Deep Classification in Large-scale Text Hierarchies

Gui-Rong Xue[1] Dikan Xing[1] Qiang Yang[2] Yong Yu[1]

[1]Dept. of Computer Science and Engineering
Shanghai Jiao-Tong University
{grxue, dkxing, yyu}@apex.sjtu.edu.cn

[2]Hong Kong University of Science and Technology
Clearwater Bay, Kowloon, Hong Kong
qyang@cs.ust.hk

ABSTRACT

Most classification algorithms are best at categorizing the Web documents into a few categories, such as the top two levels in the Open Directory Project. Such a classification method does not give very detailed topic-related class information for the user because the first two levels are often too coarse. However, classification on a large-scale hierarchy is known to be intractable for many target categories with cross-link relationships among them. In this paper, we propose a novel deep-classification approach to categorize Web documents into categories in a large-scale taxonomy. The approach consists of two stages: a search stage and a classification stage. In the first stage, a category-search algorithm is used to acquire the category candidates for a given document. Based on the category candidates, we prune the large-scale hierarchy to focus our classification effort on a small subset of the original hierarchy. As a result, the classification model is trained on the small subset before being applied to assign the category for a new document. Since the category candidates are sufficiently close to each other in the hierarchy, a statistical-language-model based classifier using n-gram features is exploited. Furthermore, the structure of the taxonomy can be utilized in this stage to improve the performance of classification. We demonstrate the performance of our proposed algorithms on the Open Directory Project with over 130,000 categories. Experimental results show that our proposed approach can reach 51.8% on the measure of Mi-F1 at the 5th level, which is 77.7% improvement over top-down based SVM classification algorithms.

Categories and Subject Descriptors

H.4.m [Information Systems]: Miscellaneous; I.5.4 [Pattern Recognition]: Applications | *Text processing*

General Terms: Algorithms, Performance, Experimentation.

Keywords: Deep Classification, Large Scale Hierarchy, Hierarchical Classification.

1. INTRODUCTION

Text classification is at the heart of Web page classification, which can find many applications ranging from Web personalization to targeted advertisements [1] on Web pages. In text classification, our aim is to categorize a given text document into predefined classes, where the main techniques used are machine learning methods such as support vector machines (SVM). However, most machine learning methods confine themselves to classifying a document into two or a few predefined categories. As such, the power of Web-page classification is severely limited. In this paper, we take the first step in exploring how to scale up the target categories from a few to hundreds of thousands, in hierarchies of classes such as the Open Directory Project (ODP) and Yahoo! Directories, thus elevating text classification to a new, practical level.

Three main difficulties exist that prevent traditional approaches to classification from being applied. The first is the sheer size of the taxonomy of categories. Our experiments show that as the number of classes increases to a moderate level, the predictive accuracy dramatically decreases to a level that renders the classifiers unusable. The second difficulty caused by the large size of the taxonomy is that a very long time for training is required by traditional methods. Traditional methods become even intractable for large scale hierarchies [12][13]. The third difficulty lies in the fact that in practice, categories are usually organized as a hierarchical structure. As a result, complex relationships, such as parent-child relations, often exist among the target classes. However, categories on a large-scale hierarchy are assumed to be independent by most of previous works. Thus, these methods cannot utilize the structure information. Moreover, the failure of this assumption may even mislead these methods and decrease their performance. Hence, it is important to utilize the structure of taxonomy in order to obtain a satisfactory performance.

Previous methods to solving the hierarchical classification problem can be classified according to the strategies used in classification [18]. These methods can be generally divided into two types: big-bang approaches and top-down level based approaches. In big-bang approaches, a single classifier is trained on the entire target hierarchy. Big bang methods may allow the classification model to consider the hierarchical structure of classes. Examples are hierarchical SVM [2] and Rocchio-like classifiers [10]. However, it is proved in [12][13] that it is infeasible to directly build a classifier for a large-scale hierarchy.

A second approach to solving the problem is the top-down approach, which constructs classifiers at each level of the category tree where each classifier works as a flat classifier at that level. A document is first classified by the classifier at the root level. It is then classified by the classifiers trained at the lower-level categories until the document reaches a final category [6]. In order to classify a document to a category correctly, it must be classified perfectly at all the ancestors. As a result, a potential problem for the top-down approach is that misclassification at a parent or ancestor category may force a document to be excluded from the child categories before it could be examined by the classifiers of the child categories. Moreover, the classifications over high-level categories may fail easily since some of the categories are too general and thus harder to discriminate as we show in the experiments. In this case, the performance of the top-down approach is significantly impaired. This indicates that the approach makes very restrictive assumptions

SIGIR'08, July 20-24, 2008, Singapore.

on the hierarchies. Liu et al. [12] evaluated a hierarchical SVM classification algorithm on the Yahoo! hierarchy, which contains 132,199 categories. The results show that the performance of classification on hierarchy drops quickly when the level of categories increased.

Generally, text classification on large-scale target hierarchies remains an unsolved problem. In this paper, we propose a novel method that can overcome those difficulties and consequently improve the performance of classification in large text hierarchies. In particular, we present a two-stage approach for large-scale hierarchical classification; we call our method *deep classification*.

- In the first stage, we organize the hierarchy into flat categories, where we perform a search process on large-scale hierarchies by retrieving the related categories for a given document. We rank the categories and take the most related categories as category candidates. Thus, a large-scale hierarchy is pruned into a much smaller but focused one.
- In the second stage, we train a classification model on such a small subset of original hierarchy and classify the given document in that small subset. During this stage, we propose several strategies for training classifiers. The structure of the original hierarchy is utilized to improve the classification performance.

