Effective Top-Down Active Learning for Hierarchical Text Classification

Xiao Li[1,2,⋆], Charles X. Ling[1], and Huaimin Wang[2]

[1] Department of Computer Science, The University of Western Ontario
[2] National Laboratory for Parallel & Distributed Processing, National University of Defense Technology
{xli485,cling}@csd.uwo.ca, whm_w@163.com

Abstract. Hierarchical text classification is an important task in many real-world applications. To build an accurate hierarchical classification system with many categories, usually a very large number of documents must be labeled and provided. This can be very costly. Active learning has been shown to effectively reduce the labeling effort in traditional (flat) text classification, but few works have been done in hierarchical text classification due to several challenges. A major challenge is to reduce the so-called out-of-domain queries. Previous state-of-the-art approaches tackle this challenge by simultaneously forming the unlabeled pools on all the categories regardless of the inherited hierarchical dependence of classifiers. In this paper, we propose a novel top-down hierarchical active learning framework, and effective strategies to tackle this and other challenges. With extensive experiments on eight real-world hierarchical text datasets, we demonstrate that our strategies are highly effective, and they outperform the state-of-the-art hierarchical active learning methods by reducing 20% to 40% queries.

Keywords: Active Learning, Hierarchical Text Classification.

1 Introduction

Given documents organized in a meaningful hierarchy (such as a topic hierarchy), it is much easy for users to browse and search the desired documents. Thus, hierarchical text classification is an important task in many real-world applications, which include, for example, news article classification [8], webpage topic classification [3,2,10] and patent classification [5]. In hierarchical text classification, a document is assigned with multiple suitable categories from a predefined hierarchical category space. Different from traditional flat text classification, the assigned categories for each document in the hierarchy have inherited hierarchical relations. For example, in the hierarchy of the Open Directory Project (ODP), one path of the hierarchy includes *Computers* (Comp.) $\rightarrow$ *Artificial Intelligence* (A.I.) $\rightarrow$ *Machine Learning* (M.L.). Any webpage belonging to *M.L.* also belongs to *A.I.* and *Comp.*.

⋆ Partial work was done when Xiao Li was an intern at National Laboratory for Parallel & Distributed Processing, Changsha, China.

In this paper, we study machine learning approaches for building hierarchical classification system. According to [13], the most effective and appropriate approach for building hierarchical classification system is to train a binary classifier on each category of the hierarchy. To train an accurate hierarchical classification system with many categories, usually a very large number of labeled documents must be provided for a large number of classifiers. However, labeling a large amount of documents in a large hierarchy is very time-consuming and costly. This severely hinders the training of accurate hierarchical classification systems.

Active learning has been studied and successfully applied to reduce the labeling cost in binary text classification [15,11,17]. In active learning, in particular the pool-based active learning, the learner intelligently selects the most informative unlabeled example from the unlabeled pool to query an oracle (e.g., human expert) for the label. This can lead a good classification model with a much smaller number of labeled examples, compared to traditional passive learning. Several works have extended binary active learning to multi-class and multi-label text classification [1,4,19]. Basically, they use the one-VS-rest approach to decompose the learning problem to several binary active learning tasks.

However, active learning has not been widely studied for hierarchical text classification. The key question is how to effectively select the most useful unlabeled examples for a *large* number of *hierarchically* organized classifiers. Many technical challenges exist. For example, how should the unlabeled pool be formed for each category in the hierarchy? If not formed properly, the classifier may select many so-called *out-of-domain* examples from the pool. For example, a classifier on *A.I.* is trained under the category of *Comp.*. These examples are called *in-domain* examples for *A.I.*. Examples not belonging to *Comp.* are the so-called *out-of-domain* examples for *A.I.*. If an unlabeled example selected by the classifier for *A.I.* is an *out-of-domain* example, such as a document belonging to *Society*, the oracle will always answer "no", and such a query will be virtually useless, and thus wasted in training the classifier for *A.I.*. Thus, avoiding the *out-of-domain* examples for hierarchical classifiers is very important.

