Combining Global and Local Information for Enhanced Deep Classification
Heung-Seon Oh, Yoonjung Choi, Sung-Hyon Myaeng
Department of Computer Science
Korea Advanced Institute of Science and Technology
373-1, Guseong-dong, Yuseong-gu, Daejeon, 305-701, South Korea
{ojs, choij35, myaeng}@kaist.ac.kr

ABSTRACT
Compared to traditional text classification with a flat category set or a small hierarchy of categories, classifying web pages to a large-scale hierarchy such as Open Directory Project (ODP) and Yahoo! Directory is challenging. While a recently proposed “deep” classification method makes the problem tractable, it still suffers from low classification performance. A major problem is the lack of training data, which is unavoidable with such a huge hierarchy. Training pages associated with the category nodes are short, and their distributions are skewed. To alleviate the problem, we propose a new training data selection strategy and a naïve Bayes combination model, which utilize both local and global information. We conducted a series of experiments with the ODP hierarchy containing more than 100,000 categories to show that the proposed method of using both local and global information indeed helps avoiding the training data sparseness problem, outperforming the state-of-art method.

Categories and Subject Descriptors
H.4.m [Information Systems]: Miscellaneous; I.5.4 [Pattern Recognition]: Applications | Text Processing

General Terms
Algorithms, Performance, Design, Experimentation

Keywords
Deep Classification, Large Scale Hierarchy, Hierarchical Classification

1. INTRODUCTION
Traditional text classification targets at classifying a document to a small number of flat category set or a shallow hierarchy of categories, and with the advances in research, operational classifiers have been deployed for various applications. With the availability of a large-scale hierarchy such as ODP and Yahoo! Directory, however, the problem of classifying a web document to a large-scale hierarchy with more than 100,000 categories presents a new challenge although its solution would be useful for various tasks such as rare query classification, contextual advertising, and web search improvement [1,2,22].

Various classification and feature selection models have been developed for traditional text classification and evaluated on benchmark collections such as Reuters 21578 and 20-newsgroup [6,11,12,14,19]. While the collections have helped making scientific comparisons among different approaches and identifying the state-of-the-art performance, directly applying well-known methods to a large-scale hierarchy of categories is problematic [17]. First of all, it takes long time to train a classifier for a large number of categories and documents. As the number of categories increases to a certain level, furthermore, the performance decreases drastically, mainly because of the lack of training data. Another source of the problem is that even though categories have a parent-child relation in a hierarchy, most past research assumed that they were independent.

Past research on hierarchical text classification can be summarized with the following: big-bang, top-down, and narrow-down approaches. In the big-bang approach, a single classifier is trained for the entire hierarchy, and therefore it takes long time for training. This method of directly building a classifier for large-scale hierarchy is shown to be ineffective [9,10]. In the top-down approach, a classifier is built at each level along with a path from the root to a leaf category. A drawback of this approach is error propagation from a high level to lower levels. If a document is misclassified at the first level, for instance, classifying it into a set of categories at a lower level would be meaningless because it is already in the wrong branch. Error propagations cause more and more significant performance drops as the classifier goes deeper into the hierarchy [9].

As an attempt to avoid the problems of the two approaches, a narrow-down approach has been proposed for a deep classification algorithm [17], which consists of two stages: search and classification. Unlike previous work that solely relies on a machine learning algorithm, the narrow-down approach utilizes a search technique to identify most likely categories first. By narrowing down the category search space from the entire hierarchy to a smaller number of categories, it reduces both error propagation and training time; classification is focused on a subset of the categories, which are highly related to a given document.

Compared to common benchmark collections, hierarchical text classification to web taxonomy suffers from significant data sparseness. Even though web taxonomy has millions of documents that are associated with individual category nodes, they are usually short with a skewed distribution. For example, a document in ODP is composed of one or two short sentences in general. In order to cope with data sparseness, the deep classification algorithm proposed category-based search and ancestor-assistant training data selection in the search and
classification stages, respectively. These strategies achieved a substantial performance improvement.

Nonetheless, the deep classification algorithm suffers from two limitations. First, in spite of the ancestor-assistant strategy that borrows data from the ancestors of a given category for training, the quantity of the documents after expansion is still insufficient. Second, classification is performed locally based on the small number of training documents associated with the chosen candidate categories after narrowing down the categorization search space. Without the path information from the root to the candidate categories, it would be difficult to make a distinction between two categories having a similar set of features although they are far apart in the hierarchy.