To evaluate our deep classification approach, we have conducted several experiments on the Open Directory Project, which contains more than 130,000 categories. We test the effectiveness of proposed deep classification algorithm by comparing to the state-of-the-art hierarchical classification algorithms. Experimental results show that our proposed approach can reach 51.8% on the measure of Mi-F1 at 5th level, which is 77.7% improvement over the top-down based SVM classification algorithm.

The rest of the paper is organized as follows. In Section 2, we give a brief overview of related work. In Section 3, we describe the framework of proposed algorithms. In Sections 4 and 5, we focus on different strategies at each stage. The evaluation results are shown in Section 6. Section 7 concludes with a summary and suggestions for future work.

2. RELATED WORK

2.1 Traditional Text Classification

In traditional text classification, many algorithms [17][22] have been proposed, such as Support Vector Machine (SVM), k-Nearest Neighbor (kNN), Naive Bayes (NB) and so on. Empirical evaluations on benchmark datasets such as Reuters 21578 [8] and RCV1 [11] have shown that most of these methods are effective in traditional text classification applications.

In Web applications, most of the classification methods, such as SVM and NB, utilized the text classification methods for Web documents by introducing many novel features related to Web document like anchor text, metadata and link structure to optimize the performance. As reported in [12], flat classification based on SVM generally has worse performance than top-down based SVM for the large-scale hierarchical classification. As the first work to investigate the performance on large-scale hierarchy, Liu et al. conducted a large scale analysis on the entire Yahoo categories and reported that the performance of flat SVM is about 30% lower on measures of Micro-F1 at the 4th level and deeper. A recall system [13] was proposed on performing large scale flat classification in which a simple feature based intermediate filtering is used to reduce the potential categories for an instance to a small manageable set. However, the system did not investigate the rich structure among the hierarchical categories. Our experimental results in Section 6.3.4 show that higher performance will be achieved by considering such structure information.

2.2 Hierarchical Text Classification

There are generally two approaches adopted by the existing hierarchical classification methods [18], namely, *big-bang approach* and *top-down approach*.

2.2.1 Big-bang Approach

As described in [18], for the big-bang approach, only a single classifier is used by considering the hierarchical structure of the categories. Given a document, the classifier assigns it to one or more categories in the category tree. The big-bang approach has been designed using SVM [2], Rocchio-like classifier [10], rule-based classifier [16] and association rules [19]. Assuming the distribution of hierarchical categories follows the power law, Yang et al. [24] gave a theoretical analysis of scalability of text classification on flat and hierarchical methods. As reported in their work, the time cost of big-bang classification is larger than that of top-down hierarchical classification. In [2], a modified SVM version is applied on the whole hierarchy. In [4], a search based approach is proposed to find the top K most similar categories for further search result filtering. In [14], McCallum et al. proposed a hierarchical classification approach using a shrinkage approach, in which smoothed parameter estimation of a data-sparse child node is used with its parent node in order to obtain robust parameter estimates. An EM algorithm is used to evaluate the interpolating parameters. However, it is very difficult to conduct this process on our problem setting due to the large number of categories.

Furthermore, in most previous works, experiments were conducted with at most a few thousand categories. The task of building even a single classifier for a large-scale hierarchy is known to be intractable [12]. In contrast, as we show in this paper, our method is scalable in handling large text hierarchies with hundreds of thousands of categories.

2.2.2 Top-down Approach

Top-down level-based classification has been designed based on multiple Bayesian classifiers in [9] and SVM classifiers in [5] and [6]. In [5] and [6], Dumais and Chen proposed a classifier on the top-two levels of the LookSmart categories with 163 categories in total. A top-down based SVM is performed on a very large scale hierarchy in [12]. As reported in the work, the performance is about 40% lower on measures of Micro-F1 at the 5th level and deeper on Yahoo! directory. Directly building top-down classifiers cannot work well in large scale hierarchy due to the problem of error propagation. TAPER [3] is a system for large scale hierarchical classification using naive Bayesian and feature selection on different level categories. TAPER also performed top-down classification on the whole hierarchy.

In **Error! Reference source not found.**, a search result classification system was developed by classifying the search results into deep hierarchies by using category candidates retrieved by query. However, the work focused on the search results analysis through the query, and did not directly solve the document classification issue. This paper proposes a new algorithm for document classification on deep hierarchies.

3. DEEP CLASSIFICATION

In this section, we propose a deep-classification algorithm for large scale category hierarchy. Our algorithm works as follows. For a given document, the entire categories can be divided into two kinds according to their similarity to the document: related categories to

the document and unrelated categories to the document. For a very large scale hierarchy, the number of related categories for a document is much less than the number of the unrelated categories. Traditional hierarchical classification algorithms only focused on building a global classification algorithm to optimize the performance for all categories despite the fact that most of the categories may not be related to a given document. Our deep classification approach can utilize such a property and thus focus on the categories related to the document. We first extract a small subset of related categories from the large-scale hierarchies. We then perform classification on these extracted categories utilizing the structure of the original hierarchy.

