# Support Vector Machines Classification with A Very Large-scale Taxonomy

Tie-Yan Liu[1], Yiming Yang[2], Hao Wan[3], Hua-Jun Zeng[1], Zheng Chen[1], and Wei-Ying Ma[1]

[1]Microsoft Research Asia, 5F, Sigma Center, No. 49, Zhichun Road, Beijing, 100080, P. R. China
[2] School of Computer Science, Carnegie Mellon University, PA, USA
[3]Dept. Electronic Engineering, Tsinghua University, Beijing, 100084, P. R. China

{t-tyliu, hjzeng, zhengc, wyma}@microsoft.com, yiming@cs.cmu.edu

## ABSTRACT

Very large-scale classification taxonomies typically have hundreds of thousands of categories, deep hierarchies, and skewed category distribution over documents. However, it is still an open question whether the state-of-the-art technologies in automated text categorization can scale to (and perform well on) such large taxonomies. In this paper, we report the first evaluation of Support Vector Machines (SVMs) in web-page classification over the full taxonomy of the Yahoo! categories. Our accomplishments include: 1) a data analysis on the Yahoo! taxonomy; 2) the development of a scalable system for large-scale text categorization; 3) theoretical analysis and experimental evaluation of SVMs in hierarchical and non-hierarchical settings for classification; 4) an investigation of threshold tuning algorithms with respect to time complexity and their effect on the classification accuracy of SVMs. We found that, in terms of scalability, the hierarchical use of SVMs is efficient enough for very large-scale classification; however, in terms of effectiveness, the performance of SVMs over the Yahoo! Directory is still far from satisfactory, which indicates that more substantial investigation is needed.

## Categories and Subject Descriptors

F.2 [Analysis of Algorithms and Problem Complexity]: Miscellaneous; H.4.m [Information System Applications]: Miscellaneous; I.5.4 [Pattern Recognition]: Applications – Text processing.

## General Terms

Technology Assessment, Performance and Scalability Analysis, Empirical Validation.

## Keywords

Text Categorization, Very Large Web Taxonomies, Threshold Tuning Strategies and Algorithm Complexity

## 1. INTRODUCTION

Text categorization (TC), which is the problem of assigning predefined categories to free-text documents by means of supervised learning, is a key technology of text mining and has attracted wide attention from many different research fields. In the literature, many algorithms [5][16][18] have been proposed, such as Support Vector Machines (SVMs), *k*-Nearest Neighbor (*k*-NN), Naïve Bayes (NB) and so on. Empirical evaluations on benchmark datasets such as Reuters 21578 [21] and RCV1 [14] have shown that most of these methods are quite effective in traditional text classification applications. However, in recent years, there has emerged a trend for the scale of text categorization problems to become larger and larger. For example, Web taxonomies (i.e. the Yahoo! Directory http://dir.yahoo.com/ and the Open Directory Project (ODP) http://dmoz.org/) often have hundreds of thousands of categories. This naturally raises the question of whether the aforementioned TC methods can successfully handle these very large-scale classification taxonomies. In this paper, we will try to answer this question from the views of both scalability and effectiveness. For simplicity, we will only discuss SVMs, although the same methodology could be adopted in the analysis of other classifiers.

To the best of our knowledge, there are two main kinds of implementation for SVMs, namely flat SVMs and hierarchical SVMs. Flat SVMs refer to those SVM classifiers that do not take advantage of the structure of the taxonomy tree. Many multi-class versions of SVMs can be regarded as flat SVMs, such as one-against-rest SVMs, one-against-one SVMs and SVMs with error correcting output coding [2][3][9][12]. Among these, the earliest, yet most popular, are one-against-rest SVMs, in which an SVM model is trained to distinguish each category from all the other categories combined. In this paper, we will use this as the representative of flat SVMs. Hierarchical SVMs refer to those methods that decompose the training tasks according to the structure of the taxonomy [4][5][6][10][17][19]. That is, an SVM model is trained to distinguish only among those categories with the same parent node in the taxonomy tree.

As far as we know, the maximum number of categories ever tested in an evaluation of SVMs, whether flat or hierarchical, is no more than 5000 [4]. This is less than 2% of the number of categories of the Yahoo! Directory. Such limited empirical studies are far from sufficient for understanding the performance of SVMs over very large-scale taxonomies. In order to gain this understanding, in this paper, we assess the usefulness of SVMs for very large-scale categorization tasks where the categories are hierarchically arranged. In particular, we theoretically analyze the scalability and effectiveness of SVMs and two threshold tuning methods (score cut and rank cut). We also describe a scalable system we have developed using distributed classifiers, which can handle experiments on large-scale real-world datasets. Lastly, we provide the first thorough experimental results on the tradeoff between classification effectiveness and efficiency of SVMs with the Yahoo! taxonomy.

The paper is organized as follows. Section 2 gives a brief literature review on SVMs. Section 3 introduces some characteristics of the Yahoo! Directory. Theoretical analysis and experimental results on scalability and effectiveness of SVMs are discussed in Section 4 and 5. Concluding remarks are provided in Section 6.

## 2. BRIEF LITERATURE REVIEW

According to our categorization of SVMs in the previous section, for flat SVMs, each SVM model is trained to distinguish one category from all the other categories. For the testing phase, an exhaustive search is used to classify an instance into the category with the highest confidence score. It is clear that the complexity of flat SVMs is proportional to the number of categories. Therefore, when handling hundreds of thousands of categories, the computational load will increase to unacceptable levels.