Suppose that there are two category candidates D and D' that are highly similar to each other based on the associated documents (hence similar feature sets) but have entirely different paths: A/B/C/D and E/F/G/D', where A and E are the roots of two sub-trees in the hierarchy, respectively. By using the path information, a document is classified correctly into D if it is more similar to A than E. Even if the ancestor-assistant method extends the training data set for a category node such as D and D', the use of global information along the path would provide a proper guidance. While the idea of using global information may sound like going back to the top-down approach, our approach is an extension of the search-based deep classification to improve classification effectiveness while maintaining computational efficiency, as can be seen in the following sections.

Our contributions can be summarized as follows:

1. A noble training data selection strategy is proposed to further alleviate the training data sparseness problem. Our strategy provides additional training data while preserving discrimination among categories for the classification model.
2. A method for utilizing global information is proposed for higher classification quality. A naive Bayes combination model is proposed to combine local and global information, whereas the previous classification algorithms only focus on local information.
3. A series of experiments demonstrate that our algorithm achieves a remarkable performance improvement for deep categorization over the state-of-the-art algorithm.

For the experiments, we used the ODP hierarchy containing about 100,000 categories and 1,500,000 documents. For testing, documents were selected in proportion to the numbers of documents at different levels as opposed to selecting them randomly from the hierarchy.

2. RELATED WORK
2.1 Text Classification

Common text classification consists of two parts: feature selection and classification. Taking the back-of-word approach, valuable terms are retained for classification by feature selection. Comparisons of various feature selection methods for text categorization are found in [20]. A recent study showed remarkable performance in an experiment with common benchmark collections by introducing a novel feature weighting scheme [6]. A survey in [14] introduces many machine learning algorithms such as decision trees, regression methods, the Rocchio method, neural networks, support vector machines (SVM), naive Bayes (NB) etc. for text classification with their effectiveness measured on Reuters 21578 and RCV1 collections. NB was combined with a statistical language model and various smoothing [12]. While they all provide interesting insights and advance classification techniques, they deal with a relatively small number of categories or shallow levels in a hierarchy.

2.2 Hierarchical Text Classification

In big-bang approaches, a single classifier is trained for the entire set of categories in a hierarchy [15]. To build a classifier, various algorithms have been utilized, including SVM [3], Centroid-based [8], rule-based [13], and association rule-based [16] approaches. It’s been shown that it takes much more time in a big-bang approach than in a top-down one [21] even when a hierarchy contains thousands of categories. With hundreds of thousands of categories, moreover, building a single classifier for large-scale hierarchy is intractable [9].

In [23], shrinkage approach was proposed for hierarchical classification. Probabilities of a word for a class corresponding to a leaf node are calculated for the parents up to the root and combined with a set of parameters. This method of dealing with the data sparseness problem is similar to the proposed method in that it uses the data on the ancestor nodes. However, the proposed method stops gathering training data right before a shared ancestor while the shrinkage approach uses all the parent data up to the root of a hierarchy. The shrinkage method requires heavy computation not only because of the need to consider all the data on the path to the root but also because of the time for parameter estimations with the EM algorithm. By focusing on the top- and leaf-level information, the proposed method reduces time complexity considerably.

In a top-down approach, a classifier is trained for each level in a category hierarchy. Several studies adapted a top-down approach with various algorithms such as multiple Bayesian [7], SVM [5,9], and naive Bayes classifier [4]. In [9], it reports that a top-down based SVM classifier results in big performance drops on Yahoo! Directory as it goes to deeper levels. While the micro-averaged F1 value at the top level is about 0.70, it is lower than 0.50 for the 4th level and beyond. This is caused by propagations of errors at early stages involving the ancestors in classifications.

A recent study [17] proposed a deep classification algorithm adapting the narrow-down approach. For a given document, the algorithm searches for category candidates by computing cosine similarity between the input document and those associated with the categories in the hierarchy. For the selected top-k categories, training data are collected not only from the selected category itself but also from its ancestor categories in the hierarchy as long as they are not shared by another selected category. A trigram language model-based classifier is trained based on the selected category candidates and training data. The algorithm showed a considerable overall classification performance improvement, while guaranteeing a promising time complexity due to narrowing-down categories and off-line parameter estimation.