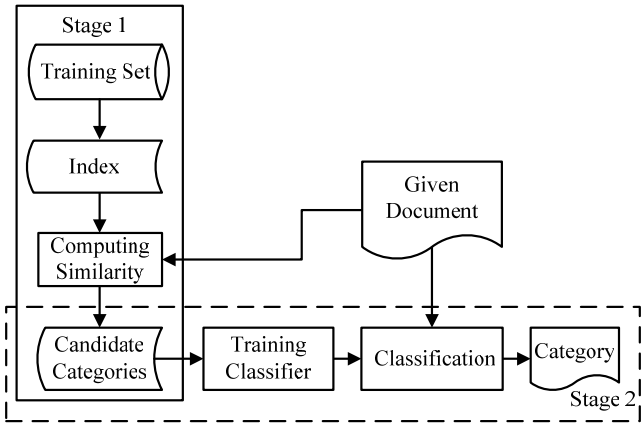

Figure 1. Flowchart of Deep Classification

The algorithm is shown in Figure 1, where we present a two-stage algorithm consisting of a search stage and a classification stage. In the search stage, we try to find a subset of categories from the large scale hierarchy related to given document. As a result, the large scale hierarchy is pruned into a small one. Then, in the classification stage, we train the classifier on this small hierarchy. It is intuitive that the classification performance on a few categories will be better than that on a larger set of categories. Moreover, structure information of the original hierarchy is applied in this stage to enhance the classification results.

In the search stage, a search based algorithm is used to find the category candidates for the given document. We begin with a set of categories and a pre-classified training set of pages. One can obtain the training set from taxonomies like ODP, Yahoo! or from some other resources depending on the desired application. Compared with the entire hierarchy, this narrowing-down procedure helps reduce the number of target category candidates. The details of this part will be discussed in Section 4.

Next, based on the structure of the pruned hierarchy, a classifier is trained and used to categorize the document into categories. In this stage, by considering the pruned hierarchical structure, three training data selection strategies are proposed in Section 5.1 which utilize the hierarchical structure. Then, based on selected training data, we perform classification for the given document. Since the classification model needs building instantly, it is important for the algorithm to be efficient in order to make our method scalable. To satisfy this goal, we compare different classifiers and propose a light-weighting classifier based on naïve Bayes classifier which is described in Section 5.2.

4. STRATEGIES IN SEARCH STAGE

In the search stage, we propose two strategies to find the category candidates for a given document: document-based search strategy and category-based search strategy.

4.1 Document based Strategy

Document based strategy compares the relevance between the given document and these documents in the training set. The documents in a training set and the given document to be classified are both represented with normalized term frequency vectors. A comparison is done using the cosine similarity measure. Top *N* most similar documents are selected as related documents to the given document. These categories are taken as the category candidates.

4.2 Category based Strategy

With Category based strategy, we represent the category with the Web pages in this category and then perform the similarity calculation between the categories and the given document. From these pre-classified pages in the categories, we can build a vector of term frequencies for each of the categories. The given document is also represented with the term frequency vector of the document. Then, we compute the cosine similarity between the vector of a given document and the categories.

Based on the search stage, we can acquire the related categories, which can be either a leaf node or an internal node of the hierarchy. In the next step, we can classify the given document into these category candidates.

5. STRATEGIES IN CLASSIFICATION STAGE

Based on the related category candidates, a large hierarchy is pruned into a narrow one. A category is kept if the category or its child category is among the candidates. The remaining categories are removed from the hierarchy. An example of pruned hierarchy is shown in Figure 2. Nine categories are shown with bold font as the related categories to the given document, which are acquired based on the related categories search stage.

Then, we perform classification on the pruned hierarchy. Since the pruned hierarchy still has the relationship links among the categories, we wish to use these relations to enhance the results of classification. We apply classification with different strategies in this stage. Below, we consider the steps of this stage in detail.

5.1 Strategies for Training Data Selection

5.1.1 Flat Strategy

The flat strategy is a simple strategy for training data selection in which we just consider the category candidates as a flat structure without considering the category information of their ancestors. From the viewpoint of hierarchical classification, this strategy places all the category candidates directly at the root, which is shown in Figure 3. Then, we directly train the classifier based on the Web pages in the candidate categories.

5.1.2 Pruned Top-down Strategy

Considering the tree structure of pruned hierarchy, we can use the pruned top-down based strategy to train the classifiers. The pruned top-down strategy can be taken as specific type of a top-down classification method proposed in [6][12] by firstly simplifying the large hierarchy into a narrow one. A document is first classified by the classifier at the root level. It is then classified by the classifiers of the lower-level categories until it reaches a final category.