As far as we know, only one work [9] has been published previously on hierarchical active learning. To solve the *out-of-domain* problem, the authors use the prediction of higher-level classifiers to refine the unlabeled pools for lower-level classifiers. In their approach, the quality of the lower-level unlabeled pools depends critically on the classification performance of the higher-level classifiers. However, the authors seemed not to pay enough attention to this important fact, and their methods allow all classifiers to simultaneously select examples to query oracles (see Section 2 for a review). This still leads to a large number of *out-of-domain* queries, as we will show in Section 4.4.

As the hierarchical classifiers are organized based on the top-down tree structure, we believe that a natural and better way to form the unlabeled pools is also in the top-down fashion. In this paper, we propose a novel top-down active learning framework, to effectively form the unlabeled pools, and select the most informative, *in-domain* examples for the hierarchical classifiers. Under our top-down active learning framework, we discuss effective strategies to tackle various challenges encountered. With extensive experiments on eight real-world hierarchical text datasets, including the RCV1-V2 and ODP datasets, we demonstrate that our method is highly effective, and it outperforms the state-of-the-art hierarchical active learning methods including [9] by reducing 20% to 40% queries.

2 Previous Works

To our best knowledge, only one work [9] has been published previously in active learning for hierarchical text classification. We call it the *parallel* active learning framework. In their approach, at each iteration of active learning (see Figure 1), the classifiers for all categories *independently* and *simultaneously* query the oracles for the corresponding labels. To avoid selecting the *out-of-domain* examples, they use the prediction of higher-level classifiers to refine the unlabeled pools for lower-level classifiers. Specifically, an unlabeled example will be added into the lower-level unlabeled pools only if its predictions from all the ancestor classifiers are positive.

A drawback of their approach is that they do not consider the hierarchical dependence of classification performance of the classifiers in their framework but allow all classifiers to *simultaneously* form the pools and select examples to query oracles. Considering a typical running iteration of their approach (see Figure 1). If the quality of the unlabeled pool $\mathcal{U}_{comp}$ (formed by the classifier for *Comp.*) is not good, possibly many *out-of-domain* examples (e.g., examples from *Society*) may still be selected by the classifiers for *A.I.*. This will lead to a large number of *out-of-domain* (wasted) queries, as we will show in Section 4.4.

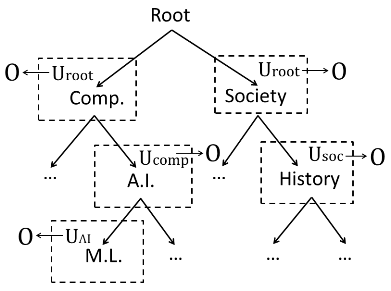
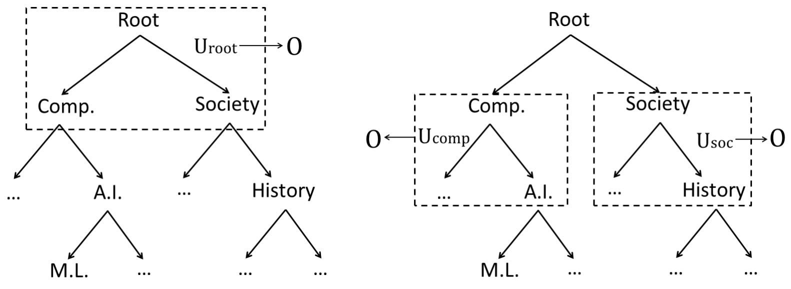


Fig. 1. A typical iteration of the parallel active learning framework. Multiple active learning processes (represented by dashed windows) are simultaneously conducted. U denotes the unlabeled pool and O denote the oracle. The horizontal arrows mean querying the oracle while the down arrows mean building the unlabeled pools.

How can we effectively solve the *out-of-domain* problem and the other challenges to improve active learning in hierarchical text classification? As the hierarchical classifiers are organized based on the top-down tree structure, we believe that a natural and better way to do active learning in hierarchical text classification is also in the top-down fashion. In the next section, we propose a new top-down active learning framework for hierarchical text classification to effectively tackle these challenges.