3. OVERVIEW OF THE PROPOSED METHOD

As in Figure 1, our classification method consists of two stages: search and classification. The search stage is responsible for finding the most related categories in the entire hierarchy. The key observation is that most categories are not related to a given document. Hence, the search stage narrows down the number of categories to be considered for actual classification. To support
this stage, we build an inverted index for all the training data consisting of the categories and the associated documents in the hierarchy. Given a document to be classified, top-k categories are retrieved as the candidates for classification based on similarity computation.

The classification stage assigns a category to the input document. The classification algorithm uses two models constructed from the hierarchy: global and local models. A local model is constructed in a way similar to that of the deep classification method proposed recently [17] except that we employ more extensive training data to avoid data sparseness. A global model is generated from the top level categories and used in conjunction with the local model as a way to provide a “high-level guidance” for branching out from the top. This process includes feature generation that needs to be done only once for the entire hierarchy and used every time classification is done. As a result, classification is performed by combining information from local and global models. The details are described in 5.2.

Figure 1. Overview of Advanced Deep Classification

4. SEARCH STAGE
This stage is responsible for selecting category candidates highly relevant to a given document. Two search strategies were proposed in [17]: document and category based. The document based strategy computes relevance between the input document and other documents in the training data, which are all encoded as normalized term frequency vectors. Top-k documents are then selected and the categories containing them are taken as the category candidates.

The category based strategy first represents a category as a term frequency vector using all the documents associated with it and then computes the similarity between each of the category representations and the document also represented as a term frequency vector. Top-k categories are taken as category candidates based on the similarity scores. In both of the strategies, cosine similarity measure is used.

After the search stage, a set of relevant categories are available as candidates for actual classification of the input document. Since the number of categories to which the document is to be classified has been reduced to a manageable size, the next step including both training and classifying steps can be done efficiently.

Given that the next stage entirely depends on the result of the first stage, it is essential to ensure that highly relevant categories be included in the top-k documents selected at the search stage. Among several open source search engines publicly available, we chose Lucene1 for the current implementation and the experiments. Based on our preliminary testing with Lucene, we adopted the category-based strategy that outperforms the other.

5. CLASSIFICATION STAGE
After the search stage, the resulting set of candidate categories become a basis for forming a pruned hierarchy, which is used for training for the particular input document to be classified. Figure 2 shows an example for a pruned hierarchy. The selected candidate categories become the leaf nodes at this point (bold-faced nodes) and their ancestors remain in the tree. While only the resulting tree and the associated documents are used for training in the previous work [17], we use a neighbor-assistant strategy by incorporating the children of the candidate nodes and their siblings for training, as a way to overcome the data sparseness problem. When building a model for each category, we use both the local and global information combined with a naïve Bayes combination model.

5.1 Training Data Selection
Three training data selection strategies were considered in [17]: flat, pruned top-down, and ancestor-assistant strategies. The flat strategy assumes that the ancestors of category candidates are the root node only, ignoring all the intermediate nodes. As a result, the documents associated with the leaf nodes are only considered as the training data. It is most conservative in that it does not use other related nodes in the hierarchy. In the pruned top-down strategy, training data are collected along the path from the root to the category candidate node. While additional documents can participate in the training data, a common set of documents can be used for more than one candidate category.

The ancestor-assistant strategy takes a middle ground. While it collects training documents from the ancestor nodes of the given candidate node by traversing the tree upward, it stops when an ancestor node is also an ancestor of another candidate node (i.e. a leaf node in the pruned hierarchy). The strategy attempts to alleviate the data sparseness problem and, at the same time, ensure no training document be shared by more than one candidate nodes. The latter goal allows for more discriminative training data set. Figure 3 shows how the ancestor-assistant strategy works with respect to the pruned hierarchy in Figure 2. The tree is a result of eliminating all the spurious nodes not used in the training data set. The boxes with a dotted line represent the candidate nodes expanded to their ancestors. Since the nodes 874 and 902 share a common ancestor of node 24, for example, the documents associated with their ancestors below the common node, 834 and 875 for 874 and 854 for 902 are included in the training set.

While this strategy sounds reasonable, our preliminary experiments revealed many of category candidates were so close to each other in the pruned hierarchy that their common ancestors appeared very close to them, making the strategy useless in finding additional training documents. In other words, the tree like the one in Figure 3 is so flat that only a small number of additional documents are found.