5.1.3 Ancestor-assistant Strategy

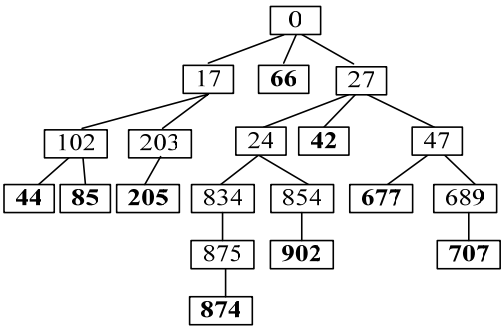

Figure 2. Pruned Hierarchy

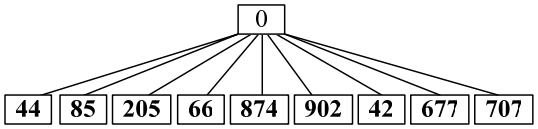

Figure 3. Flat Strategy

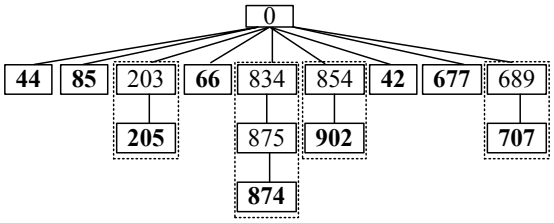

Figure 4. Ancestor-Assistant Strategy

The structure of the hierarchy is largely ignored by previous two strategies. However, as discussed in Section 1, an ideal strategy for training data selection should take this structural information into account. Thus, we propose the ancestor-assistant strategy to utilize this information. This strategy is guided by the following two observations. First, the training data from the category candidate itself may be insufficient in size, especially for a deep category. Thus, we need to obtain more data elsewhere. Second, although the training data from its higher up ancestors may be too general to reflect the characteristics of the deep category candidate, we can borrow data from the ancestors. We should not do this for ancestors that are too high up. Hence, we propose a trade-off between the hierarchical strategy and flat strategy by combining the training data from the category candidate itself and the training data from its ancestors, as long as they do not share the common ancestors of other category candidates. By considering the structure of the hierarchy, the scarcity of training data on deep categories can be alleviated. In addition, we include the training data from a node itself to reserve the characteristics of the categories and the training data will not be largely affected by the training data from its ancestors.

As shown in Figure 2, since the common ancestor is category 24, the training data for category 874 are from those of 834, 875 and 874 while the training data for category 902 are from those of 854 and 902. The tree in Figure 4 can clearly clarify this strategy.

If the node may go up to a higher level, too many training data will be involved. As a result, large amounts of training data may cause the data to be unbalanced and degrade the performance. In this work, we limit the height a node to be two-level-higher than the node itself when applying this method.

5.2 Strategies for Classifier Selection

For a given document, we need to train a specific classifier. Thus, it is preferred to employ a lightweight classifier that does not cost too much time for training. This is because a classifier on various collections of categories may be required in response to different documents. If a classifier such as SVM is employed, the long training time might prevent us from delivering the results to the user in a timely manner. To this end, we prefer the Naive Bayes Classifier (NBC) by considering that probabilistic estimation of NB can be acquired off-line. In the experimental part, we also give the experimental results from SVM and compare the efficiency and effectiveness among them.

5.2.1 Standard NBC

Standard NBC estimates the probability that a test example belongs to a category by computing the following:

$$P(c_i \mid d) \propto P(d \mid c_i)P(c_i) = P(c_i)\prod_{j=1}^{N} P(t_j \mid c_i)^{d_j} \qquad (1)$$

where c_i is a category, d is the test example, N is the vocabulary size, t_j is each term in vocabulary, and d_j is the corresponding value in d for term t_j (usually term frequency).

During the classification stage, the classifier is to assign the category to the given document according to:

$$c^* = \arg\max_{c_i \in C}\{P(c_i \mid d)\} = \arg\max_{c_i \in C}\{P(c_i)P(d \mid c_i)\} \qquad (2)$$
$$= \arg\max_{c_i \in C}\{P(c_i)\prod_{j=1}^{N} P(t_j \mid c_i)^{v_j}\}$$

It is clear that the probability $P(d \mid c_i)$ for each category c_i can be acquired off-line. NBC will take less training time than SVM algorithm on the pruned hierarchies. Thus, it is a kind of lightweight classifier.

5.2.2 N-Gram Language Models for Classifiers

In NBC, terms are considered independent of each other given the category. However, in our situation, most of candidate categories are very close to each other. It is difficult for NBC to distinguish them based on the features of independent terms. In our work, we propose to use Markov *n*-gram language model to perform the classification on candidate categories by considering the Markov dependency between adjacent terms [7][15].

For a term sequence $t_1 t_2 \cdots t_T$, the probability of the sequence is written as:

$$P(t_1 t_2 \cdots t_N) = \prod_{i=1}^{T} P(t_i \mid t_1 \cdots t_{i-1}) \qquad (3)$$

An n-gram model approximates this probability by assuming that the only terms relevant to predicting $P(t_i \mid t_1 \cdots t_{i-1})$ are the previous *n*-1 terms; that is, it assumes the Markov *n*-gram independence assumptions

$$P(t_i \mid t_1 \cdots t_{i-1}) = P(t_i \mid t_{i-n+1} \cdots t_{i-1})$$

We make a straightforward maximum likelihood estimate of n-gram probabilities from a corpus by the observed frequency. We note that different smoothing strategies have been proposed and evaluated in [15].

By using n-gram features to text classification, our prediction is:

$$c^* = \arg\max_{c_i \in C}\{P(c_i \mid d)\} = \arg\max_{c_i \in C}\{P(c_i)P(d \mid c_i)\} \qquad (4)$$

$$= \arg\max_{c_i \in C}\{P(c_i)\prod_{i=1}^{T} P_{c_i}(t_i \mid t_1 \cdots t_{i-1})\}$$

In this work, we use a 3-gram for our classification based on the result reported in [15], which states that 3-grams can often result in the best performance for text classification.

