary, the scores are less extreme in large feature spaces and the thresholding algorithm can work properly by reducing the high number of misclassifications, at the cost of increasing the rejection rate.

Comparing NB and SVM it is noticeable that SVM has a higher misclassification rate, while NB has a higher rejection rate. This means that even when they do not perform best, NB classifiers can be a valid alternative to SVM in those application contexts where a "commission error" is considered more serious than an "omission error."

## 7 Related work in hierarchical text categorization

In the literature, several approaches have been proposed that face the problem of hierarchical document categorization. They differ in terms of several aspects that principally involve the document representation and the learning strategy. As for the document representation, each document can be described by several sets of features, each of which is useful for the classification of the 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, Blockeel, Bruynooghe, Dzeroski, Ramon and Struyf (2002) defined a specific decision tree induction algorithm for the case of hierarchical multi-category classification. 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.

Some of these aspects have been considered in related works. In particular, in the seminal work by Koller and Sahami (1997) 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 (Sahami, 1996). A distinct 1-of-r 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 the new document should be sent. 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. (1998) proposed a method based on the naïve Bayes learner. 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 mutual information at each internal node of the tree, using the node's immediate children as classes. Because of the uniqueness of the feature set, Bayesian classifiers associated at internal nodes are *homogeneous*, and, as formalized by Mitchell (1998), the hierarchical organization of homogeneous classifiers is equivalent to a single flat classifier. In other words, the hierarchical structure would have no practical impact on the classification process. This explains why, in the learning step, McCallum et al. use a statistical technique known as *shrinkage* to smooth parameter estimates for lower-level categories with parameter estimates for their ancestors in the category hierarchy. For the classification step, the authors compare two techniques: exploring all possible paths in the hierarchy and greedily selecting the most probable one/two branches as done by Koller and Sahami (1997). Results show that greedy selection is not only 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.

Mladenić (1998b) 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, a document can be assigned to any node of the hierarchy.

In the work by D'Alessio et al. (2000) 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-r 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 (2000) use the hierarchical structure for two purposes. First, it is for training several SVMs, 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.[15] Second, it is for classifying documents by combining scores from SVMs 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 with respect to flat models.

In the system CLASSI by Ng et al. (1997), the hierarchical classification of documents is obtained by combining several linear classifiers according to a tree structure (hierarchical classifier). The tree structure corresponds to the hierarchy of categories, which means that a linear classifier is associated to each category. The output of the classifier defines a degree of membership of a document to a category. In the classification phase, the hierarchical classifier receives a document and checks whether it belongs to any of the first level nodes. If the tested document activates any of the first level nodes, then the descendant categories of that node are tested recursively. Multiple classifications of documents are allowed, while the classification of documents in non-leaf categories seems not to be supported. Weights of each linear classifier are determined by means of the perceptron learning algorithm. The

[15] Note that differently from D'Alessio et al. (2000), 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.

training set of each linear classifier includes all positive documents of the associated category (i.e., no hierarchical training set) and some selected documents of other categories. Two peculiarities of this work are the use of WordNet (Miller, 1990) to replace each word with its morphological root form and the use of the correlation coefficient to select the best subset of words. However, $F_1$-scorevalues reported on the Reuters dataset are well below those reported by (Yang, 1999) on the same dataset.

In the work by Ruiz and Srinivasan (2002) a variant of the Hierarchical Mixture of Experts (HME) model is used. A tree of backpropagation neural networks is used. Neural networks are of two types: experts and gates. The former take the feature-vector representation of a document as input and are trained to recognize whether the document belongs to a specific category. There are as many experts at the leaves of this tree-structured classifier as categories (leaves and non-leaves) in the hierarchy. Gating networks are the internal nodes of the tree-structured classifier and map the non-leaf categories of the hierarchy. They have two kinds of input: the feature-vector representation of a document and the output of the expert/gating networks below in the tree. Their role is that of restricting the number of experts to be activated for a given document. Indeed, the classification of a document proceeds top-down in the tree of neural networks, starting from the gate at the root towards the experts at the leaves. Multiple classification is supported. The gates are trained to recognize whether any of the categories of their descendants is present or not in the document. The experts are trained to recognize the presence or absence of particular categories. Therefore, the set of positive examples for an expert includes documents of the uniquely associated category while the set of positive examples for a gate includes all training documents of the set of associated categories. Some form of filtering is used for negative examples, since unbalanced data sets may affect the learning capability of backpropagation neural networks. Different feature sets are selected for each expert and gating network. The proposed method is tested on some MEDLINE records. Only categories with positive examples are selected, since this method cannot work when intermediate categories have no positive examples.

