Learning and Evaluation in the Presence of Class Hierarchies: Application to Text Categorization

Kiritchenko, Svetlana; Famili, Fazel; Matwin, S.; Nock, R.

National Research Council Canada
Conseil national de recherches Canada

Institute for Information Technology
Institut de technologie de l'information

NRC · CNRC

*Learning and Evaluation in the Presence of Class Hierarchies: Application to Text Categorization **

Kiritchenko, S., Famili, F., Matwin, S., and Nock, R.
June 2006

* published in the Proceedings of the 19th Canadian Conference on Artificial Intelligence. Québec City, Québec, Canada. June 7-9, 2006. NRC 48737.

Canada

Learning and Evaluation in the Presence of Class Hierarchies: Application to Text Categorization

Svetlana Kiritchenko[1,4], Stan Matwin[1,2], Richard Nock[3], and A. Fazel Famili[4]

[1] University of Ottawa, Canada
{svkir, stan}@site.uottawa.ca
[2] Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland
[3] Université Antilles-Guyane, Martinique, France
rnock@martinique.univ-ag.fr
[4] Institute for Information Technology, National Research Council Canada
Fazel.Famili@nrc-cnrc.gc.ca

Abstract. This paper deals with categorization tasks where categories are partially ordered to form a hierarchy. First, it introduces the notion of consistent classification which takes into account the semantics of a class hierarchy. Then, it presents a novel global hierarchical approach that produces consistent classification. This algorithm with AdaBoost as the underlying learning procedure significantly outperforms the corresponding "flat" approach, i.e. the approach that does not take into account the hierarchical information. In addition, the proposed algorithm surpasses the hierarchical local top-down approach on many synthetic and real tasks. For evaluation purposes, we use a novel hierarchical evaluation measure that has some attractive properties: it is simple, requires no parameter tuning, gives credit to partially correct classification and discriminates errors by both distance and depth in a class hierarchy.

1 Introduction

Hierarchical categorization deals with categorization problems where categories (aka classes) are organized in hierarchies. More formally, categories are partially ordered, usually from more generic to more specific. The hierarchical way of organization of entities or notions is very helpful for humans to retain, find and analyze things. Therefore, it is not surprising that people maintain large collections of articles, images or emails in hierarchies of topics or systematize a large body of biological knowledge in hierarchies of concepts (aka ontologies). Such organization allows to focus on a specific level of details ignoring specialization of lower levels and generalization of upper levels.

Hierarchical categorization is an automatic approach of placing new items into a collection with a predefined hierarchical structure. In this work we focus mainly on one application area, hierarchical text categorization. However, the proposed techniques can be applied to automatic hierarchical categorization of entities of any kind. Hierarchical text categorization has many important real-world applications. In fact, most of the large textual collections are organized hierarchically, e.g. web repositories, digital libraries, patent libraries, email folders, etc. Dealing

L. Lamontagne and M. Marchand (Eds.): Canadian AI 2006, LNAI 4013, pp. 397–408, 2006.

with hierarchies effectively and efficiently is becoming a necessity in many text categorization applications.

Theoretically, hierarchical categorization can be easily substituted with "flat" categorization if we ignore the class structure and replace a hierarchy with a set of categories. However, by doing this we would disregard relevant information. For most text categorization tasks the category hierarchies have been carefully composed by humans and represent our knowledge on the subject matter. This additional information can boost the performance of a classification system if we find the way to incorporate it in the learning process.

In this work we explore two main aspects of hierarchical text categorization: learning algorithms and performance evaluation. First, we introduce the notion of consistent hierarchical classification that makes classification results even more comprehensible. In consistent classification any category label is assigned together with all its ancestor labels to any instance. Among the previously introduced hierarchical learning algorithms, only a local top-down approach produces consistent classification. We propose a new global hierarchical approach that is aimed to perform consistent classification. This is a general framework of converting a conventional "flat" learning algorithm into a hierarchical one. In our experiments we used AdaBoost as the underlying learning approach. However, any conventional method capable of performing multi-label classification can be used within this framework. Our experiments on real and synthetic data indicate that the proposed approach significantly outperforms the corresponding "flat" approach as well as the local top-down method. In addition, we design a new hierarchical evaluation measure. We argue that conventional "flat" measures as well as the existing hierarchical measures cannot discriminate between different types of errors a hierarchical classification system can make. Therefore, we propose a new hierarchical evaluation measure that is simple and straight-forward to compute, gives credit to partially correct classification and has much discriminating power.