6. EXPERIMENTS

6.1 Experimental Setup

6.1.1 Dataset

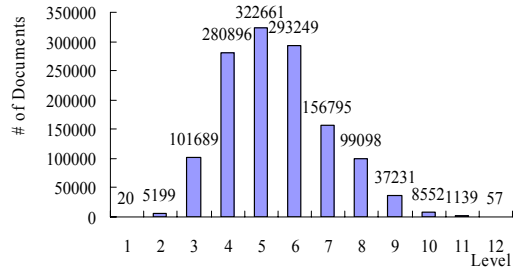

Figure 5. Documents Distribution on Different Level

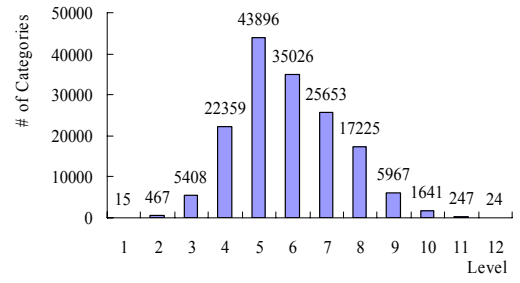

Figure 6. Categories Distribution on Different Level

To evaluate the performance of our algorithm, experiments are conducted using a set of classified Web pages extracted from the Open Directory Project (ODP) (http://dmoz.org/). ODP has about 4,800,870 Web pages and 712,548 categories, in which each Web page is classified by human experts into 17 top level categories (*Arts, Business and Economy, Computers and Internet, Games, Health, Home, Kids and Teens, News, Recreation, Reference, Regional, Science, Shopping, Society, Sports, Adult and World*). Because the Web pages in the regional category are also included in other categories and because many Web pages in the category of the world are not written in English, these two categories are removed in our experiments. Accordingly, 15 categories in all are used in the experiments. After downloading from the Web, we obtain about 1.3 million Web documents in all. The data are divided into a training set and a testing set.

The distribution of these Web pages on 130,000 categories is shown in Figure 5. As shown in the figure, about 76.8% of the documents belong to the top six level categories and about 68.6% of the documents belong to forth-to-sixth-level categories. The distribution of 130,000 categories is shown in Figure 6. As shown in the figure, about 67.8% of the categories are in the top 6 level categories and about 64.1% of categories belong to four-to-six-level category. This shows that classifying the Web pages into deep categories is very important.

As we mentioned in Section 1, the number of related categories for a given document is small. In this part, we present statistics to show the category number for each document. As shown in Table 1, about 93.46% of the documents belong to one category. Only 6.54% of the documents have two or more categories. It is thus reasonable to select a small subset of the large scale hierarchy to perform the classification in this dataset.

Table 1. Categories Number Distribution

Number of Categories	Number of Documents	Percentage
1	1214977	93.460%
2	74237	5.711%
3	2410	0.185%
>=4	195	0.015%

Since the whole data set is too large, we take 130, 000 documents from 1.3 million documents as the testing data. Furthermore, in order to tune the performance of different strategies, 2,000 additional documents are also randomly selected, which is called validation data. The remaining data set is taken as the training data. We build the documents indexing and the categories indexing at the related categories search stage.

6.1.2 Evaluation Metrics

In typical classification experiments, the number of documents is usually a magnitude greater than the number of categories. However, the number of target categories in our tests exceeds 130,000. Conducting experiments with 130K*10 or even more testing documents is very time-consuming. To avoid the 'undefined' problem of Ma-F1 measurements on a number of categories, we use the metric Mi-F1 [21] described in [12] to measure the Mi-F1 on different level.

The process of evaluation is as follows. First, we classify a document into the whole deep hierarchy. For example, a Web page *p* can be classified into the category *Top/Computers/Programming/Languages/JavaScript/W3C_DOM*. Then, we evaluate the performance for each level of the hierarchies according to the classified category. That is, when evaluating the performance of level one, we will judge whether *p* belongs to the category *Top/Computers*. When evaluating the performance of level 2, we will judge whether the Web page *p* belongs to *Top/Computers/Programming*. Hence, it is different from that traditional method that trains the classifier at level 1 or level 2 by aggregating the data of children nodes into its parent category and only evaluating the performance at that level.

6.2 Overall Performance

Three algorithms are compared in this work:

- *Hierarchical SVM*: Top-down classification is an efficient algorithm. In this work, we employ the hierarchical SVM as a representative algorithm for top-down classification.
- *Search based Strategy*: As described in our deep classification algorithm, we can take the most similar category as the category for the given document, which is similar to the nearest neighbor approach.
- *Deep Classification*: This is our proposed algorithm. As we mentioned, there are several strategies for each step. We tune these strategies in Section 6.3. Then, we take the strategies which achieve highest performance. Top 10 categories are taken as category candidates. Category-based search, ancestor-assistant strategy and 3-gram language model for classifiers are taken as the setting for deep classification.

Each algorithm is tuned to achieve the highest performance on the validation data. The overall performance for three algorithms is shown in Figure 7.

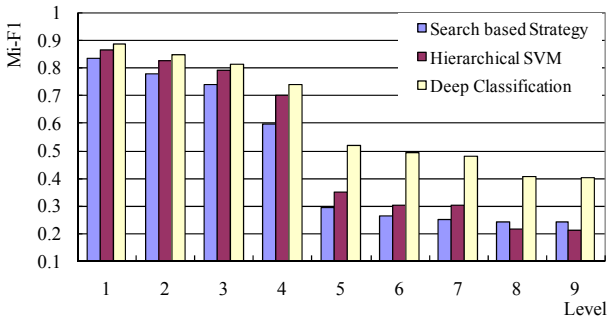

Figure 7. Performance on Different Level

As shown in Figure 7, our proposed deep classification algorithm can achieve consistent improvement over other algorithms at different levels of the hierarchy. As shown in Figure 7, the performance of our proposed algorithm can reach 51.8% at level 5 while the hierarchical SVM only achieve 29.2% at same level. The result shows that our algorithm can get about 77.4% improvements over the top-down approach at level 5. By using the two-stage schema, our algorithm can make accurate classification on a pruned hierarchy. Since the hierarchical SVM is conducted through a top-down method, as we discussed above, the structure of the hierarchy is not properly utilized, so the error at higher levels will be propagated to deeper level. As a result, the deep-level classification cannot achieve good performance. Another reason is that hierarchical SVM cannot construct training set that are sufficient in size when learning deep categories of the hierarchy. As a result, the performance of hierarchical SVM is significantly reduced over the deep level categories.