To tackle this problem, people have utilized the hierarchical structure of the taxonomy tree to decompose the classification task. In [5] and [6], Dumais used hierarchical SVMs to classify the LookSmart dataset. For the training phase, a classifier was trained to distinguish only those categories with the same parent node in the taxonomy tree. And for testing, a *pachinko-machine* search was used, where an SVM model is used only if the model of its parent category says YES on the test instance. They claimed improved classification performance with a significant (i.e. more than 80%) reduction in computation compared to the flat baseline. However, because they only used the top two levels of the LookSmart categories (163 categories in total) in their experiments, their conclusions might not easily generalize to the case of classifying hundreds of thousands of categories.

Our previous work, [19], is the first paper to give a theoretical analysis of the scalability of TC algorithms. Using the power law to model the category distributions, we derived the bounds of complexity for both flat and hierarchical SVMs. Experiments were conducted on OHSUMED [11] to verify the theoretical analysis: for example, it took 102 hours to train flat SVMs over OHSUMED and only took 26.3 minutes to train hierarchical SVMs. However, these experiments were not conducted over the full domain of OHSUMED (with 14,321 categories in total) but projected from 94 categories in the heart-disease sub domain. Furthermore, the classification performance was not reported, so the trade-off between effectiveness and efficiency was not discussed.

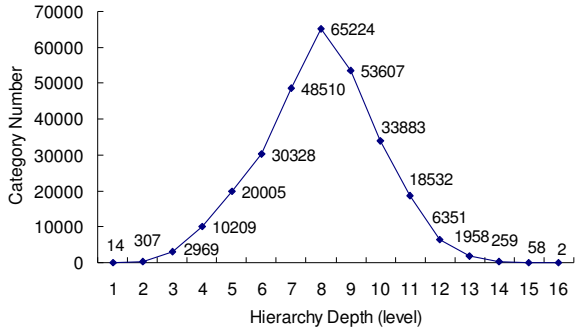
Besides the aforementioned work, other work has also been proposed to investigate the problem of SVM classification over hierarchical taxonomies [4][5][6][10][17][19]. Once again, they verified their findings over datasets with only hundreds, or at most a few thousand categories (such as Reuters 21578, RCV1, the heart-disease sub tree of OHSUMED, and WIPO-alpha [4]). So in summary, the question still remains open as to whether SVMs can scale to hundreds of thousands of categories, and what the tradeoff between efficiency and effectiveness will be. In this regard, it will be very meaningful to conduct a thorough experimental study on SVM classification over the full domain of a very large-scale data corpus, which is the motivation of our paper.

## 3. DATASET

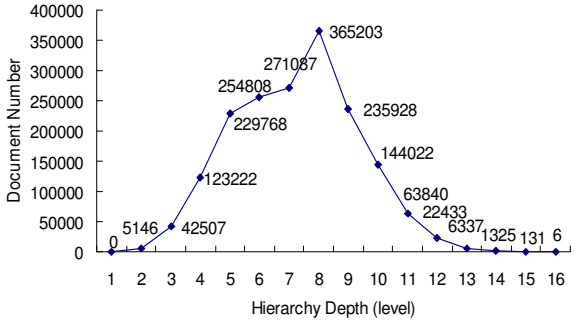
In order to investigate the efficiency and effectiveness of SVMs over very large-scale taxonomies, the full domain of the Yahoo! Directory was used as our experimental data. In this section, we will describe some distinguishing characteristics of the Yahoo! Directory[1] to show that experiments on it can give new insights over previous work on traditional benchmark datasets.

[1] Note that very similar characteristics were also observed in other very large-scale corpora such as ODP, Looksmart and so on.

The first characteristic of the Yahoo! Directory is the deep taxonomy hierarchy. At the time of our crawling (June, 2004), there were 292,216 categories in the Yahoo! Directory organized into a 16-level hierarchy. Since the taxonomy is hierarchical, some categories are only conceptual nodes and have no labeled documents of their own. Therefore, the crawled 792,601 documents actually belong to 246,279 categories. From Figure 1, we can see that both the categories and documents have spindle distributions over levels. That is, there are more categories and documents in the middle than at the upper and lower levels of the taxonomy hierarchy.




(a)




(b)

**Figure 1. Category and document distributions in the Yahoo! Directory.**

The second characteristic is the skewed category distribution over documents. If we only consider the documents assigned by the human editors directly to those 246,279 categories (without counting in the documents assigned to their child categories), the number of documents per category follows the power law distribution (see Figure 2). That is, most categories have very few labeled documents. For instance, over 76% of the Yahoo! categories have fewer than 5 labeled documents. If we denote such categories as "rare categories", we further find that the proportion of rare categories increases at deeper hierarchy levels. As shown in Figure 3, there is no rare category in the first level, but about 36% are rare categories at deep levels.

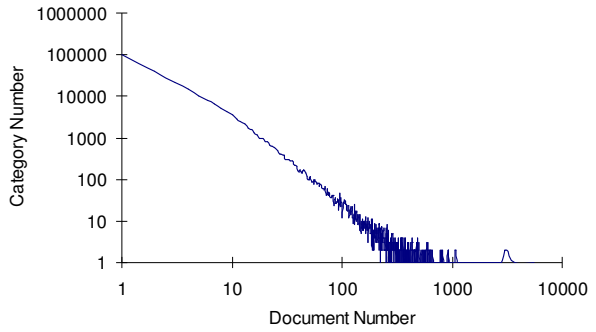



**Figure 2. Power law distribution of the sizes of categories in the Yahoo! Directory.**

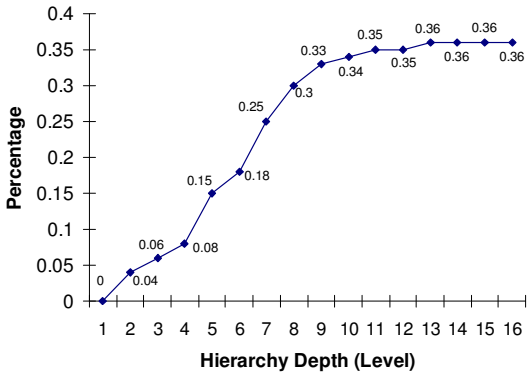



**Figure 3. Percentage of rare categories at different levels of the Yahoo! Directory.**

The third characteristic is that many documents in the Yahoo! Directory have multiple labels (see Figure 4). On average, each document has 2.23 labels, while the largest number of labels for a single document is 31. The existence of multiple labels indicates the necessity of conducting threshold tuning along with the classifier training [20]. However, none of the previous work has analyzed or conducted experiments on the scalability of threshold tuning algorithms.