2 Related Work

Until the mid-1990s machine learning researchers mostly ignored the hierarchical category structure present in some text categorization applications by turning a hierarchy into a flat set of categories. In 1997 Koller and Sahami carried out the first proper study of a hierarchical text categorization problem [1]. They presented a divide-and-conquer (aka local) principle, the most intuitive for hierarchical text categorization. After this work a number of approaches to hierarchical text categorization have been proposed [2, 3, 4].

Hierarchical categorization methods can be divided in two types [3]: *global* (or big-bang) and *local* (or top-down level-based). In a *global* approach only one classifier is built to discriminate all categories in a hierarchy simultaneously. It is similar to the "flat" approach except it somehow takes into account the relationships between the categories in a hierarchy. Hierarchical modifications to association rule learning [5], decision tree learning [6], SVM [7] and probabilistic learning [8] are considered global approaches.

A *local* approach builds separate classifiers for each internal node of a hierarchy. A local classifier usually proceeds in a top-down fashion first picking the most relevant categories of the top level and then recursively making the choice among the low-level categories, children of the relevant top-level categories. The local approach has been widely used with different learning algorithms: probabilistic learning [1], neural networks [4], and SVM [2].

Unlike previous work, we focus on hierarchical learning methods that build classifiers consistent with a given class hierarchy. The local approach naturally produces consistent labeling since we classify an instance into a category only if we have already classified it into the parent category in the previous classification step. However, a local classifier works only with limited (local) information at each classification node. Moreover, it is highly sensitive to the decisions made at the top of a hierarchy: once an error is committed near the top, it cannot be recovered regardless of how good the classifiers are at lower levels. A global approach, on the other hand, uses all available information at the same time and, therefore, has a better chance for correct classification. Finally, in many real-life situations, one classifier produced by a global approach is easier to maintain and to interpret by end users than a bunch of classifiers built by a local method. For these reasons, we propose a new global approach specifically designed to produce consistent classification.

3 Hierarchical Categorization Task

In this section we formally define a hierarchical classification task. We start with a definition for partial ordering, a relation present in a hierarchical structure.

Definition 1 (Poset). *A finite partially ordered set (poset) is a structure* $\mathcal{H} = \langle C, \leq \rangle$*, where* C *is a finite set and* $\leq \subseteq C \times C$ *is a reflexive, anti-symmetric, transitive binary relation on* C*.*

Given a relation $\leq$, we define a relation $<$ as $q < p$ if $q \leq p$ and $q \neq p$. For any two categories $p, q \in C$ such that $q < p$ and $\nexists r \in C : q < r < p$, we call p a parent category of q and q a child category of p. For any category $p \in C$, its ancestor set is $Ancestors(p) = \{q \in C : q \geq p\}$, and its offspring set is $Offspring(p) = \{q \in C : q \leq p\}$ (note that both sets include category p). We call categories that have no children leaves and categories that have both parents and children intermediate (or internal) classes.

Definition 2 (Hierarchical Categorization). *Hierarchical categorization task is the task of assigning a Boolean value to each pair* $\langle d_j, c_i \rangle \in D \times C$*, where* D *is a domain of instances and* $C = \{c_1, \ldots, c_{|C|}\}$ *is a set of predefined categories with a given poset structure* $\mathcal{H} = \langle C, \leq \rangle$*.*

In a hierarchical categorization task the category hierarchy $\mathcal{H} = \langle C, \leq \rangle$ describes the relations between the categories and comes from the application task at hand. The hierarchy is assumed to represent the domain knowledge and is not modified in any way.