Furthermore, as shown in the Figure 7, the deep classification algorithm also achieves higher performance than the search based strategy. The result can prove that it is very necessary to perform the classification stage for deep classification algorithm, which can lead to more precise results for the deep hierarchy.

6.3 Strategy Selection

In this section, we will evaluate different strategies used in each stage of proposed deep classification algorithm. Both algorithms are tested on 2000 documents in the validation data, which are randomly chosen. We tune these strategies one by one and fix the other strategies when tuning one strategy.

6.3.1 Search Strategy

As proposed in Section 4, there are two strategies in finding the category candidates for a new document: document-based strategy and category-based strategy. Here we evaluate which strategy can produce higher performance. NB classifier is used as the classifier for its simplicity. All top 10 categories are used.

The experimental results are shown in Figure 8. As shown in Figure 8, the category-based strategy can produce higher performance than the document-based strategy at each level. At level 5, the category-based strategy can achieve 69.2% improvement over the document-based strategy on the measure of Mi-F1. We explain this observation by the fact that the similarity score between several retrieved documents in a category and a given document cannot represent the similarity between the whole category and the given document. The category can provide more information than an individual document in that category. Furthermore, the time cost for category-based strategy is much less than the document-based strategy. Thus, we use the category-based strategy in the search stage for the deep classification algorithm.

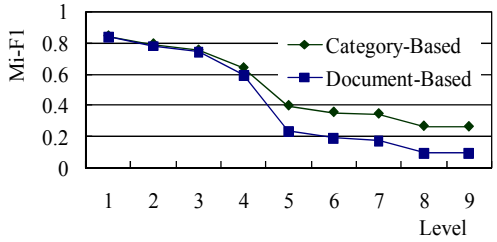

Figure 8. Performance on Different Search Strategies

6.3.2 Candidate Category Number Selection

In the search stage, the system can return different numbers of category candidates. We try to decide how many top ranked categories to be used so the category candidates are adequate. If we only choose one category, the two-stage method is degenerated to the search based strategy only.

We perform evaluation on the tuning data. Our experimental result is reported in Figure 9. As shown in Figure 9, the more categories chosen by the search stage, the more likely we can find the correct target category in the classification stage. However, too many categories also aggravate the burden on training time in the classification stage.

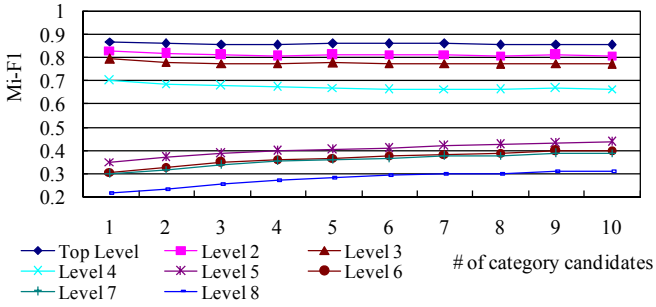

Figure 9. Performance on Different Number of Category Candidates

As shown in the figure, the performance on the top-3 levels is reduced when the number of candidate categories is increased from 1 to 10, although very slightly. However, in deeper levels, the performance increases significantly and tends to be stable near 10 categories. Thus, the number of category candidates is set to 10 considering the trade-off between the time complexity and the performance.

In the following experiments, we set the search strategy as the category-based strategy and use the top 10 categories as the number of category candidates.

6.3.3 Feature Selection

Based on the search stage, category candidates for a new document are found to reduce a large hierarchy into a small one. In our problem, the number of all features exceeds 10,000 in most situations. To solve this problem, we carry out feature selection and show the performance based on using different numbers of features. We perform the CHI-Square feature selection, which is verified as the best feature selection method for text classification in [23]. Two different learning methods are evaluated: Hierarchical SVM and naïve Bayesian (NB). As shown in Figure 10, we can find that the performance with selected 2000 features is similar to that with the whole features. But it is an obvious advantage that fewer features

can reduce time of training and testing. Therefore, in this work, the feature number is limited to 2000 selected by CHI-Square feature selection.

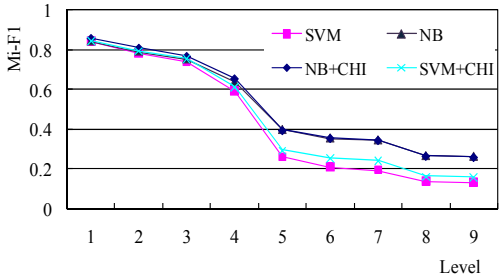

Figure 10. Performance on Feature Selection

6.3.4 *Training Data Selection*

Based on the pruned hierarchy, we considered three strategies of training data selection for further classification. In order to show the performance of different strategies, we conduct an experiment on the small hierarchy generated from the category candidates using the naïve Bayesian classifier. The experimental results are shown in Figure 11. As shown in the figure, we can find the Ancestor-Assistant strategy for training data selection can achieve highest performance. There are about 131.6% and 9.5% improvement over the hierarchical strategy and the flat strategy on the measurement of Mi-F1, respectively, at level 5.