Because of these particular characteristics, experiments based directly on the full set of Yahoo! Directory categories will be much more informative than any previous work. Therefore, we partitioned the Yahoo! Directory into a training set and a testing set with a ratio of 7:3 to conduct our experimental study. Note that for ease of evaluation, we need to guarantee that each category has at least one positive training example and one test instance, which means that we have to remove those categories containing only one labeled document[2]. As a result, 132,199 categories remained in total with 492,617 documents for training and 275,364 documents for testing. To our knowledge, this dataset is the largest dataset that has been ever used for evaluating SVM classification.

[2] Note that according to [19], if there are labeled documents in a non-leaf node category, an "others" child category will be created and attached to that non-leaf node to include all of these labeled documents. In this way, all documents are placed in leaf-node categories, and removing those categories with only one document will not destroy the connectivity of the hierarchical taxonomy.

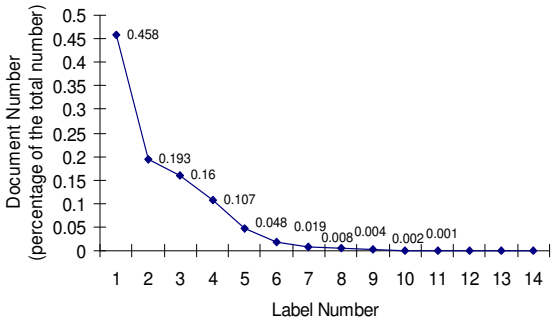



**Figure 4. Distribution of labels per document in the Yahoo! Directory.**

# 4. COMPLEXITY AND EFFECTIVENESS OF SVM CLASSIFICATION

In this section, we discuss whether SVMs (either flat or hierarchical) can theoretically scale up with reasonable classification performance. Specifically, we estimate the time complexity of training and testing over the Yahoo! Directory, and discuss factors that may influence the effectiveness of SVM classification.

## 4.1 Complexity Analysis

### *4.1.1 Complexity of SVM Classification*

First, let us consider the complexity of flat SVMs. With the one-against-rest strategy, when training the SVM model for any individual category, we always use the entire training set[3]. Considering that the complexity of SVMs grows super-linearly with the number of training documents [13][15], the overall complexity of flat SVMs can be represented by

$$Q_{train}^{flat} = M \cdot \mathrm{O}(N^c), \quad c > 1. \tag{1}$$

where $N$ is the number of training documents; $M$ is the number of categories in the training set, and $\mathrm{O}(N^c)$ denotes the average training time per SVM model. For our case, $M$=132,199 and $N$=492,617.

For testing, since we need to pass an instance onto all of the $M$ SVM models to find the category with the highest confidence score, the time complexity is

$$Q_{test}^{flat} \sim M \cdot \mathrm{O}(1) \tag{2}$$

where O(1) denotes the average test time per document per SVM model.

Compared to flat SVMs, the case for hierarchical SVMs is a little more complicated, because the size of the training set varies for different nodes in the hierarchy. The size of the training set for a category in a hierarchical SVM setup equals the number of documents in the sub-tree rooted by its parent category[4] [6]. As

[3] The documents in that category are used as positive examples and those in all the other categories are used as negative examples.

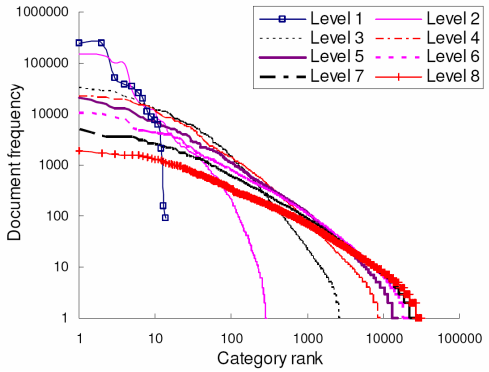
[4] The reason is that each local classification task only distinguishes the categories with the same parent, and both their own labeled documents and the documents of their child categories at lower levels will be used as the training examples.

shown in [19], this size approximately obeys a power law distribution:

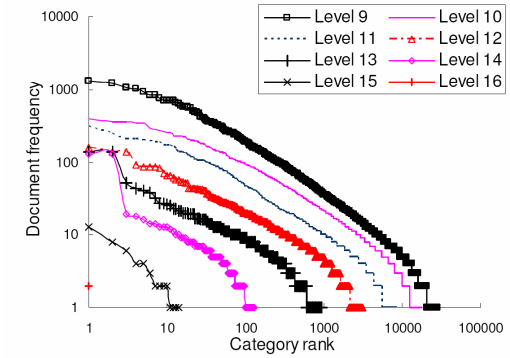
$$n_{ij} \approx n_{i1} j^{-\beta_i},\ j = 1, 2, ..., m_i \tag{3}$$

where $m_i$ is the number of categories defined at the $i$-th level; $j$ is the size-based rank of the categories; $n_{ij}$ is the number of training documents for the $j$-th category at the $i$-th level; $n_{i1}$ is the number of training documents for the most common category at the $i$-th level; and $\beta_i$ is a level-specific parameter.