In general, a hierarchical categorization task is *multi-label* which means that an instance can be assigned to any number of categories from 0 to $|C|$.

For any poset $\mathcal{H} = \langle C, \leq \rangle$ that represents a hierarchy we assume the existence of the root (or top) class $Root(\mathcal{H})$ that is the ancestor of all other categories in the hierarchy: $\{Root(\mathcal{H})\} = \bigcap_{p \in C} Ancestors(p)$. The root class itself has no parents.

Generally, category hierarchies are of a broader-narrower type where a subcategory represents a subtype or a part of the parent category. Category hierarchies are usually represented in the form of a directed acyclic graph (DAG). DAGs are more general than trees in that nodes in a DAG can have multiple parents.

4 Hierarchical Consistency

The notion of hierarchical consistency is intended to make the results of hierarchical classification more comprehensible for users. Since hierarchies are mostly designed in the way that lower level categories are specialization of higher level categories, which is represented by transitive relations, such as "is-a" and "part-of", we can assume that an instance belonging to a category also belongs to all ancestor nodes of that category. Therefore, we would like a classifier explicitly assign all the relevant labels, including the ancestor labels, to a given instance. In this way, the assigned labels would clearly indicate the position of an instance in a category hierarchy.

Definition 3 (Hierarchical Consistency). *A label set $C_i \subseteq C$ assigned to an instance $d_i \in D$ is called consistent with a given hierarchy if C_i includes complete ancestor sets for every label $c_k \in C_i$, i.e. if $c_k \in C_i$ and $c_j \in Ancestors(c_k)$, then $c_j \in C_i$.*

We assume that every instance belongs to the root of a class hierarchy; therefore, from now on we will always exclude the root node from any ancestor set since including it does not provide any additional information on the instance.

Definition 4 (Hierarchical Consistency Requirement). *Any label assignments produced by a hierarchical classification system on a given hierarchical categorization task should be consistent with a corresponding class hierarchy.*

5 Hierarchical Global Learning Algorithm

We propose a new hierarchical global approach to learn a classifier that produces consistent labeling on unseen instances. The method is simple and effective and can be applied to any categorization task with a class hierarchy represented as a DAG. The main idea of the algorithm is to transform an initial (possibly single-label) task into a multi-label task by expanding the label set of each training example with the corresponding ancestor labels or, in other words, by expanding intermediate classes with examples from their offspring nodes. As a result,

in the modified dataset each intermediate category would contain training examples originally assigned to this category and examples originally assigned to descendant nodes of the category in a hierarchical graph. This data modification forces a learning algorithm to focus on high level categories by providing a large number of training examples for those categories. The correct classification of unseen instances into high level categories is very important in hierarchical categorization since high level categories define the most general topics for documents. For example, if we classify a news article about an art exhibition into category "sports" (if "arts" and "sports" are among the top level categories), it would be completely wrong. On the other hand, a mistake made for lower levels, e.g. classification of a document on minor hockey into category "professional hockey", would not be so drastic.

We expect the presented strategy to be successful in the hierarchical settings because a hierarchical structure is typically designed to reflect the semantic closeness of categories. Therefore, we anticipate that related categories share some attributes. In the text categorization context, that means shared vocabulary. For example, categories "hockey" and "American football" have their own specific vocabulary, such as "goalkeeper" or "NHL" for "hockey" and "Super Bowl" or "touchdown" for "football". At the same time, these two categories likely share some common terms, such as "team" or "game", that also appear in their parent category "sports". Our method allows a learning algorithm to explore such common attributes in order to improve classification, especially for high level categories.

Overall, the algorithm consists of three steps:

1. Transformation of training data making them consistent with a given class hierarchy;
2. Application of a regular learning algorithm on a multi-label dataset;
3. Re-labeling of inconsistently classified test instances.