3 Top-Down Hierarchical Active Learning Framework

In this section, we propose our top-down hierarchical active learning framework. Different to the parallel framework which simultaneously forms the unlabeled pools for all categories, our top-down approach forms the unlabeled pools in the top-down fashion. We use Figure 2 to describe our basic idea.



(a) The top level learning stage. **(b)** The second level learning stage.

Fig. 2. Examples of two typical active learning stages in the top-down active learning framework. Only partial nodes in the hierarchy are allowed to do active learning. The notations in this figure follow Figure 1.

In Figure 2a, we start active learning at the top level of hierarchy. The top-level classifiers for *Comp.* and *Society* select examples from the global unlabeled pool $\mathcal{U}_{root}$ to query the oracle for the labels of top-level categories. The answered examples from the oracle will be used to form the unlabeled pools $\mathcal{U}_{comp}$ and $\mathcal{U}_{soc}$. After the top-level classifiers are well trained (estimated by our stopping strategy. see in Section 3.2), we start active learning in the second level. In Figure 2b, the second-level classifiers for A.I. (or History) and its sibling categories select examples from the unlabeled pool $\mathcal{U}_{comp}$ (or $\mathcal{U}_{soc}$) to query the oracle. As the examples in $\mathcal{U}_{comp}$ (or $\mathcal{U}_{soc}$) have true labels of *Comp.* (or *Society*) which are answered by the oracle, we can ensure that the second-level classifiers will not select *any out-of-domain* examples.

Comparing Figure 1 and Figure 2, we can see that the main difference between the parallel framework and our top-down framework is which nodes are chosen to do active learning (the dashed windows in both figures) at each iteration. The parallel framework chooses all the nodes while our top-down framework only chooses a subset of appropriate nodes in the top-down fashion. We call the set of those nodes as *working set*, denoted by $\mathcal{W}$. We present the pseudo code of our top-down active learning framework.

```
Input: Query budget $B$
Output: Classifiers for all nodes
repeat
    Add the root nodes $n_0$ into $\mathcal{W}$;
    repeat
        Select examples from $\mathcal{U}_n$ to query oracles and update children classifiers
        for each node $n$ in $\mathcal{W}$ until its stopping criteria is satisfied;
        Form the unlabeled pools for the children nodes of the finished nodes;
        Replace the finished nodes in $\mathcal{W}$ with their children nodes;
    until $\mathcal{W}$ is empty;
until $B = 0$;
```

For our top-down active learning framework, two critical challenges need to be resolved for effective active learning. The first challenge is that the unlabeled pools may be too small. We use the examples answered by the oracle to form the unlabeled pools, and they can be too small for lower-level classifiers to learn effectively. The problem may become worse when active learning is applied to even lower-level categories. The second challenge is how do we stop learning as it is critical for the effective scheduling of active learning in different levels. We will tackle the two challenges in the following subsections.

3.1 Dual-Pool Strategy

For the second and lower-level nodes, we need to form the unlabeled pools that are large enough but have few *out-of-domain* examples. In this section, we propose a novel dual-pool strategy to enlarge the unlabeled pools. Two different unlabeled pools will be built: the *answered pool* and the *predicted pool.*

Answered Pool. Our top-down active learning framework schedules the nodes to query the oracle from the top level to the bottom level. For a node (category) in the working set, we ask oracles for the labels of its children categories. For a child category c, among the answered examples from the oracle, the *positive* examples of c will be used to form the unlabeled pool for the children categories of c. The *negative* examples will not be used as they are already *out-of-domain.* By doing so, we can ensure that no *out-of-domain* examples will be selected into the unlabeled pools of children categories. We call such a pool the *answered pool* and use $\mathcal{U}^a$ to denote it.