As discovered in [19], the power law distributions at the different levels in OHSUMED are almost the same. As a result, a global $\beta$ was used for the complexity analysis. However, data statistics on the Yahoo! Directory show that the distributions are not similar, especially for the top and bottom levels (see Figure 5). Therefore, instead of using a global $\beta$, we use local $\beta_i$'s derived from the slopes of the regression lines in Figure 5 for complexity estimation.




(a)




(b)

**Figure 5. Power law distribution of document frequency over category rank in the training set of the Yahoo! Directory.**

Furthermore, $m_i=b^i$ was used in [19] to approximate the number of categories at the $i$-th level (where $b$ is a global branching factor). However, our observations of the Yahoo! Directory show that the branching factors are quite different for different levels, and the number of categories per level follows a spindle distribution (see Figure 1). This distribution cannot be modeled well by $m_i=b^i$, which is an increasing function of $i$. Therefore, we use different branching factors $b_i$ for different levels, and choose $m_i$ based on statistics.

To summarize, we use the $m_i$ and $b_i$ values in Table 1 to conduct our complexity analysis, where level 0 is a virtual level for ease of derivation. In particular,

$$Q_{train}^{hierarchical} = \sum_{i=0}^{15}\sum_{j=1}^{m_i} b_i \cdot \mathrm{O}(n_{ij}^c) \approx \sum_{i=0}^{15}\left(b_i \sum_{j=1}^{m_i} j^{-c\beta_i}\right)\cdot \mathrm{O}(n_{i1}^c) \tag{4}$$

Because $\sum_{j=1}^{m_i} n_{i1} j^{-\beta_i} = N_i < N$, or $n_{i1} < \frac{N}{\sum_{j=1}^{m_i} j^{-\beta_i}}$, we can get

$$Q_{train}^{hierarchical} \le \sum_{i=0}^{15} \frac{b_i \sum_{j=1}^{m_i} j^{-c\beta_i}}{\left(\sum_{j=1}^{m_i} j^{-\beta_i}\right)^c} \cdot \mathrm{O}(N^c). \tag{5}$$

Considering that for $\theta > 0$ and $\theta \neq 1$,

$$\frac{1-(m+1)^{1-\theta}}{\theta-1} = \int_1^{m+1} x^{-\theta} dx < \sum_{j=1}^{m} j^{-\theta} < 1 + \int_1^m x^{-\theta} dx = \frac{\theta - m^{1-\theta}}{\theta-1} \tag{6}$$

Replacing the numerator and denominators in (5) with the upper and lower bounds from (6) respectively, with appropriate substitutions for θ, we have

$$\begin{aligned} Q_{train}^{hierarchical} &\le \left(b_0 + \sum_{i=1}^{15} \frac{\frac{b_i}{c\beta_i - 1}\left(c\beta_i - m_i^{1-c\beta_i}\right)}{\left[\frac{1}{\beta_i - 1}\left(1-(m_i+1)^{1-\beta_i}\right)\right]^c}\right)\cdot O(N^c) \\ &= \frac{1}{M}\left(b_0 + \sum_{i=1}^{15} \frac{\frac{b_i}{c\beta_i - 1}\left(c\beta_i - m_i^{1-c\beta_i}\right)}{\left[\frac{1}{\beta_i - 1}\left(1-(m_i+1)^{1-\beta_i}\right)\right]^c}\right)\cdot Q_{train}^{flat} \end{aligned} \tag{7}$$

**Table 1. Statistics of the training set of the Yahoo! Directory**

| Level | $\beta_i$ | $m_i$ | $b_i$ |
|---|---|---|---|
| 0 | -- | 1 | 14 |
| 1 | 2.60 | 14 | 20.7 |
| 2 | 1.85 | 290 | 9.5 |
| 3 | 1.27 | 2745 | 3.3 |
| 4 | 1.06 | 9073 | 1.7 |
| 5 | 0.92 | 15378 | 1.4 |
| 6 | 0.82 | 21824 | 1.3 |
| 7 | 0.82 | 29006 | 1.3 |
| 8 | 0.71 | 36425 | 0.8 |
| 9 | 0.73 | 28659 | 0.6 |
| 10 | 0.67 | 17896 | 0.5 |
| 11 | 0.66 | 8462 | 0.4 |
| 12 | 0.57 | 3159 | 0.3 |
| 13 | 0.53 | 929 | 0.1 |
| 14 | 0.26 | 128 | 0.1 |
| 15 | 0.82 | 14 | 0.1 |

For the testing phase of hierarchical SVMs, if no threshold tuning is used, only one category with the highest confidence score will be selected at each level. Supposing we have finished the category selection at the $i$-th level, then on average we will test $b_i$ SVM models to further classify the instance into one of the child categories at the ($i$+1)th level. Accordingly, the complexity is

$$Q_{test}^{hierarchial} \approx \frac{1}{M}\sum_{i=0}^{15} b_i \cdot Q_{test}^{flat} \tag{8}$$

### *4.1.2 Complexity of SVM Classification with Threshold Tuning*

As indicated by the third characteristic of the Yahoo! Directory, threshold tuning is necessary for categorizing this multi-labeled data corpus. In this section, we will investigate two popular threshold tuning strategies, rank cut (RCut) and score cut (SCut) [20], to see their effects on the computational cost.

RCut – for each test document, one sorts categories by descending confidence scores and assigns YES to each of the *t*-top categories. Obviously when using an empirical parameter *t* (i.e. the average number of labels per document in the training set), RCut will not introduce any additional computations to the training process.