On the first step, we replace each example (d_i, C_i), $d_i \in D$, $C_i \subseteq C$, with $(d_i, \hat{C}_i)$, where $\hat{C}_i = \{\bigcup_{c_k \in C_i} Ancestors(c_k)\}$. Then, we apply a regular learning algorithm, e.g. AdaBoost, on the modified multi-label dataset. Since we train a classifier on the consistent data, we expect that most test instances would be classified consistently as well. However, it is not guaranteed. Some of the test instances can end up with inconsistent labels. This happens if the confidence score of some class A passes a given threshold while the confidence score of one of its ancestor classes does not. For such instances we need to do the third post-processing step. At this step we re-label the instances in a consistent manner by considering the confidence in the predictions for class A and all its ancestor classes. One possible procedure here is to calculate the average of these confidences. If the average is greater than a threshold, we label the instance with class A and all its ancestor classes; if the average is lower than the threshold, we do not assign class A to the instance. This procedure acts as a kind of weighted voting. Each ancestor class votes with its own confidence score. Large positive scores would indicate some certainty in the assigning the class, while negative values would vote against this class assignment.

5.1 Hierarchical AdaBoost

In this work we use the new hierarchical global approach with a state-of-the-art learning algorithm AdaBoost.MH [9],[1] a boosting method designed for multi-class multi-label problems.

AdaBoost.MH works iteratively at each iteration t learning a new "weak" hypothesis h_t on a current distribution P_t over the training examples. After each step, the distribution P_t is modified to increase the weight of the incorrectly classified training examples and decrease the weight of the correctly classified examples. As a result, on the next round $t+1$ a "weak" learner h_{t+1} is forced to focus on examples that are hardest to classify. After a specified number of iterations T, the learning process is stopped, and a weighted voting of the "weak" predictions is used as a final hypothesis: $H(d, \ell) = \sum_{t=1}^{T} \alpha_t h_t(d, \ell)$, $d \in D$, $\ell \in C$, where D is a domain of documents and C is a set of categories. In other words, for each test instance and each class the final hypothesis outputs a real value, called a confidence score. For single-label classification, the classification decision is simply the top-ranked class, the class with the highest confidence score. In a multi-label case, however, we have to select a threshold to cut off class labels for a given instance. One such possible threshold is zero: any positive confidence score indicates that the class should be assigned to an instance, any negative score indicates that the class should not be assigned to an instance. However, we can optimize this threshold value with a simple procedure defined as follows. We train AdaBoost.MH on an available training set S, get the confidence predictions on the same set S, and sort the confidence scores in the decreasing order (t_1, t_2, ..., t_n). Then, we try the confidence scores one by one as possible thresholds and find t_k that results in the best F-measure on the training set S. The final threshold to use on test data is calculated as $\frac{t_k + t_{k+1}}{2}$. This threshold smoothing helps us avoid overfitting when the boosting progresses and gains high confidence in prediction.

6 Hierarchical Evaluation Measure

Most researchers evaluate hierarchical classification systems based on standard "flat" measures: accuracy/error and precision/recall. However, these measures are not suitable for hierarchical categorization since they do not differentiate among different kinds of misclassification errors. Intuitively, misclassification to a sibling or a parent node of the correct category is much better than misclassification to a distant node. To overcome this problem, a hierarchical measure based on the notion of distance has been proposed. The distance between a correct and assigned category, $distance(x, y)$, is the length of the (unique) undirected path from node x to node y in a hierarchical tree. This distance measure gives different

[1] In our experiments we used software BoosTexter (http://www.cs.princeton.edu/~schapire/BoosTexter/), an implementation of AdaBoost.MH specifically designed for text categorization. BoosTexter uses decision stumps (one-level decision trees) as its "weak" learners.

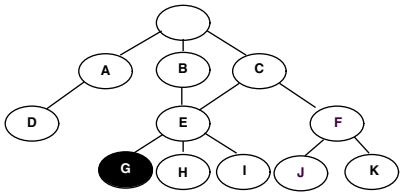

Fig. 1. A sample DAG class hierarchy. The solid ellipse G represents the real category of an instance.

penalties to misclassification into a neighboring or a distant category. However, it has some drawbacks. First, it is not easily extendable to DAG hierarchies (where multiple paths between two categories can exist) and multi-label tasks. Second, it does not change with depth. Misclassification into a sibling category of a top level node and misclassification into a sibling of a node 10-level deep are considered the same type of error (distance of 2). However, an error at the 10th level seems a lot less harmful than an error at the top level.