To harvest more training data, we propose neighbor-assistant strategy which is an extension of the ancestor-assistant strategy. The main idea is to include the children of the candidate node as

1 Lucene, http://lucene.apache.org
well as the ancestors in the neighborhood so that additional documents can be used for training. Intuitively it makes sense because the documents associated with a child of a given node would be more specific than those associated with the given node. Including more specific documents as opposed to more general ones would be less risky in introducing irrelevant concepts in category representations. Figure 4 illustrates the proposed strategy. We expand the training data with the descendants of the category candidates and their ancestors. The larger dotted circles indicate expanded candidate categories based on the neighbor-assistant strategy whereas the smaller dotted circles are for the ancestor-assisted strategy. For example, the neighbors of node 874 are the ancestors of a candidate category (875, 834), their descendants (98, 472), and the descendants of the category candidate (11, 23).

The remaining problem is to build a classifier based on the training data. An important criterion, as well as a constraint, for deep classification is that the time requirement for training be within a certain bound so that the total classification time for a document is acceptable for a user. This is because a new classifier must be trained for each new document to be classified. This constraint led us to employ a naive Bayes classifier (NBC) with n-gram language modeling, which is lightweight and produces a promising performance as in [11,19].

NBC estimates the posterior probability of a test or input document as follows:

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

where $c_i$ is a category, $d$ is a test document, $N$ is the vocabulary size, i.e. the number of unique features in $d$, $t_j$ is each term in $d$, and $v_j$ is term frequency of term $t_j$. NBC assigns a category to a document as follows:

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

In general, NBC estimates $P(c_i)$ and $P(t_j \mid c_i)$ from a document collection $D$ containing all the documents belonging to each of the categories. In deep classification, however, $D$ denotes the training data for a particular category candidate.

Even though training data have been expanded to the ancestors and neighbors as described previously, classification is still based on local information surrounding the candidate node. We conjecture that global information from the top of the hierarchy would be helpful in deciding which path would be more fruitful when there are more than two sub-trees to check. For example, when a classification decision is to be made between $D$ and $D'$ having paths $A/B/C/D$ and $E/F/G/D'$ respectively, the knowledge that the document is more similar to $A$ than $E$ at the top level can help choosing $D$ over $D'$.

Based on the conjecture, we propose a naive Bayes combination classifier (NBCC) which combines local and global information. NBCC assigns a category to a given document based on NBC with the following parameters:

$$ P(t_j \mid c_i) = \lambda P(t_j \mid c_i^{local}) + (1 - \lambda) P(t_j \mid c_i^{global}) $$  (3)

$$ P(c_i) = \lambda P(c_i^{local}) + (1 - \lambda) P(c_i^{global}) $$  (4)

where $c_i^{local}$ is a category candidate, $c_i^{global}$ is the top-level category of a candidate category, and $\lambda$ is a mixing weight ($0 \leq \lambda \leq 1$). For example, for node 44 in Figure 2, $c_i^{local} = c_{44}$ and $c_i^{global} = c_{17}$. If $P(c_{44} \mid d)$ is similar to $P(c_{42} \mid d)$, for example, the difference between $P(c_{44} \mid d)$ and $P(c_{42} \mid d)$ would help choosing one between $c_{44}$ and $c_{42}$.

5.2 Classification Model

For a given input document to be classified we now have a set of category candidates and an associated set of training documents for each category obtained with the neighbor-assisted strategy.
5.2.1 Global Model

In our global model, we only focus on parameters of categories at the toplevel below the root. Large-scale taxonomies have hundreds of thousands categories. Because of huge diversity of topics, the vocabulary size is very large despite the small amount of training data for each. In order to select terms relevant to the categories at the top level, the chi-square feature selection method known to be the best one in text classification [20] is applied. For each category at the top level below the root, top 15,000 terms are chosen as global features.

Based on the selected terms, prior and conditional probabilities are estimated as follows, respectively:

\[ P(c_{i}^{\text{global}}) = \frac{|D_i|}{|D|} \]  

(5)

where \( D \) is the entire document collection, \( D_i \) is a sub-collection in \( c_{i}^{\text{global}} \), and

\[ P(t_j | c_{i}^{\text{global}}) = \frac{\sum_{d_k \in c_{i}^{\text{global}}} \text{tf}_{jk} + 1}{\sum_{d_k \in c_{i}^{\text{global}}} \sum_{d_k \in c_{i}^{\text{global}}} \text{tf}_{jk} + |V_{\text{global}}|} \]  

(6)

where \( \text{tf}_{jk} \) is the term frequency of term \( t_j \) in top-level category \( c_{i}^{\text{global}} \), \( V_{\text{global}} \) is a set of terms selected by the chi-square feature selection method over the entire document collection. Here we apply add-one smoothing [12].