Predicted Pool. The quality of the answered pool $\mathcal{U}^a$ is perfect. However, as the size of $\mathcal{U}^a$ depends on the positive class ratio of the ancestor nodes, it could be very slow to accumulate enough examples. Thus, we can also use the prediction of the higher-level classifiers to enlarge the unlabeled pools. Although this method is also used in the parallel framework (see Section 2), it should be noted that when we build the lower-level unlabeled pools, the higher-level classifiers are already assumed to be well-trained. The prediction of higher-level classifiers would be accurate. Thus, the risk of introducing *out-of-domain* examples would be much smaller than the parallel framework. We call the pool built by this method as *predicted pool*, denoted by $\mathcal{U}^p$.

Refiltering Dual Pools. We have two unlabeled pools for each node n_i in our top-down framework, i.e., the answered pool $\mathcal{U}_i^a$ and the predicted pool $\mathcal{U}_i^p$. When we select a batch of examples to query the oracle, a natural question is how do we allocate the batch of queries to each pool? On one hand, the quality of $\mathcal{U}_i^a$ is perfect but the uncertain (useful) examples maybe be too few due to the small pool size; on the other hand, more useful examples may exist in the larger predicted pool $\mathcal{U}_i^p$ but we may take the risk of selecting the *out-of-domain* examples. To balance the tradeoff, we propose a *refiltering strategy* for allocating the queries to both $\mathcal{U}_i^a$ and $\mathcal{U}_i^p$.

Our basic idea is to filter out the certain examples from the pools before we allocate the batch of queries. Specifically, given the batch size M, we firstly filter out the certain examples from both $\mathcal{U}_i^a$ and $\mathcal{U}_i^p$ to generate two small candidate pools $\mathcal{C}_i^a$ and $\mathcal{C}_i^p$. The filtering threshold will be empirically tuned in our experiments (See Section 4.4). As the examples in $\mathcal{C}_i^a$ are all perfect (answered by oracle) and uncertain (worthy to learn), we put more queries into the perfect candidate pool $\mathcal{C}_i^a$ by allocating $\min\{|\mathcal{C}_i^a|, M\}$ queries. The rest of queries will be allocated to $\mathcal{C}_i^p$.

3.2 Stopping Strategy

An important factor of our top-down hierarchical active learning framework is knowing when to stop learning for the nodes in the working set. In other words, how to estimate if the classifiers are well-trained or not? A heuristic approach is to estimate the classification performance by cross-validation. However, from our pilot experiments, such method is quite unstable due to the small size of the labeled examples in active learning.

In this paper, we adopt a simple yet effective approach to stop learning. Simply speaking, if no uncertain examples can be further selected from the candidate pools, we stop learning. This is reasonable as querying very certain examples can not improve the classification performance [20]. In our top-down framework, this strategy can be implemented by checking the size of the two candidate pools $\mathcal{C}^a$ and $\mathcal{C}^p$. If both pools are empty, that means all the examples in the unlabeled pools are very certain, we stop learning.[1]

To summarize, in this section, we propose our top-down hierarchical active learning framework with several strategies to tackle the *out-of-domain* problem and the other challenges encountered. In the next section, we will conduct extensive experiments to verify the effectiveness of our framework.

4 Experiments

In this section, we conduct extensive empirical studies to evaluate our top-down hierarchical active learning framework compared to the state-of-the-art hierarchical active learning approaches.

4.1 Datasets

We use eight real-world hierarchical text datasets in our experiments. The first three datasets (20 Newsgroup, OHSUMED and RCV1-V2) are common benchmark datasets for evaluation of text classification methods. The other five datasets are webpages collected from Open Directory Project (ODP).

The first dataset is *20 Newsgroups*[2], a collection of news articles partitioned evenly across 20 different newsgroups. We manually group these categories into a

[1] For the root node which selects examples from the very large global unlabeled pool, this stopping strategy could be very slow. Thus, we empirically set 25% remained budget as the query limit for the root node.