SCut – one scores a validation set of documents for each category and tunes the threshold over the local pool of scores until the optimal performance of the classifier is obtained for that category. Practically, *k*-fold (most frequently 5-fold) cross validation is often adopted in SCut, which randomly partitions the training set into *k* sets and circularly uses *k*-1 sets for training and the other one for validation. Then, after determining the optimal threshold through cross-validation, one trains a final SVM model using the entire training set for that category. In this way, for any category, *k* additional SVM models need to be trained, each using $\frac{(k-1)}{k}n$ training documents (where *n* is the size of the original training set size for that category, i.e. $n=N$ for flat SVMs and $n=n_{ij}$ for hierarchical SVMs). This corresponds to $k \times \frac{(k-1)^c}{k^c} = \frac{(k-1)^c}{k^{c-1}}$ times the normal training complexity of SVMs. The cross validation also requires the testing of $k \times \frac{n}{k} = n$ documents. Thus, overall we have:

1) The training complexity of flat SVMs with SCut is

$$\tilde{Q}_{train}^{flat} \approx \left(1+\frac{(k-1)^c}{k^{c-1}}\right) \cdot Q_{train}^{flat} + N \cdot Q_{test}^{flat} \tag{9}$$

2) The training complexity of hierarchical SVMs with SCut is

$$\begin{aligned} \tilde{Q}_{train}^{hierarchical} &= \left(1+\frac{(k-1)^c}{k^{c-1}}\right) \cdot Q_{train}^{hierarchical} + \sum_{i=0}^{15} b_i \sum_{j=1}^{m_i} n_{ij} \cdot \mathrm{O}(1) \\ &\le \left(1+\frac{(k-1)^c}{k^{c-1}}\right) \cdot Q_{train}^{hierarchical} + \sum_{i=0}^{15} b_i N \cdot O(1) \\ &= \left(1+\frac{(k-1)^c}{k^{c-1}}\right) \cdot Q_{train}^{hierarchical} + \frac{N}{M} \sum_{i=0}^{15} b_i \cdot Q_{test}^{flat} \end{aligned} \tag{10}$$

It is easy to derive that if $\log \frac{1}{k} \Big/ \log \frac{k-1}{k} > c$ , $\frac{(k-1)^c}{k^{c-1}} > 1$ . Accordingly, we come to the conclusion that SCut with 5-fold cross validation could dominate the training time of SVMs, because $\log \frac{1}{5} \Big/ \log \frac{4}{5} \approx 7.2 > c$.[5] On the other hand, SCut will not affect the testing phase of SVMs much. In fact, the testing complexities of flat SVMs with and without SCut are equal to each other.

$$\tilde{Q}_{test}^{flat} = Q_{test}^{flat} \tag{11}$$

For the testing of hierarchical SVMs, the major change brought by SCut is that more than one category may be selected at each level. To reflect this change, we introduce a new parameter, the average number of selected categories per document (denoted by $\alpha_i$) at each level, to refine (8) to

$$\tilde{Q}_{test}^{hierarchical} \approx \frac{1}{M}\left(b_0 + \sum_{i=1}^{15} \alpha_i b_i\right) \cdot Q_{test}^{flat} \tag{12}$$

Since $\alpha_i$ is dependent on both attributes of the data and the method of classification, we cannot determine its value merely based on the statistics of the corpus (as what we have done for $b_i$, $\beta_i$ and $m_i$). So as a result, we will further discuss (12) with our experimental results in Section 5.

[5] According to [13] and [15], $c \approx 2$ for SMO and c=1.2~1.5 for SVM-light.

In summary, we have given a detailed scalability analysis of SVM classification over the Yahoo! Directory. This analysis gives us some intuitive understanding of the differences between flat and hierarchical SVMs. For example, letting $k$=5 and $c$=2, we find that SCut consumes more than 75% of the computations in the training process. And, whether using SCut or not, hierarchical SVMs save more than 90% of the training computations of flat SVMs. Note also that this analysis can generalize to other very large-scale data corpora such as ODP, Look Smart, etc.

## 4.2 Effectiveness Analysis

Compared to scalability analysis, classification effectiveness is not as clear and predictable because it may be affected by many other factors, such as the number and quality of the training examples. So in this subsection, we will just provide some general ideas on how the particular characteristics of the Yahoo! Directory will affect the effectiveness of SVMs.

The literature has shown that SVMs have high training performance and low generalization error [14][16][18]. However, some papers have also pointed out the potential problems of SVMs when the training set is noisy and imbalanced. For example, [1] found that when working with an imbalanced dataset, SVMs will produce a less effective classification boundary skewed to the minority class. And for the extreme case, when there are too few positive examples, SVMs may totally fail, since there is insufficient evidence for statistical learning. Unfortunately, many Yahoo!-like large-scale data corpora are seriously imbalanced. For instance, Figure 2 shows that most of the categories in the Yahoo! Directory have very few positive examples, but hundreds of thousands of negative examples. In this case, we cannot expect the performance of SVMs to be very good. Furthermore, if evaluating the classification performance with respect to hierarchy depth, we would expect the performance to drop at deeper levels in the hierarchy, because the proportion of rare categories becomes larger (see Figure 3).

This imbalance will also affect the classification effectiveness of the threshold tuning methods. As shown in previous benchmark evaluations [20], SCut outperforms RCut for common categories but performs worse for rare categories. Therefore we can predict that SCut will perform better than RCut at top levels but perform poorer at deeper levels of the Yahoo! Directory.