To express the desired properties of a hierarchical evaluation measure, we formulate the following requirements:

1. The measure gives credit to partially correct classification, e.g. misclassification into node I when the correct category is G (Figure 1) should be penalized less than misclassification into node D since I is in the same subgraph as G and D is not.

2. The measure punishes distant errors more heavily:

a) the measure gives higher evaluation for correctly classifying one level down comparing to staying at the parent node, e.g. classification into node E is better than classification into its parent C since E is closer to the correct category G;

b) the measure gives lower evaluation for incorrectly classifying one level down comparing to staying at the parent node, e.g. classification into node F is worse than classification into its parent C since F is farther away from G.

3. The measure punishes errors at higher levels of a hierarchy more heavily, e.g. misclassification into node I when the correct category is its sibling G is less severe than misclassification into node C when the correct category is its sibling A.

Formally, if we denote $HM(c_1|c_2)$ the hierarchical evaluation of classifying an instance $d \in D$ into class $c_1 \in C$ when the correct class is $c_2 \in C$ in a given tree hierarchy $\mathcal{H} = \langle C, \leq \rangle$, then

1. for any instance $(d, c_0) \in D \times C$, if $Ancestors(c_1) \cap Ancestors(c_0) \neq \oslash$ and $Ancestors(c_2) \cap Ancestors(c_0) = \oslash$, then $HM(c_1|c_0) > HM(c_2|c_0)$;

2. a) for any instance $(d, c_0) \in D \times C$, if $c_1 = Parent(c_2)$ and $distance(c_1, c_0) > distance(c_2, c_0)$, then $HM(c_1|c_0) < HM(c_2|c_0)$;

b) for any instance $(d, c_0) \in D \times C$, if $c_1 = Parent(c_2)$ and $distance(c_1, c_0) < distance(c_2, c_0)$, then $HM(c_1|c_0) > HM(c_2|c_0)$;

3. for any instances $(d_1, c_1) \in D \times C$ and $(d_2, c_2) \in D \times C$, if $distance(c_1, c_1') = distance(c_2, c_2')$, $level(c_1) = level(c_2) + \Delta$, $level(c_1') = level(c_2') + \Delta$, $\Delta > 0$,

$c_1 \neq c'_1$, $c_2 \neq c'_2$, and $level(x)$ is the length of the unique path from the root to node x, then $HM(c'_1|c_1) > HM(c'_2|c_2)$.

Clearly, conventional "flat" measures do not satisfy any of the three requirements. Distance-based hierarchical measures satisfy the second principle, but not always the first and not the third. Thus, we propose a new hierarchical evaluation measure that satisfies all three principles. The new measure is the pair precision and recall with the following addition: each example belongs not only to its class, but also to all ancestors of the class in a hierarchical graph, except the root (we exclude the root of a class hierarchy, since all examples belong to the root by default). We call the new measures hP (hierarchical precision) and hR (hierarchical recall).

Formally, in the multi-label settings, for any instance (d_i, C_i), $d \in D$, $C_i \subseteq C$ classified into subset $C'_i \subseteq C$ we extend sets C_i and C'_i with the corresponding ancestor labels: $\hat{C}_i = \{\bigcup_{c_k \in C_i} Ancestors(c_k)\}$, $\hat{C}'_i = \{\bigcup_{c_l \in C'_i} Ancestors(c_l)\}$. Then, we calculate (micro-averaged) hP and hR as follows:

$$hP = \frac{\sum_i |\hat{C}_i \cap \hat{C}'_i|}{\sum_i |\hat{C}'_i|} \qquad hR = \frac{\sum_i |\hat{C}_i \cap \hat{C}'_i|}{\sum_i |\hat{C}_i|}$$