[2] `http://people.csail.mit.edu/jrennie/20Newsgroups/`

meaningful three-level hierarchy. The second dataset is *OHSUMED*[3], a clinically-oriented MEDLINE dataset. We use the subcategory *heart diseases* which is also used by [7,12]. The third dataset is *RCV1-V2* [8], a news archive from Reuters. We use the 23,149 documents from the topic classification task in our experiments.[4] The other five datasets are webpages collected from ODP. ODP is a web directory with a complex topic hierarchy. In our experiments, we focus on a subset of the webpages extracted from the Science subtree.[5] The original Science subtree has more than 50 subcategories. We choose five subcategories closely related to the academic disciplines.[6] They are *Astronomy*, *Biology*, *Chemistry*, *Earth Sciences* and *Math*.

For each dataset, we use bag-of-words model to represent documents. Each document is represented by a vector of term frequency. We use Porter Stemming to stem each word and remove the rare words occurring less than three times. Small categories which have less than ten documents are also removed. After the preprocessing, we give the detailed statistics of the datasets in Table 1.

Table 1. The statistics of the datasets. Cardinality is the average categories per example (multi-label).

Dataset	Examples	Features	Nodes	Cardinality	Height
20 Newsgroup	18,774	61,188	27	2.20	3
OHSUMED	16,074	12,427	86	1.92	4
RCV1-V2	23,149	47,152	96	3.18	4
Astronomy	3,308	54,632	34	1.91	4
Biology	17,450	148,644	108	3.03	4
Chemistry	4,228	56,767	34	1.44	4
Earth Sciences	5,313	71,756	58	2.16	4
Math	11,173	108,559	107	1.93	4

4.2 Performance Measure

In this paper, we use the hierarchical F-measure [16,13,9], a popular performance measure in hierarchical text classification, to evaluate the performance of hierarchical classification methods. It is defined as,

$$hF = \frac{2 \times hP \times hR}{hP + hR} \quad where \quad hP = \frac{\sum_i |\hat{P}_i \bigcap \hat{T}_i|}{\sum_i |\hat{P}_i|}, \quad hR = \frac{\sum_i |\hat{P}_i \bigcap \hat{T}_i|}{\sum_i |\hat{T}_i|} \tag{1}$$

where hP is the hierarchical precision and hR is the hierarchical recall. $\hat{P}_i$ is the hierarchical categories predicted for test example x_i while $\hat{T}_i$ is the true categories of x_i.

[3] `http://ir.ohsu.edu/ohsumed/`

[4] `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`

[5] It can be freely downloaded at `http://olc.ijs.si/dmozReadme.html`.

[6] Most of the other subcategories are A-Z index lists and non-academic topics (e.g., Publications and Conferences).

4.3 Experiment Configuration

We adopt the hierarchical SVMs [10,14,18] as the base learner. On each category, a linear SVM classifier is trained to distinguish its sibling categories under the same parent category. We use LIBLINEAR [6] as the implementation of linear SVM. Following the configuration of [9], we set up the penalty C as 1,000 and the cost coefficient w as the ratio of negative examples in the training set. Other parameters of LIBLINEAR are set to the default values.

We compare our top-down framework with the parallel framework [9] and the baseline random approach. We use the *average uncertainty* [4] as the informativeness measure. It measures the example based on the average uncertainty among all children classifiers under the same parent. For the parallel framework, we choose the *uncertainty sampling* [15] which is also used in their experiments. For both approaches, the uncertainty of example is measured by the absolute SVM margin score. For the random approach, we simply select the examples randomly from the global unlabeled pool.

We set up the total query budget as 1000. The active learning experiment is decomposed into several iterations. In each iteration, each node in the working set selects M examples to query the oracle. Similar to [9], the batch size M is set as the logarithm of the unlabeled pool size on each category. We use the simulated oracle in our experiments. When receiving an query, the oracle replies the true labels for all its subcategories. It should be noted that in [9], each query only returns one label. To make a fair comparison, we also return the labels of all the subcategories for the parallel framework and the random approach.

To avoid the impact of randomness, we use 10-fold cross validation to evaluate the performance of active learning approaches. Specifically, when conducting active learning experiments on each dataset, we randomly split the dataset into 10 subsets with equal size. Of the 10 subsets, one set is retained as testing data. For the remain nine sets, we randomly sample 0.1% data as the labeled set. The remaining examples will be used as the unlabeled pool. The active learning experiments are then repeated 10 times. The final results are averaged over the 10 runs and accompanied by the error margins with 95% confidence intervals.