A hierarchical classifier combining several neural networks is also proposed by Weigend et al. (1999). Neural networks at internal nodes are "meta-topic classifier," while those at the leaves are "individual classifiers." The method has been devised and tested only on two-level hierarchies, although the extension to more than two levels should be straightforward. The dimensionality reduction of the original feature space is obtained by means of two statistical techniques: Latent Semantic Indexing, to transform the original feature set into a new set of features that are a linear combination of the original features, and $\chi^2$ statistics to select the most discriminant features. Moreover, selected feature sets can either be specific for each category or unique for all categories. The former gave a better performance on the Reuters dataset, thus empirically confirming Mitchell's finding (Mitchell, 1998) also for classifiers based on neural networks.

A summary of the referenced papers is reported in Table 1. We are aware that the list of related works summarized in the table is not exhaustive, although it is representative of the most well-known contributions. For the sake of completeness, we report a brief note on three additional works. Sun and Lim (2001) have proposed the use of category-similarity measures and distance-based measures to consider the

**Table 1** Classification of previous works

| Work | Hierarchy | Feature sets | Feature selection | Learning | Training set | Classification |
|---|---|---|---|---|---|---|
| Koller and Sahami (1997) | Doc.s only at the leaves | A separate feature set for each category | Probabilistic approach | Naive Bayes & KDB.<br>A *1-of-r* classifier for each internal node | One hierarchical training set per category | Greedy search of a single classification path.<br>Single category assignment |
| McCallum et al. (1998) | Doc.s only at the leaves | A unique feature set built from category vocabularies | Mutual information | Shrinkage + *1-of-r* naive Bayes classifier. Parameters estimated for each category. | Single set | Both greedy and extensive search of classification paths.<br>Single category assignment. |
| Mladenić (1998b) | Doc.s at any node | A separate feature set for each category | Several measures tested | *1-of-r* naive Bayes classifier | One hierarchical training set per category | Extensive search with pruning.<br>Single class assignment |
| D'Alessio et al. (2000) | Doc.s only at the leaves | A positive and negative feature set per category | A variant of the ACTION algorithm | Feature weight estimation.<br>Both binary and *1-of-r* classifier | One set per category.<br>Pos.: docs of the category<br>Neg.: docs of the parent category | Both greedy and extensive search with pruning.<br>Single or multiple category assignment |
| Dumais and Chen (2000) | Doc.s only at the leaves | A separate feature set for each category. | Mutual information | Binary SVM classifier | One set per hierarchy level, with docs of all categories at the same level | Extensive search with pruning.<br>Multiple category assignment |
| Ng et al. (1997) | Doc.s only at the leaves | A separate feature set for each category | Correlation coefficient | Binary Perceptron-based classifier | One set per category.<br>Pos.: docs of the category<br>Neg.: some selected docs | Extensive search.<br>Multiple category assignment |
| Ruiz and Srinivasan (2002) | Doc.s at any node | A separate feature set for each category | Correlation coefficient,<br>Mutual information<br>Odds ratio | Neural Networks for binary classification | One set per category.<br>Pos.: docs of the category<br>Neg.: some selected docs | Extensive search.<br>Multiple category assignment |
| Weigend et al. (1999) | Doc.s only at the leaves | Both separate and unique feature set | LSI and $\chi^2$ | Neural Networks for binary classification | One hierarchical training set per category | Extensive search.<br>Multiple category assignment |
| This work WebClassIII | Doc.s at any node. Internal nodes without docs and single-child are allowed | A separate feature set (hierarchical or proper) for each internal category | $maxTF \times DF^2 \times IC$ | *1-of-r* naive Bayes, centroid-based and SVM-based classifiers.<br>Automatic threshold definition | One hierarchical training set per category | Greedy search of a single classification path.<br>Single category assignment |