Given the training process of hierarchical SVMs (as in the previous subsection), the imbalance of the training set in hierarchical SVMs is not as serious as in flat SVMs[6]. Therefore the training error of each individual model in hierarchical SVMs will be smaller than in flat SVMs. Moreover, some previous work showed that local feature selection and local parameter tuning in hierarchical SVMs can also improve training accuracy. However, considering that the *pachinko-machine* search will result in error propagation in the testing phase, we cannot conclude how hierarchical SVMs will perform relative to flat SVMs in terms of classification accuracy without an empirical study, which is the subject of Section 5.

[6] For the same category, there are more positive examples for hierarchical SVMs (including its own labeled documents and the documents in its child categories) than for flat SVMs (including only its own labeled documents), while there are fewer negative examples (only coming from its brother categories) than for flat SVMs (coming from all the other categories in the corpus).

# 5. EXPERIMENTAL RESULTS

## 5.1 Experimental Settings

In this section, we present extensive experimental results to verify the analysis in the previous sections. For this purpose, we developed a distributed classification system as shown in Figure 6 to handle the large scale of our experiments. In the figure, the controller is responsible for dispatching classification tasks to computation agents. The computation agent then trains and saves a model locally with the given examples and parameter settings. When testing, the controller multicasts a test instance to the selected agents and collects their classification results according to a specified strategy (an exhaustive search for flat SVMs and a *pachinko-machine* search for hierarchical SVMs). In our current system, 10 machines were used, each with four 3GHz CPUs and 4GB of memory. When logging the time complexity, we ignored network transmission time.

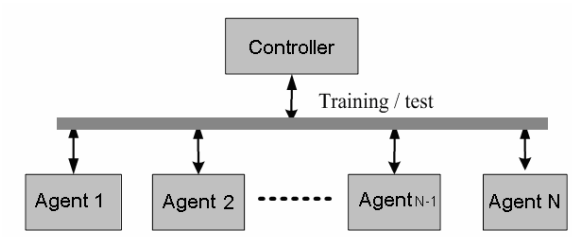



**Figure 6. Distributed classification system.**

For our implementation of SVMs, we used the following settings:

1) We used the SMO [15] algorithm with a linear kernel for the binary version of the SVM classifier.
2) We selected 4000 features for each binary classification task using the CHI algorithm [7][8].
3) For our implementation of SCut, we used 5-fold cross validation. When the number of positive examples $k$ was less than 5, we used $k$-fold cross validation instead. For our implementation of RCut, we chose to set $t = 2$ empirically.
4) For "hierarchical SVMs", we used the same divide-and-conquer strategy as in [5] and [6].

Note that RCut is not compatible with hierarchical SVMs. Since RCut will always say YES to the $t$ top categories and never refuse all, a *pachinko-machine* search with RCut will always descend to a leaf and will never terminate on any non-leaf node. We therefore did not test the combination of hierarchical SVMs with RCut, but only investigated the other three combinations: flat SVMs with SCut, flat SVMs with RCut, and hierarchical SVMs with SCut.

To show the validity of our implementation of SVM algorithms, we verified our code on a well-recognized benchmark corpus, RCV1. We used both micro-averaged and macro-averaged F1 scores (denoted by Micro-F1 and Macro-F1 respectively) as the metrics [18]. Our flat SVMs with SCut had a Micro-F1 of 0.818 and a Macro-F1 of 0.601, which is as good as that reported in [14]. Thus we are confident that our experimental results in the following subsections are representative and correct.

## 5.2 Results on Complexity of SVMs

To better understand the time complexity of SVMs, we logged their run-times, which are given in Table 2. Note that the testing time in this table is the total time for classifying all of the instances in the testing set. So to determine the average response time for one single document, one must divide the given value by the number of test documents. For example, the average response time of hierarchical SVMs is [7]

$$0.12 / 275364 = 4.38e\text{-}7\ h = 0.0016\ s \tag{13}$$

which means that hierarchical SVMs can label about 625 documents per second.

**Table 2. Time complexity of SVMs ($h$)**

| Algorithms | Training | | Testing |
|---|---|---|---|
| | without SCut | with SCut | with SCut |
| Hierarchical SVMs | 0.42 | 2.10 | 0.12 |
| Flat SVMs | 310.87 | 1304.64 | 53.63 |

Based on Table 2, we come to the following conclusions:

1) SCut dominates computations (about 80%) in the training process of both flat and hierarchical SVMs.
2) Flat SVMs cannot be used in very large-scale real-world applications due to their high computational complexity. Even with 10 powerful machines running in parallel, it still took us about 13 days (without SCut) or 1.8 months (with SCut) for training, and 2.2 days for testing. And an average response time of 0.69$s$ for testing one single document is not acceptable for large-scale online classification.
3) The complexity of hierarchical SVMs is much lower and can fulfill the needs of practical applications. Using our distributed classification system, only 0.42$h$ (without SCut) or 2.1$h$ (with SCut) was needed for training; and on average only 0.0016$s$ was needed for classifying each test instance. This result is very interesting: the classification of a dataset as large as the Yahoo! Directory is not as difficult as we previously imagined, if an appropriate method is used.