5.2.2 Local Model

The role of our local model is to estimate parameters of a category candidate directly. Unlike the global model we don’t limit the feature space using the chi-square method or terms already selected. There are two reasons for this. The first is that applying the chi-square feature selection method increases time complexity since generating a local model is on-line processing. The second is because selected terms are limited to the categories at the top level. As a result, limited feature space can not reflect the characteristics of category candidates for deep categories where many of them are close to each other. Parameter estimation is equal to global model except that it is done on selected training data of category candidates and smoothing is not applied.

\[ P(c_{i}^{\text{local}}) = \frac{|D_i|}{|D|} \]  

(7)

where \( D \) represents the document collection for the category candidates and \( D_i \) is for the documents in category candidate \( c_{i}^{\text{local}} \).

\[ P(t_j | c_{i}^{\text{local}}) = \frac{\sum_{d_k \in c_{i}^{\text{local}}} \text{tf}_{jk}}{\sum_{d_k \in c_{i}^{\text{local}}} \sum_{d_k \in c_{i}^{\text{local}}} \text{tf}_{jk}} \]  

(8)

where \( \text{tf}_{jk} \) is the term frequency of term \( t_j \) in category candidate \( c_{i}^{\text{local}} \), and \( V_{\text{local}} \) is a set of terms among \( D \).

6. EXPERIMENTS

6.1 Experiment Setup

6.1.1 Dataset

For experiments, we downloaded ODP\(^2\), which is a large-scale category hierarchy where the documents were classified to the categories by human experts. There are 17 categories at the top level: Arts, Business, Computer, Games, Health, Home, Kids and Teens, News, Recreation, Reference, Regional, Science, Shopping, Society, Sports, World, and Adult. Like the reasons in [17], we filtered out the documents in World and Regional categories because those in Regional are included in other categories and those in World are not written in English. We identified the documents not written in English in other categories and removed them from our dataset.

![Figure 5. Document Distribution over Different Levels in Filtered ODP](image1)

![Figure 6. Category Distribution over Different Levels in Filtered ODP](image2)

We also filtered out the categories with less than two documents because a category should appear in both training data and test data. As a result, 15 top-level categories were used for experiments, Figures 5 and 6 show document and category distributions over different levels, respectively, after the filtering process. The distributions are skewed with their means positioned at the 5th level. Among all the documents, 13,000 were selected for testing and the rest for training.

While the test data over 1.3 million documents were chosen randomly in [17], we opted to select test documents in proportion to the number of documents and categories at each level against the entire distribution to have a better control over a possible imbalance.

\(^2\) See http://rdf.dmoz.org/ for Open Directory Project
Figure 7 shows two distributions in the test data similar to the one in Figure 5. As a result, 1,427,426 and 13,000 documents were used for training and testing, respectively. An inverted index was constructed using Lucene for the training data. In our dataset, about 96.46% of the documents have one category.

6.1.2 Evaluation Metric
In text classification, evaluation is done based on whether a document is classified into one or more correct categories among the entire set of possible target categories. In deep classification, however, the sheer number of target categories in a large hierarchy prohibits the same kind of evaluation because it is too costly to make so many judgments for each test document. Therefore, evaluations are done for each level of the hierarchy with micro-averaged F1 measure [9,17,18]. If a test document is classified to A/B/C/D, which means D with a path from A, the first evaluation is about whether A is correct at the top level, the second one is with B at the second level, and so on.

6.2 Evaluation
6.2.1 Overall Performance Comparison
In order to compare our method against the previously proposed deep classification algorithm [17], we implemented it. Table 1 shows the details of the two algorithms. The best performing case takes top 10 category candidates and employed the category-based search, ancestor-assistant strategy, and trigram language model classifier. For the proposed method, we use the same setup for the search stage with top 10 category candidates and the category-based search strategy. On the other hand, the classification stage adopted the neighbor-assistant strategy and naive Bayes combination modeling with the bias being 0.6. The particular combination achieved the best performance in our experiments.