degree of misclassification in measuring the classification performance. Experiments were performed on the Reuters-22173 collection with SVM$^{light}$ Version 3.50 implemented by Joachims (1998a). Chuang, Tiyyagura, Yang and Giuffrida (2000) have tested a Rocchio-based classifier on a collection of approximately 200 documents on professional baseball and basketball news. Finally, Tikk and Biro (2003) tested a centroid-based classifier on the WIPO-alpha[16] English patent database that consists of about 75,000 XML documents distributed over 5,000 categories in four levels. Unfortunately, studies on the WIPO-alpha collection are not publicly available because of the strongly business sensitive nature of the research. As future work, we plan to extend our experimental results to this dataset as well.

### 7.1 Comparison with related work: The method

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 explicitly considered and tested only in (Mladenić, 1998b) and (Ruiz & Srinivasan, 2002). However, unlike Mladenić'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. Other special conditions that are considered in this work are: 1) no document for some internal nodes; 2) some internal nodes have only one child.

A second difference is in the feature selection process for each internal category. In WebClassIII it is based on an upgrade of the technique implemented and tested by Malerba et al. (2002), named $maxTF \times DF^2 \times ICF$. Unlike other feature selection methods proposed in the literature on hierarchical document categorization (Mladenić & Grobelnik, 2003), $maxTF \times DF^2 \times ICF$ answers the demand for terms that are shared by most of the documents of the same category and possibly no document of other categories. Moreover, it considers the document (and not a term) as an observation unit.

A third difference is that we do not propose a specific method, but we investigate a framework for hierarchical text categorization that can be applied to any classifier that returns a degree of membership (e.g., distance or probability based) of a document to a category. We applied the framework to three classifiers, two of which present some variants with respect to the original methods reported in the literature.

The fourth difference is in the development of a technique for the automated selection of thresholds for the degree of membership returned by the classifier. The thresholds are used to determine whether a document has to be passed down to one of the child categories during the top-down classification process.

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

We conclude by observing that the main contribution of this work is the systematic investigation of the usage of information provided by the category hierarchy in

[16] World Intellectual Property Organization, Geneva, Switzerland, 2002 - http://www.wipo.int/.

all aspects of text categorization, such as definition of training sets, feature sets, classifiers, threshold-based document classification and evaluation measures.

### 7.2 Comparison with related work: experimental results

Previous studies on hierarchical text categorization have already contributed to clarifying some aspects that have not been explored in this work. Koller and Sahami (1997) experimentally showed that there is a substantial improvement in accuracy when feature selection is aggressively employed versus the case where all domain features are used. This improvement has been observed both in the hierarchical case and in the flat case for Bayesian classifiers. McCallum et al. (1998) show that aggressive feature selection is not necessary if shrinkage is used to smooth parameter estimates. Shrinkage helps especially when training data are sparse, which is the case when small sets of documents are assigned to leaf categories. Mladenić (1998b) compared six feature selection techniques for automatic document categorization, based on text hierarchies and her conclusions were in favor of Odds ratio when combined with a naive Bayes classifier. D'Alessio et al. (2000) investigated the possibility of restructuring a pre-existing hierarchy, and concluded that the usage of a hierarchy, either modified or built from scratch, can significantly improve both the speed and effectiveness of the categorization process. Dumais and Chen (2000) explored two ways to combine probabilities returned by the classifiers for the first and second level of a two-level hierarchy. The multiplicative approach assigns the document to a leaf if the product of both probabilities exceeds a given threshold, which is unique for all categories. The Boolean approach assigns the document to a leaf if the threshold is exceeded at every level. No difference between the two approaches was observed in terms of F1 measure, hence leading to the recommendation for the Boolean approach which is the most efficient. Ruiz and Srinivasan (2002) reported good results for the (flat) Rocchio classifier when both training data and features are selected, and categories have a medium/high number ($\geq 15$) of training examples. Results reported by Weigend et al. (1999), who observed that the largest gains in average precision for the hierarchical classifier concern "rare" (i.e., with few training examples) categories, are also consistent with Ruiz and Srinivasan's findings. The main difference between the two findings is that in the work by Ruiz and Srinivasan, rare categories can occur at any node in the hierarchy, while in the work by Weigend et al. they are always leaf categories.