Furthermore, with Table 2, we can validate the formulas theoretically derived in Section 4. For this purpose, we treat the complexities of flat SVMs as references:

$$Q_{train}^{flat} = 310.87h \tag{14}$$

$$Q_{test}^{flat} = \tilde{Q}_{test}^{flat} = 2105.2/275364 = 1.91e\text{-}4h = 0.69s \tag{15}$$

$$\tilde{Q}_{train}^{flat} = 1304.64h \tag{16}$$

According to (9), we have $\tilde{Q}_{train}^{flat} = \left(1 + 4^c / 5^{c-1}\right) Q_{train}^{flat} + 492617 \times Q_{test}^{flat}$. From this, we can determine that $c$ equals 2.45 for SMO on the Yahoo! Directory, which is a little bit larger than that reported in [15]. Furthermore, we can estimate the training complexity of hierarchical SVMs according to (7) and (10) as follows.

$$Q_{train}^{hierarchical} \le \frac{102.9}{132199} \times Q_{train}^{flat} = 0.24h \tag{17}$$

$$\tilde{Q}_{train}^{hierarchical} \le 3.89 \times Q_{train}^{hierarchical} + \frac{492617 \times 56.1}{132199} \times Q_{test}^{flat} = 0.98h \tag{18}$$

In order to estimate the testing complexity of hierarchical SVMs, we further logged the average number of selected categories per document at each level in Table 3. With this table and (12), we can compute

[7] Here, we use $h$ to denote hour and $s$ to denote second.

$$\tilde{Q}_{test}^{hierarchical} \approx \frac{81.8}{132199} \times Q_{test}^{flat} = 1.18e-7h = 0.00043s \quad (19)$$

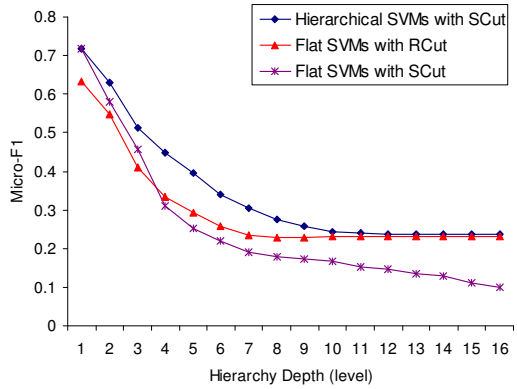
**Table 3. Average number of categories selected per document by SCut at each level**

| **Level** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| $\alpha_i$ | 1.70 | 1.81 | 1.58 | 1.87 | 1.82 | 1.62 | 1.33 | 0.98 |
| **Level** | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| $\alpha_i$ | 0.63 | 0.34 | 0.16 | 0.06 | 0.02 | 0.01 | 0.00 | 0.00 |

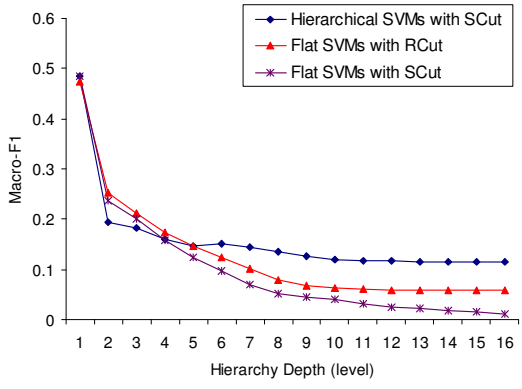
It can be seen that (17), (18) and (19) are a little smaller than what we observed in the experiments (0.42$h$, 2.10$h$ and 0.0016$s$ respectively). Our explanation of this is that our logged time complexity included the overhead of disk I/O and the influences of other processes running on the same machine. Therefore the observations may be noisy and larger than the real complexity.

## 5.3 Results on Effectiveness of SVMs

Classification performance of SVMs with respect to hierarchy depth is shown in Figure 7. In this figure, when performance was calculated for the $i$-th level, we neglected the existence of the deeper levels in the taxonomy tree and put all documents in them into their parent categories at the $i$-th level. In this regard, this figure shows the performance when the task is to classify only the top $i$ levels. It is clear that the data points at the 16-th level correspond to the classification performance over the full domain of the Yahoo! taxonomy.




(a)

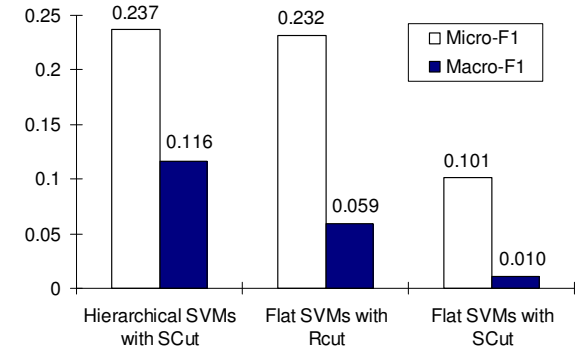



(b)

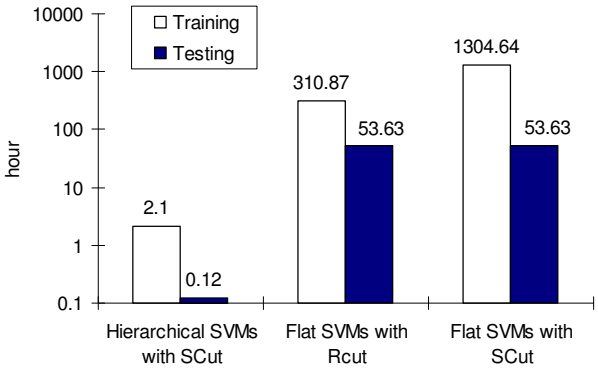
**Figure 7. Classification performance of SVMs over the Yahoo! Directory**

By considering effectiveness and efficiency together, we can obtain Figure 8, the F1 scores in which correspond to the data points at the 16-th level in Figure 7. From these two figures, we come to the following conclusions.

1) For flat SVMs, SCut outperforms RCut at upper levels of the hierarchy. However, as hierarchy depth increases, RCut eventually outperforms SCut.
2) The classification performance of hierarchical SVMs is higher than that of flat SVMs, indicating that the benefits of local feature selection and parameter tuning in the training phase actually overcomes the error propagation caused by the *pachinko-machine* search in the test phase.
3) In terms of effectiveness, the performance of SVMs, whether flat or hierarchical, is far from satisfactory for very large-scale real-world applications. Even the best setting, hierarchical SVMs with SCut, only lead to a Micro-F1 of 0.24 and a Macro-F1 of 0.12.