For example, suppose an instance is classified into class F while it really belongs to class G (Figure 1). To calculate our hierarchical measure, we extend the set of real classes $C_i = \{G\}$ with all ancestors of class G: $\hat{C}_i = \{B, C, E, G\}$. We also extend the set of predicted classes $C'_i = \{F\}$ with all ancestors of class F: $\hat{C}'_i = \{C, F\}$. So, class C is the only correctly assigned label from the extended set: $|\hat{C}_i \cap \hat{C}'_i| = 1$. There are $|\hat{C}'_i| = 2$ assigned labels and $|\hat{C}_i| = 4$ real classes. Therefore, we get $hP = \frac{|\hat{C}_i \cap \hat{C}'_i|}{|\hat{C}'_i|} = \frac{1}{2}$ and $hR = \frac{|\hat{C}_i \cap \hat{C}'_i|}{|\hat{C}_i|} = \frac{1}{4}$.

Following the common practice in conventional text categorization, we can combine the two values hP and hR into one hF-measure:

$$hF_\beta = \frac{(\beta^2 + 1) \cdot hP \cdot hR}{(\beta^2 \cdot hP + hR)}, \beta \in [0, +\infty)$$

In our experiments we used $\beta = 1$, giving precision and recall equal weights.

Theorem 1. *The new hierarchical measure hF satisfies all three requirements for hierarchical evaluation measures listed above*[2]*.*

The new measure is easy to compute: it is based solely on a given hierarchy, so no parameter tuning is required. It is formulated for a general case of multi-label classification with a DAG class hierarchy. Furthermore, we have experimentally proved (results not shown here) that the new measure is superior to standard "flat" measures in terms of statistical consistency and discriminancy - the two criteria Huang and Ling propose to compare classification performance measures [10]. The consistency property means that if we have two classifiers A and B

[2] The proof of the theorem is straight-forward and is available at http://www.site.uottawa.ca/∼svkir/papers/thesis.zip.

and A is more accurate in terms of non-hierarchical measures, it is most likely that our hierarchical measure agrees, and A is better than B in terms of hF as well. The discriminancy property implies that if non-hierarchical measures cannot tell apart the performances of classifiers A and B, our measure is more discriminating and prefers one over the other in most situations.

7 Experiments

We report the results of the experiments on real and synthetic data to compare the proposed hierarchical global approach with "flat" and hierarchical local methods.

7.1 Datasets

Synthetic. We make use of synthetic data to be able to control the size of a class hierarchy and the presence or absence of attribute inheritance between an ancestor class and its descendant classes. The data are designed as follows. For a specified number of levels and for a specified out-degree, i.e. the number of children classes for each intermediate category, we build a balanced tree hierarchy. For each class, including the internal ones, we allocate 3 binary attributes and generate 10 training and 5 test instances per class. Each instance is assigned to exactly one class. The instances are generated randomly according to the following distribution: attributes associated with the class of an instance are set to 1 with 70% probability, all other attributes are set to 1 with 20% probability. We test synthetic data for two extreme situations. The first one is when each class inherits the distribution of attributes from its parent class on top of its own distribution. In other words, the attributes for a class and all its ancestor classes have the high probability (70%) of 1; all other attributes have the small probability (20%). The second situation is when there is no inheritance of attribute distribution: only the attributes associated with the class of an instance have 70% probability of 1, all others have 20% probability. We ran experiments for hierarchies with the number of levels and out-degree each ranging from 2 to 5. Experiments are repeated 100 times for every configuration.

20 newsgroups. This is a widely used dataset of Usenet postings. Following [8], we use a two-level tree hierarchy grouping 15 categories in 5 parent nodes.

RCV1_V2. This is a cleaned version of a new benchmark collection of Reuters news articles. The dataset consists of over 800,000 documents labeled with 103 topics comprising a 4-level hierarchy. Due to the large size of the corpus, we are able to split the data in training and testing subsets in a time-sensitive manner. Articles from 10 full months (September, 1996 - June, 1997) form 10 splits: the first half of a month is used for training, while the second half is used for testing.