As to the real advantages of the hierarchical vs. flat approach, no conclusive result has been reported for predictive accuracy. Koller and Sahami (1997) observed that the hierarchical approach appears to provide few benefits when attention is restricted to simple classifiers, such as naïve Bayes. Dumais and Chen (2000) reported minor improvements for hierarchical models over flat models. Similarly, Ruiz and Srinivasan (2002) do not show a clear superiority of the HME with respect to Rocchio. On the contrary, McCallum et al. (1998) demonstrate that shrinkage with a class hierarchy significantly reduces the classification error, Ng et al. (1997) report accuracy improvements of the hierarchical method with respect to the flat method, and Weigend et al. (1999) attribute a statistically significant overall improvement of 5% for averaged precision to the hierarchical approach. This confirms our experimental observation that there is an interaction, in terms of accuracy, between the hierarchical framework and the type of classifier.

All related works examined here show the clear computational advantage of the hierarchical approach. We have confirmed this conclusion both analytically and experimentally. This work, however, presents additional empirical findings not reported elsewhere. They are summarized in the following points:

1. Among flat classifiers, SVM performs the best across the three datasets.
2. Even though SVM or centroid-based flat classifiers are more accurate than the corresponding hierarchical classifiers, they tend to commit "more serious" errors ("severity" is based on a tree-distance measure).
3. As the number of features increases, the classifier trained with a proper feature set asymptotically tends to the performances of the classifier trained with a hierarchical feature set.
4. Errors committed by NB and SMV show the same trend, which is different from the trend of centroids.
5. Increasing the number of the features, the percentage of misclassifications for NB and SVM increases, while the percentage of "rejected" documents (unknown error) decreases. This behavior is reversed for centroids.

All these results, which extend those reported in a previous work (Ceci & Malerba, 2003), are obtained by extensive experimentation on three datasets with category hierarchies of different complexity.

## 8 Conclusions and future work

Most of the research on text categorization has focused on classifying text documents into a set of categories with no structural relationships among them. However, in this case it is difficult to browse or search documents in a large number of categories. Hierarchies are often used to make large collections of document categories more manageable, since they permit the application of the well-known principle of divide-and-conquer. The hierarchical structure is employed in many Internet directories (e.g., Yahoo and Google Directory) and in text databases (e.g., MEDLINE and patent databases), as well as in other document management tools (e.g., Netscape Bookmark). Therefore, whether and how to exploit the additional information on the hierarchical structure among categories in text categorization is an important issue that demands systematic investigation.

Our research adds to a growing body of work exploring how hierarchical structures can be used to improve the efficiency and efficacy of text classification. We have presented and evaluated a hierarchical text categorization framework that involves the hierarchy of categories in all phases of text categorization, namely feature extraction, learning, and classification of a new document. Our conclusion is that for large collections of documents organized in complex hierarchies, the hierarchical approach can offer two main advantages: efficiency gain and reduction of severity of classification errors. The former is particularly important when the hierarchy of categories is subject to changes, since in the flat approach changes affect all classifiers, while in the hierarchical approach they are all localized. The latter advantage is quite important if a trained user cannot supervise decisions taken by the document classifier.

Although we observed good results for the flat SVM across all three datasets used in our experimental validation of the framework, in the hierarchical approach the naive Bayes classifiers, built with proper feature sets, seem to be a valid alternative to SVM, especially in those application contexts where a "commission error" is considered more serious than an "omission error."

In this work we have not investigated the possibility of restructuring the original category hierarchy. Vinokourov and Girolami (2002) proposed a probabilistic mixture model for the hierarchic partition organization of a collection of documents. Sona, Veeramachanemi, Avesani and Polettini (2004) address the problem of document clustering where documents are assigned both to the leaves and to internal nodes. An alternative to building the hierarchy from scratch is restructuring a given hierarchy on the basis of some training examples. It can be realized by means of a greedy procedure that adds or removes categories until no further improvements can be made. Hierarchy restructuring can substantially improve the accuracy of the hierarchical approach, which can eventually give better performance than the flat approach.