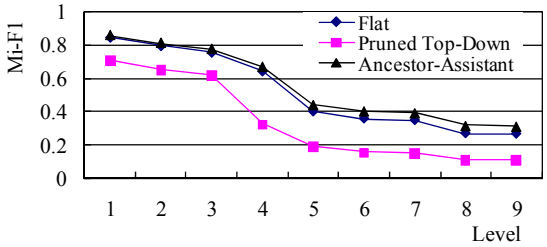

Figure 11. Performance on Different Strategies on Training Data Selection

As shown in these figures, we can find that the performance of the flat strategy is lower than that of the Ancestor-Assistant strategy since this strategy ignores the structure of the hierarchy. Thus it cannot acquire enough training data at some cases since the information from the ancestors is not used to enhance the classifier. The information from the ancestors is vitally important when the training data from the category candidate itself is insufficient. The performance of the flat strategy will be very poor in this case. This experiment also proves that using rich structure of hierarchical categories can enhance the performance of large scale classification, which is largely ignored in [13].

The low performance of the Top-down strategy is due to two factors:

(1) In the top-down scheme, error rates are accumulated at each level which gradually reach an unbearable amount at some deep level of the hierarchy. This problem is overcome in our flat and Ancestor-Assistant strategies where the classification is performed using a flat classifier.

(2) The training data from an ancestor may be too general and cannot characterize the category candidates. In other words, this method improperly utilizes the structure information and thus introduces noise when supplementing the training examples. For example, in Figure 2, training data from category 834 and 854 are used to train classifier when classifying the documents in category 874 and 902, respectively. Our Ancestor-Assistant strategy can overcome this problem since both generalized information from the structure and specific information from the category itself are employed together.

6.3.5 *Classifier Selection*

Classifier selection is a key step to get the final category for the new document. Since the model is trained instantly when given a document, NB and 3-gram NB are proposed to use by considering their efficiency. Here we conduct the experiments to show the performance of two algorithms and also compare to the SVM algorithm. We show the performance of SVM with the features generated by the 3-gram language model. We call it as 3-gram SVM.

As shown in Figure 12, we find that our proposed 3-gram based classification method can achieve higher performance than traditional NB. Since the candidate categories are much similar with each other, it is difficult for NB to distinguish them without considering dependency between words. Another explanation for this issue is that since the category candidates are acquired based on the independent term features, if we still rely on such features to do classification, the effectiveness of classifiers will be decreased. 3-gram classifier takes associated terms into account and thus more discriminative features are used than NBC method. As a result, 3-gram classifier will achieve higher performance.

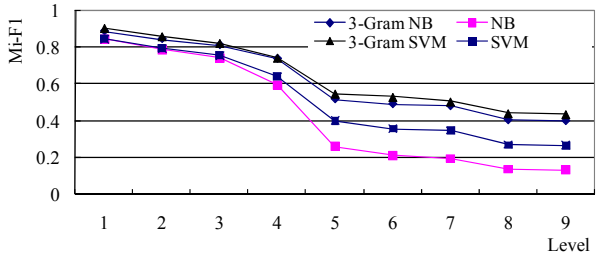

Figure 12. Performance on Different Classifier Selection

Generally, SVM and 3-gram SVM based algorithms can achieve higher performance that NB algorithm and 3-gram NB algorithm, respectively. However, the second stage of deep classification needs an efficient classifier because of the online computation. If we use the 3-gram based SVM, it is very time-consuming to train the model in the online step. Hence, in this work, a 3-gram NB is taken as the second-stage classifier because of its higher performance and efficiency.

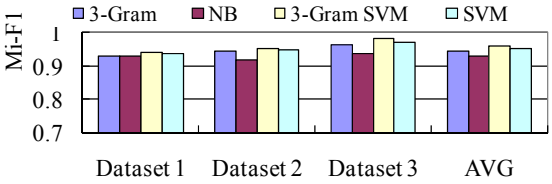

Figure 13. Performance for Different Classifier on Far-Distance Categories

We also conducted additional experiments to validate this conclusion. We randomly picked three groups of deep categories. Each group contains three categories which are far apart from each another (they differ at the first level). We then performed both 3-gram classifier, NB, 3-gram SVM and SVM with a linear kernel on the same training and testing data under each category group. As shown in Figure 13, these classifiers achieve comparable performance to each other. Furthermore, SVM and 3-gram SVM can achieve better performance than NB and 3-gram classifier, respectively.

6.3.6 Time Complexity

The indexing process and the training process for NB classifier and 3-gram language model for classification are conducted off-line. The time complexity of online computation is calculated as follows. As estimated in [24], the average time for document-based search and category-based search are $O(nl_n^2/|V|)+O(n)$ and $O(ml_n^2/|V|)+O(m)$, respectively. Here l_n is the average length of new documents, V is the vocabulary size, m and n is the number of categories and training document, respectively. Since n is much bigger than m, testing time for category-based search will be less than that of document-based search. For the classification stage, we perform the classification only on a narrow hierarchy. Assume that we have m' categories, which is a constant, the time cost is about $O(l_d*m'+m'\log m')$ for NBC and about $O(l_d^3*m'+m'\log m')$ for 3-gram language model. Therefore, the online time complexity is acceptable, which indicates that our algorithm is scalable and can handle very large hierarchies efficiently.

7. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a novel algorithm for Web classification on a large scale text hierarchy. A two-stage algorithm is presented, consisting of a search stage and a classification stage. The search stage prunes the original large hierarchy into a small and tractable one. The structure of the original hierarchy is considered when we train a classifier in the classification stage. As a result, our method is both efficient and effective in handling very large scaled hierarchies. Experimental results showed that our proposed algorithm can achieve 77.7% improvement over top-down based SVM classification algorithm on the accuracy at 5th level on the large-scale hierarchies.

As one future work, we will extend the deep classification algorithm for different kinds of applications, such as online advertisement classification. Another work is to improve the efficiency of the search stage algorithm of deep classification. We will develop more effective indexing algorithms to improve the classification performance.

8. REFERENCES

[1] Broder, A., Fontoura, M., Josifovski, V., and Riedel, L. A Semantic Approach to Contextual Advertising. In Proc. of ACM SIGIR '07. ACM, New York, NY, pp. 559-566, 2007.

[2] Cai, L. and Hofmann, T. Hierarchical Document Categorization with Support Vector Machines, In Proc. of CIKM 2004, pp. 78-87, 2004.

[3] Chakrabarti, S., Dom, B., Agrawal, R., and Raghavan, P., Scalable Feature Selection, Classification and Signature Generation for Organizing Large Text Databases into Hierarchical Topic Taxonomies. The VLDB Journal, vol. 7, no. 3, pp. 163-178, 1998.

[4] Chekuri, C., Goldwasser, M., Raghavan, P., and Upfal, E. Web search Using Automatic Classification. In Proc. of ACM WWW-96, San Jose, US, 1996.

[5] Chen, H., and Dumais S. Bringing Order to the Web: Automatically Categorizing Search Results. In Proc. of CHI, pp. 145-152, 2000.

[6] Dumais, S. and Chen, H. Hierarchical Classification of Web Content. In Proc. of 23th ACM SIGIR, pp. 256-263, 2000.

[7] Gao, J. F. and Nie, J. –Y. Wu, G. Y. and Cao, G. H. Dependence Language Model for Information Retrieval. In Proc. of 27th ACM SIGIR, pp. 170-177, ACM Press, 2004.

[8] http://www.daviddlewis.com/resources/testcollections/reuters21578/.

[9] Koller, D. and Sahami, M. Hierarchically Classifying Documents using Very Few Words. In Proc. of the 14th ICML, 1997.

[10] Labrou, Y. and Finin, T. W. Yahoo! as an Ontology: Using Yahoo! Categories to Describe Documents. In Proc. of the 8th ACM CIKM, pp. 180-187, 1999.

[11] Lewis, D. D., Yang Y., Rose T. G., Li F. RCV1: a New Benchmark Collection for Text Categorization Research. Journal of Machine Learning Research, Vol. 5, pp. 361-397, 2004.

[12] Liu, T.-Y., Yang, Y.-M., Wan, H., Zeng, H.-J., Chen, Z. and Ma, W.-Y. Support Vector Machines Classification with a Very Large-scale Taxonomy. SIGKDD Explorations, 7(1): pp. 36-43, 2005.

[13] Madani, O., Greiner, W., Kempe, D., and Salavatipour, M. Recall Systems: Efficient Learning and Use of Category Indices. In Proc. of AISTATS, 2007.

[14] McCallum, A. and Rosenfeld, R. Improving Text Classification by Shrinkage in a Hierarchy of Classes. Tom Mitchell and Andrew Ng. ICML-98, 1998.

[15] Peng, F. C, Schuurmans, D. and Wang, S. J. Augumenting Naive Bayes Text Classifier with Statistical Language Models. Information Retrieval, 7 (3-4), pp. 317-345, Kluwer Academic Publishers, 2004

[16] Sasaki, M. and Kita, K. Rule-based Text Categorization using Hierarchical Categories. In Proc. of the IEEE Int. Conf. on Systems, Man, and Cybernetics, pp. 2827-2830, 1998.

[17] Sebastiani, F. Machine Learning in Automated Text Categorization. ACM Computing Surveys, Vol. 34, No. 1, pp. 1-47, 2002.

[18] Sun, A. and Lim, E.-P. Hierarchical text classification and evaluation. In Proc. of IEEE ICDM (pp. 521-528). IEEE Computer Society, 2001.

[19] Wang, K., Zhou, S., and He, Y. Hierarchical Classification of Real Life Documents. In Proc. of the 1st SIAM Int. Conf. on Data Mining, Chicago, 2001.

[20] Xing D. -K., Xue G.-R., Yang Q., Yu Y. Deep Classifier: Automatically Categorizing Search Results into Large-Scale Hierarchies. In Proc. of ACM WSDM 2008. pp. 139-148.

[21] Yang, Y. An Evaluation of Statistical Approaches to Text Categorization. Journal of Information Retrieval, Vol. 1, No. 1/2, pp. 67-88, 1999.

[22] Yang, Y. and Liu, X. A Re-examination of Text Categorization Methods, In Proc. of ACM SIGIR'99, pp. 42-49, 1999.

[23] Yang, Y. and Pedersen, J.P. A Comparative Study on Feature Selection in Text Categorization. In Proc. of 14th ICML, pp. 412-420, 1997.

[24] Yang, Y., Zhang, J. and Kisiel, B. A Scalability Analysis of Classifiers in Text Categorization. In Proc. of ACM SIGIR'03, pp. 96-103, 2003.