(a)




(b)

**Figure 8. Tradeoff between effectiveness and complexity of SVMs over the Yahoo! Directory**

In our opinion, the low classification performance is mostly a result of a *data sparseness problem* in the rare categories of the Yahoo! Directory. That is, most categories simply cannot provide enough evidence for us to learn accurate SVM models. Figure 9 plots the macro-averaged performance of hierarchical SVMs vs. the number of training examples in each category: the curve climbs sharply near the right end, meaning that more training examples (more than 100 per category) would improve the accuracy significantly. However, fewer than 0.7% of the Yahoo! categories have more than 100 examples (see Figure 2). This is not only a problem for SVMs though. Other statistical classifiers will face similar challenges. That is, current machine learning methods still need significant improvement when applied to very large-

scale datasets. A possible future research direction is determining how to automatically expand the training set or how to leverage the unlabeled documents on the Web for better classifier training.

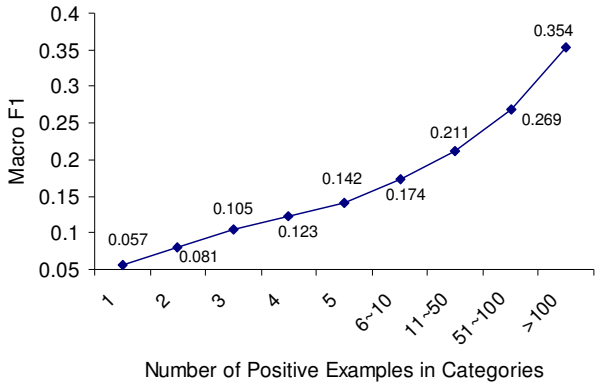



**Figure 9. Classification performance of hierarchical SVMs vs. the number of positive training documents**

## 6. CONCLUSIONS

The difficulties in applying text categorization algorithms to very large problems, especially large-scale Web taxonomies, have been underestimated or at least not studied thoroughly in the literature. In order to gain a better understanding, we conducted the first evaluation of SVMs with the full Yahoo! web-page taxonomy, which yielded the following new conclusions:

1) Threshold tuning (SCut in our paper) dominates the time complexity of offline training of SVMs, which was not well understood until this study.
2) In terms of scalability, while the complexity of flat SVMs is too high, hierarchical SVMs are efficient enough for very large-scale real-world applications.
3) In terms of effectiveness, neither flat nor hierarchical SVMs can fulfill the needs of classification of very large-scale taxonomies.
4) The skewed distribution of the Yahoo! Directory and other large taxonomies with many extremely rare categories makes the classification performance of SVMs unacceptable. More substantial investigation is thus needed to improve SVMs and other statistical methods for very large-scale applications.

## 7. ACKNOWLEGDEMENTS

We would like to thank Bryan Klimt and Qiankun Zhao for their help on polishing the language of this paper.

## 8. REFERENCES

[1] Akbani R., Kwek S., Japkowicz N. Applying support vector machines to imbalanced datasets. ECML, 39-50, 2004

[2] Bottou, L., Cortes, C., Dcnkcr, J., et al. Comparison of classifier methods: A case study in handwriting digit recognition. ICPR, 77-87, 1994.

[3] Bredensteiner, E. J., and Bennett, K. P. Multicategory Classification by Support Vector Machines, Computer Optimization and Applications. 53-79, 1999.

[4] Cai, L. and Hofmann, T. Hierarchical Document Categorization with Support Vector Machines, CIKM, 78-87, 2004.

[5] Chen, H., and Dumais, S. Bringing order to the web: automatically categorizing search results. CHI, 145-152, 2000.

[6] Dumais, S., Chen, H. Hierarchical classification of Web content, In Proc. SIGIR, 256-263, 2000.

[7] Dunning, T. E. Accurate methods for the statistics of surprise and coincidence. Computational Linguistics, Vol. 19, No. 1, 61-74, 1993.

[8] Forman, G. An extensive experimental study of feature selection metrics for text classification. Journal of Machine Learning Research, Vol. 3, 1289-1305, 2003.

[9] Ghani, R., Using Error-Correcting Codes for Text Classification, ICML, 303-310, 2000.

[10] Granitzer, M. Hierarchical text classification using methods from machine learning, Master's Thesis, Graz University of Technology, 2003.

[11] Hersh, W., Buckley, C., Leone, T., and Hickam, D. OHSUMED: An interactive retrieval evaluation and new large test collection for research. SIGIR, 192-201, 1994.

[12] Hsu, C. W., and Lin, C. J. A comparison of methods for multi-class support vector machines. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, 2001.

[13] Joachims, T. Making large-scale SVM learning practical. LS8-Report, 24, University Dortmund, LS VIII-Report, 1998.

[14] 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, 361-397, 2004.

[15] Platt, J. Fast training of support vector machines using sequential minimal optimization. Advances in Kernel Methods - Support Vector Learning, 185-208, MIT Press, Cambridge, MA, 1999.

[16] Sebastiani, F. Machine learning in automated text categorization. ACM Computing Surveys, Vol. 34, No. 1, 1-47, 2002.

[17] Sun, A. and Lim, E. Hierarchical Text classification and evaluation, ICDM, 521-528, 2001.

[18] Yang, Y., and Liu, X. A re-examination of text categorization methods, SIGIR, 42-49, 1999.

[19] Yang, Y., Zhang, J., and Kisiel, B. A scalability analysis of classifiers in text categorization. SIGIR, 96-103, 2003.

[20] Yang, Y. A study of thresholding strategies for text categorization, SIGIR, 137-145, 2001.

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