Medline. We have also composed 3 large-scale biological datasets for a task of predicting gene functions from biomedical literature. We chose Gene Ontology (GO) to be our category hierarchy. GO provides a hierarchically organized set

of all possible functions that a gene can have in a living organism. It consists of 3 parts: biological process (P), molecular function (F), and cellular component (C). Each part can be seen as an independent DAG hierarchy. We use the Saccharomyces Genome Database (SGD) to obtain manually assigned pairs of a biomedical article describing a yeast gene and the gene's function in GO terms. Overall, we collect 3 datasets, one for each GO hierarchy.

For real datasets, the conventional "bag-of-words" representation is used. All documents are pre-processed: stop words are removed, remaining words are stemmed and converted into binary attributes (a stem is present or not). A simple feature selection technique based on document frequencies is applied. Experiments are run on 10 random training/test splits (in proportion 2:1) for each dataset, except for RCV1_V2 where splits are time-sensitive.

7.2 Comparison with "Flat" AdaBoost

The first set of experiments compares the performance of hierarchical global AdaBoost with the corresponding "flat" approach, i.e. standard AdaBoost that does not take into account any hierarchical information. Both algorithms are run for equal numbers of iterations. The results are presented in Table 1 (columns 5, 7). Evidently, hierarchical AdaBoost significantly outperforms its "flat" version. The differences are more pronounced for larger hierarchies with attribute distribution inheritance as expected. The main difference between the two algorithms is the initial re-labeling that makes training data consistent with a class hierarchy. In really hard tasks, e.g. on the biological data, where the number of classes is very large and the number of training instances per class is very small, the "flat" algorithm suffers a lot producing very poor results. At the same time, this additional step allows the hierarchical method to assemble more training data and learn more accurate classifiers for high level categories, which are favored by the hierarchical evaluation measure.

7.3 Comparison with Hierarchical Local Approach

In the second set of experiments we compare the performances of the hierarchical global and hierarchical local approaches using AdaBoost as the underlying learning algorithm. In the hierarchical local approach we run AdaBoost at each internal node of a hierarchy for the same number of iterations as the global hierarchical AdaBoost. Table 1 (columns 6, 7) shows the results. For most synthetic and real tasks the global approach outperforms the local method. Both algorithms take advantage of extended training data. However, the global approach explores all the categories simultaneously (in a global fashion) assigning only labels with high confidence scores. The local method, on the other hand, uses only local information and, therefore, is forced to make classification decisions at each internal node of a hierarchy, in general, pushing most instances deep down. As a result, the global algorithm is always superior to the local one in terms of precision while slightly yielding in recall. This reflects the conservative nature of the global approach comparing to the local one. Therefore, it should be the method of choice for tasks where precision is the key measure of success. We

Table 1. Comparison of "flat", hierarchical local, and hierarchical global AdaBoost. Numbers in bold are significantly better with 99% confidence.

dataset	# of categories	depth	out-degree	boost. iter.	hF_1 measure		
					"flat"	local	global
newsgroups	20	2	3	500	75.51	**80.01**	79.26
RCV1_V2	103	4	4.68	500	73.10	74.03	**75.86**
medline_P	1025	12	5.41	500	15.32	**59.27**	**59.25**
medline_F	1078	10	10.29	500	8.78	**43.36**	38.17
medline_C	331	8	6.45	500	42.81	72.07	**73.35**
synthetic	6	2	2	200	68.30	73.42	**76.22**
(with attr.	14	3	2	500	58.35	69.40	**74.21**
inheritance)	30	4	2	1000	44.90	68.18	**73.22**
	62	5	2	2000	20.88	68.44	**72.70**
	12	2	3	400	53.47	61.99	**63.45**
	39	3	3	1000	29.51	58.81	**60.69**
	120	4	3	3500	2.67	57.40	**58.22**
	20	2	4	600	41.35	54.26	**55.25**
	84	3	4	2500	6.98	**50.66**	**50.70**
	30	2	5	900	29.99	47.26	**47.87**
synthetic	6	2	2	200	61.69	59.83	**65.95**
(no attr.	14	3	2	500	42.47	44.00	**51.53**
inheritance)	30	4	2	1000	24.49	33.44	**40.18**
	62	5	2	2000	8.45	26.03	**32.61**
	12	2	3	400	41.53	43.87	**48.02**
	39	3	3	1000	14.50	26.33	**29.97**
	120	4	3	3500	0.79	17.97	**21.91**
	20	2	4	600	26.72	32.51	**35.01**
	84	3	4	2500	2.46	17.96	**19.70**
	30	2	5	900	17.14	26.04	**27.12**