<table>
<thead>
<tr>
<th>Table 1. Summary of Differences</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Deep Classification</strong></td>
</tr>
<tr>
<td>Top-k</td>
</tr>
<tr>
<td>Search</td>
</tr>
<tr>
<td>Training Data Selection</td>
</tr>
<tr>
<td>Classifier</td>
</tr>
</tbody>
</table>

Figure 8 shows performance comparison between the deep classification algorithm and our proposed method. As can be seen, our method outperforms the baseline (i.e. the deep classification algorithm in [17]) at every level. Our algorithm achieves 0.806 at the top level while the baseline is 0.652. For the deepest level, our algorithm achieves 0.230 which is more than twice as much as 0.089 in the baseline. As a result, our algorithm produces about 85.4% improvement on average over the baseline. The performance improvement was computed like macro-average: the percentage improvements for the nine levels were averaged.

6.2.2 Local vs. Global Information
A key differentiator of the proposed method is the use of global information combined with local information for classification. The parameter lambda (0.0 ≤ λ ≤ 1.0) in NBCC modulates the balance between the amounts of global and location information. A series of experiments were conducted to investigate on the role of the parameter, i.e., that of global information, for performances at different levels. Figure 9 shows performance changes for different lambda values. When lambda is 0, only global information is used, ignoring local information. On the other hand, when lambda is 1, only local information is used as in [17]. The neighbor-assistant method was used for training document selection.

As shown in Figure 9, the best performance is obtained when lambda is set to 0.4 or 0.6 – there is little difference between them. Setting lambda to 0 (global information only) resulted in a considerable drop in performance. As expected, local information alone gave a better performance than global information only. While this suggests that local information is more useful for classification, global information alone can determine the class quite well. This confirms our initial conjecture that selecting the right branch of the tree at the highest level plays an essential role in determining the class of a document.
For location information used for training, we suggested a new strategy named as neighbor-assistant method. The new method was compared against the existing one named ancestor-assistant strategy developed in [17]. In the experiment, the global information as well as the NBCC framework remained the same to see the pure effect of using the new local training data selection strategy. Figure 10 shows the result of comparisons between the two strategies. While neighbor-assistant strategy is slightly better than the ancestor-assistant one, its effect diminishes as the level gets deeper. This is due to the average number of descendants, which becomes smaller as the level gets deeper. Another factor for the marginal improvements seems that the use of global information compensates for the lack of local training data.

Figure 10. Comparison between the Two Local Training Data Selection Strategies

Since the search stage determines the categories to work on, it has a great impact on the final classification results. There are two major reasons why search can go wrong. The first one is that the input document is sometimes too short to be categorized accurately. The documents in ODP contain 11.8 words on average with standard deviation of 4.89 after moving stop words. Although the sizes of individual training documents are also short, we alleviate the problem by using the categories containing multiple documents rather than individual documents as the retrieval targets.

7. CONCLUSION

This paper proposes a novel method for deep classification where a web document is mapped to a large-scale category hierarchy. It is an extension of the narrow-down approach that relies on the initial search for a set of candidate categories in the hierarchy and builds a classifier for the input document for the small number of candidate categories. In order to alleviate the problem of the lack of training documents associated with each category, we propose to use not only the local information surrounding each candidate category node in the hierarchy but also global information from the top level nodes. Since each node at the top level serves as the root of the sub-tree encompassing the category node at hand, such global information can enrich the limited local information. In addition, we propose a new local training data selection strategy that considers the descendants of the candidate node and of its ancestors to enlarge the training document set. Through a series of experiments with the ODP hierarchy, we show that the proposed method outperforms the state-of-the-art method across all the levels from which test documents are chosen. The magnitudes of the improvements range from 24% at top level to 158% at 9th level with an average of 85.4%.

Our preliminary error analysis shows that the initial search has a great impact on the final classification result, setting the upper bound, with increasing success rates as the number of documents retrieved becomes larger. Additional work is planned to find ways to improve search results for the deep classification purposes. A shorter term plan is to do more extensive analysis of the experimental results and increase the level of our understanding about various reasons for misclassifications.

8. ACKNOWLEDGMENTS

This research was partially supported by Microsoft Global RFP Award for ‘Beyond Search – Semantic Computing and Internet Economics’, and by the 2nd phase of Brain Korea 21 sponsored by the Ministry of Education and Human Resources Development, Korea.

9. REFERENCES