Another limitation of this work is the consideration of a single-category assignment rather than the more general case of a multi-category assignment. However, the multi-category assignment occurs either when the hierarchy appears to be ill-structured with respect to documents collected over time, or when the same documents can be actually classified along several dimensions. In the former case, the single-category assignment can be kept if the hierarchy is restructured. In the latter case, it would be better to consider a multi-dimensional framework, as that investigated by Theeramunkong and Lertnattee (2002). In the future, we intend to extend this work by considering the integration of both the multi-dimensional and the hierarchical frameworks, in order to support WebClass users with OLAP-like roll-up, drill-down and pivoting operations in an information retrieval context.

## References

Apté, C., Damerau, F., & Weiss, S. M. (1994). Automated learning of decision rules for text categorization. *Information Systems*, *12*(3), 233–251.

Bennett, P. (2000). Assessing the calibration of Naive Bayes' posterior estimates. CMU-CS-00-155. Technical report, School of Computer Science, Carnegie-Mellon University.

Blockeel, H., Bruynooghe, M., Dzeroski, S., Ramon, J., & Struyf, J. (2002). Hierarchical multi-classification. In S. Džeroski, L. de Raedt & S. Wrobel (Eds.), *Multi-Relational Data Mining 2002* (pp. 21–35). Edmonton, Canada: University of Alberta.

Ceci, M., & Malerba, D. (2003). Hierarchical classification of HTML documents with WebClassII. In F. Sebastiani (Ed.), *Proceedings of ECIR-03, 25th European Conference on Information Retrieval* (pp. 57–72). Berlin Heidelberg NewYork: Springer.

Chuang, W. T., Tiyyagura, A., Yang, J., & Giuffrida, G. (2000). A fast algorithm for hierarchical text classification. In *DaWaK 2000: Proceedings of the Second International Conference on Data Warehousing and Knowledge Discovery* (pp. 409–418). Berlin Heidelberg New York: Springer.

Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., & Slattery, S. (2000). Learning to construct knowledge bases from the world wide web. *Artificial Intelligence*, *118*(1-2), 69–113.

D'Alessio, S., Murray, K., Schiaffino, R., & Kershenbau, A. (2000). The effect of using hierarchical classifiers in text categorization. In *Proc. of the 6th Int. Conf. on "Recherche d'Information Assistée par Ordinateur" (RIAO)*, (pp. 302–313). Paris, France.

Debole, F., & Sebastiani, F. (2003). Supervised term weighting for automated text categorization. In *Proceedings of SAC-03, 18th ACM Symposium on Applied Computing* (pp. 784–788). New York: ACM.

Domingos, P., & Pazzani, M. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, *29*(2-3), 103–130.

Dumais, S., & Chen, H. (2000). Hierarchical classification of web content. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 256–263). New York: ACM.

Dumais, S., Platt, J., Heckerman, D., & Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. In *Proceedings of ACM-CIKM98* (pp. 148–155). New York: ACM.

Esposito, F., Malerba, D., Tamma, V., & Bock, H.-H. (2000). Classical resemblance measures. In H.-H. Bock & E. Diday (Eds.), Analysis of Symbolic Data. Exploratory methods for extracting statistical information from complex data, volume 15 of *Studies in Classification, Data Analysis, and Knowledge Organization* (Chapter 8.1, pp. 139–152). Berlin Heidelberg New York: Springer.

Eyheramendy, S., Lewis, D., & Madigan, D. (2003). The Naive Bayes model for text categorization. In C. M. Bishop & B. J. Frey (Eds.), *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, Jan 3-6, Key West, Florida

Galavotti, L., Sebastiani, F., & Simi, M. (2000). Experiments on the use of feature selection and negative evidence in automated text categorization. In *ECDL '00: Proceedings of the 4th European Conference on Research and Advanced Technology for Digital Libraries* (pp. 59–68). Berlin Heidelberg New York: Springer.