can also notice that an increase in out-degree (k) adds a significant number of categories ($\sim k^{depth-1}$) to the global method while only slightly (linearly) complicating the task for the local method. This results in the smaller advantage of global AdaBoost on synthetic hierarchies with large out-degrees and its loss on the highly "bushy" "medline_F" data.

8 Conclusion

In this paper we study a hierarchical categorization problem. We show that hierarchical classification should be consistent with a class hierarchy to fully reproduce the semantics of hierarchical relations. We discuss performance measures for hierarchical classification and introduce natural, desired properties that these measures ought to satisfy. We define a novel hierarchical evaluation measure and show that, unlike the conventional "flat" as well as the existing hierarchical measures, the new measure satisfies the desired properties. It is also simple, requires

no parameter tuning, and has much discriminating power. We present a novel hierarchical global algorithm that produces consistent classification. This algorithm with AdaBoost as the underlying learning procedure significantly outperforms the corresponding "flat" approach, i.e. the approach that does not take into account the hierarchical information. The proposed algorithm also outperforms the hierarchical local top-down approach on many synthetic and real tasks.

In future work, we plan to perform similar experiments with other multi-label classification algorithms as underlying learning components. Unlike AdaBoost.MH, some algorithms may be found behaving consistently in the hierarchical framework even without the post-processing step.

Acknowledgments

This research has been supported by Natural Sciences and Engineering Research Council of Canada (NSERC), National Research Council of Canada (NRC), and the Ontario Centres of Excellence (OCE).

References

1. Koller, D., Sahami, M.: Hierarchically Classifying Documents Using Very Few Words. In: Proceedings of the International Conference on Machine Learning (ICML). (1997) 170–178
2. Dumais, S., Chen, H.: Hierarchical Classification of Web Content. In: Proceedings of the ACM International Conference on Research and Development in Information Retrieval (SIGIR). (2000) 256–263
3. Sun, A., Lim, E.P.: Hierarchical Text Classification and Evaluation. In: Proceedings of the IEEE International Conference on Data Mining (ICDM). (2001) 521–528
4. Ruiz, M., Srinivasan, P.: Hierarchical Text Categorization Using Neural Networks. Information Retrieval **5** (2002) 87–118
5. Wang, K., Zhou, S., He, Y.: Hierarchical Classification of Real Life Documents. In: Proceedings of the SIAM International Conference on Data Mining. (2001)
6. Blockeel, H., Bruynooghe, M., Dzeroski, S., Ramon, J., Struyf, J.: Hierarchical Multi-Classification. In: Proceedings of the SIGKDD Workshop on Multi-Relational Data Mining (MRDM). (2002) 21–35
7. Tsochantaridis, I., Hofmann, T., Joachims, T., Altun, Y.: Support Vector Machine Learning for Interdependent and Structured Output Spaces. In: Proceedings of the International Conference on Machine Learning (ICML). (2004)
8. McCallum, A., Rosenfeld, R., Mitchell, T., Ng, A.: Improving Text Classification by Shrinkage in a Hierarchy of Classes. In: Proceedings of the International Conference on Machine Learning (ICML). (1998) 359–367
9. Schapire, R., Singer, Y.: Improved Boosting Algorithms Using Confidence-rated Predictions. Machine Learning **37** (1999) 297–336
10. Huang, J., Ling, C.: Using AUC and Accuracy in Evaluating Learning Algorithms. IEEE Trans. on Data and Knowledge Engineering **17(3)** (2005) 299–310