Han, E.-H., & Karypis, G. (2000). Centroid-based document classification: Analysis and experimental results. In *PKDD '00: Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery* (pp. 424–431). Berlin Heidelberg New York: Springer.

Hastie, T., & Tibshirani, R. (1998). Classification by pairwise coupling. In *NIPS '97: Proceedings of the 1997 conference on Advances in neural information processing systems 10* (pp. 507–513). Cambridge, Massachussetts: MIT.

Joachims, T. (1997). A probabilistic analysis of the Rocchio algorithm with tfidf for text categorization. In *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning* (pp. 143–151). San Mateo, California: Morgan Kaufmann.

Joachims, T. (1998a). $SVM^{light}$, an implementation of Support Vector Machines (SVMs) in C http://ais.gmd.de/thorsten/svmlight/.

Joachims, T. (1998b). Text categorization with support vector machines: Learning with many relevant features. In *ECML '98: Proceedings of the 10th European Conference on Machine Learning* (pp. 137–142). Berlin Heidelberg New York: Springer.

Kim, S., Rim, H., Yook, D., & Lim, H. (2002). Effective methods for improving Naive Bayes text classifier. In *7th International Conference on Artificial Intelligence*, volume 2417 of *LNAI* (pp. 95–106). Berlin Heidelberg New York: Springer.

Koller, D., & Sahami, M. (1997). Hierarchically classifying documents using very few words. In *Proceedings of the Fourteenth International Conference on Machine Learning* (pp. 170–178). San Mateo, California: Morgan Kaufmann.

Lertnattee, V., & Theeramunkong, T. (2004). Effect of term distributions on centroid-based text categorization. *Information Science—Informatics and Computer Science*, *158*(1), 89–115.

Lewis, D. D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval. In *ECML '98: Proceedings of the 10th European Conference on Machine Learning* (pp. 4–15). Berlin Heidelberg New York: Springer.

Lewis, D. D., Yang, Y., Rose, T. G., & Li, F. (2004). RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, *5*, 361–397.

Malerba, D., Esposito, F., & Ceci, M. (2002). Mining HTML pages to support document sharing in a cooperative system. In *EDBT '02: Proceedings of the Worshops XMLDM, MDDE, and YRWS on XML-Based Data Management and Multimedia Engineering-Revised Papers* (pp. 420–434). Berlin Heidelberg New York: Springer.

McCallum, A., & Nigam, K. (1998). A comparison of event models for Naive Bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization* (pp. 41–48). Menlo Park California: AAAI.

McCallum, A., Rosenfeld, R., Mitchell, T. M., & Ng, A. Y. (1998). Improving text classification by shrinkage in a hierarchy of classes. In *Proceedings of the Fifteenth International Conference on Machine Learning* (pp. 359–367). San Mateo, California: Morgan Kaufmann.

Miller, G. (1990). Five papers on Wordnet. *International Journal of Lexicology*, 3(4), 278–301.

Mitchell, T. (1997). *Machine Learning*. New York: McGraw-Hill.

Mitchell, T. (1998). Conditions for the equivalence of hierarchical and flat Bayesian classifiers. Technical report, Center for Automated Learning and Discovery, Carnegie-Mellon University.

Mladenić, D. (1998a). Feature subset selection in text-learning. In *ECML '98: Proceedings of the 10th European Conference on Machine Learning* (pp. 95–100). Berlin Heidelberg New York: Springer-Verlag.

Mladenić, D. (1998b). *Machine learning on non-homogeneus, distributed text data*. PhD thesis, University of Ljubjana, Slovenia.

Mladenić, D., & Grobelnik, M. (1999). Feature selection for unbalanced class distribution and Naive Bayes. In *ICML '99: Proceedings of the Sixteenth International Conference on Machine Learning* (pp. 258–267). San Mateo, California: Morgan Kaufmann.

Mladenić, D., & Grobelnik, M. (2003). Feature selection on hierarchy of web documents. *Decision Support Systems*, *35*(1), 45–87.

Ng, H. T., Goh, W. B., & Low, K. L. (1997). Feature selection, perception learning, and a usability case study for text categorization. *SIGIR Forum*, *31*(SI), 67–73.

Platt, J. (1998). Fast training of Support Vector Machines using sequential minimal optimization. In B. Scholkopf, C. Burges & A. Smola (Eds.), *Advances in Kernel methods – support vector learning*. MIT Press.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, *14*(3), 130–137.

Rocchio, J. (1971). Relevance feedback in information retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*(pp. 313–323). Englewood Cliffs: Prentice Hall.

Ruiz, M. E., & Srinivasan, P. (2002). Hierarchical text categorization using neural networks. *Information Retrieval*, *5*(1), 87–118.

Sahami, M. (1996). Learning limited dependence Bayesian classifiers. In *Second International Conference on Knowledge Discovery in Databases* (pp. 334–338). Menlo Park, California: AAAI.

Salton, G., & Buckley, C. (1988). Term weighting approaches in automatic text retrieval. *Information Processing and Management*, *24*(5), 513–523.

Schapire, R. E., & Singer, Y. (2000). Boostexter: A boosting-based system for text categorization. *Machine Learning*, *39*(2-3), 135–168.

Schapire, R. E., Singer, Y., & Singhal, A. (1998). Boosting and Rocchio applied to text filtering. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 215–223). New York: ACM.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, *34*(1), 1–47.

Shen, Y., & Jiang, J. (2003). Improving the performance of Naive Bayes for text classification, CS224N spring. Technical report, Stanford University.

Sona, D., Veeramachanemi, S., Avesani, P., & Polettini, N. (2004). Clustering with propagation for hierarchical document classification. In M. Gori, M. Ceci & M. Nanni (Eds.), *Proceedings of the ECML/PKDD'04 Workshop on Statistical Approaches for Web Mining* (pp. 50–61). Pisa, Italy.

Sun, A., & Lim, E.-P. (2001). Hierarchical text classification and evaluation. In *ICDM '01: Proceedings of the 2001 IEEE International Conference on Data Mining* (pp. 521–528). Los Alamitos, California: IEEE Computer Society.

Theeramunkong, T., & Lertnattee, V. (2002). Multi-dimensional text classification. In *Proc. of 19th International Conference on Computational Linguistics (COLING 2002)* (pp. 1–7). Morristown, New Jersey: Association for Computational Linguistics.

Tikk, D., & Biro, G. (2003). Experiment with a hierarchical text categorization method on the WIPO-alpha patent collection. In *ISUMA '03: Proceedings of the 4th International Symposium on Uncertainty Modelling and Analysis* (p. 104). Los Alamitos, California: IEEE Computer Society.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Berlin Heidelberg New York: Springer.

Vinokourov, A., & Girolami, M. (2002). A probabilistic framework for the hierarchic organisation and classification of document collections. *Journal of Intelligent Information System*, *18*(2-3), 153–172.

Weigend, A. S., Wiener, E. D., & Pedersen, J. O. (1999). Exploiting hierarchy in text categorization. *Information Retrieval*, *1*(3), 193–216.

Yang, Y. (1996). An evaluation of statistical approaches to MEDLINE indexing. In *Proceedings of the AMIA* (pp. 358–362). Philadelphia, Pennsylvania: Hanley and Belfus.

Yang, Y. (1999). An evaluation of statistical approaches to text categorization. *Information Retrieval*, *1*(1-2), 69–90.

Yang, Y., & Liu, X. (1999). A re-examination of text categorization methods. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 42–49). New York: ACM.

Yang, Y., & Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. In *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning* (pp. 412–420). San Mateo, California: Morgan Kaufmann.

Zhang, J., Jin, R., Yang, Y., & Hauptmann, A. G. (2003). Modified logistic regression: An approximation to SVM and its applications in large-scale text categorization. In *Proceedings of the 20th International Conference on Machine Learning* (pp. 888–895). Menlo Park, AAAI Press.

Zheng, Z., Wu, X., & Srihari, R. (2004). Feature selection for text categorization on imbalanced data. *SIGKDD Explorations Newsletter*, *6*(1), 80–89.