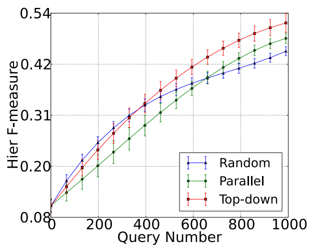
4.4 Experimental Results on Benchmark Datasets

Before the experiments, we setup the parameters for our top-down framework. We need to decide a proper uncertainty threshold to filtering out the certain examples (see dual-pool strategy in Section 3.1). As the SVM margin score based uncertainty is not comparable, we normalize it by the function $g(f) = exp(-\frac{f^2}{0.01})$ ($0 < g \leq 1$) where f is the SVM margin score. We compare the uncertainty thresholds in different values from 0.1, 0.2, to 0.9 on the RCV1-V2 dataset. We find that generally the larger the threshold is, the better the performance is.[7] Thus, we use 0.9 as the uncertainty threshold in our experiments.

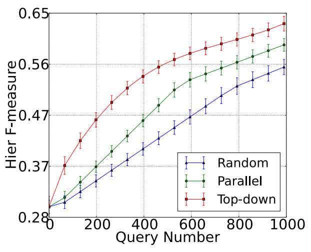
Firstly, we discuss the experimental results on the three benchmark datasets (20 Newsgroup, OHSUMED and RCV1-V2). Figure 3 shows the performance

[7] Due to page limit, the figure is omitted.

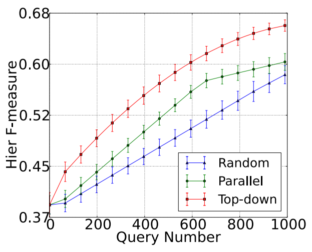
curves of hierarchical F-measure averaging over 10 runs. We can see that our top-down approach (framework) outperforms the parallel approach and the random approach significantly on all datasets. Specifically, on the OHSUMED and RCV1-V2 datasets, the performance curves of our top-down approach dominate the parallel approach and the random approach throughout the whole iterations. On the 20 Newsgroup dataset, surprisingly, during the earlier stage of active learning (before 400 queries), we observe the overlap of performance curves of our top-down approach and the random approach. The parallel approach performs even worse. This could be due to the poor initial classification performance (smaller than 0.1). However, after around 500 queries, our approach starts to outperform the random approach and the parallel approach and keeps the dominant margin till the end.



(a) 20 Newsgroup



(b) OHSUMED



(c) RCV1-V2

Fig. 3. Hierarchical F-measure on the 20 Newsgroup, OHSUMED and RCV1-V2 datasets

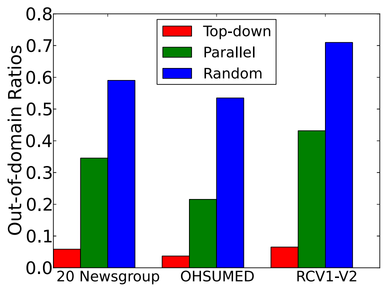


Fig. 4. The *out-of-domain* ratios of the queries on the 20 Newsgroup, OHSUMED and RCV1-V2 datasets

We examine the ratio of *out-of-domain* queries. Figure 4 shows the average *out-of-domain* ratios on the three datasets. We can see that our top-down approach has a huge reduction of the *out-of-domain* queries. Among the three datasets, our top-down approach issues less than 10% *out-of-domain* queries. By analyzing the experiment logs of our top-down approach, we discovery that for the second and the lower-level, on average about 40% queries are allocated to the answered pools (see Section 3.1). As the labels in the answered pools are given by the oracle, the quality of the selected examples is perfect. Thus, no *out-of-domain* examples will be selected. The observed few *out-of-domain* examples only occur in the predicted pools. The low ratio also indicates that the predicted pools built by our dual-pool strategy are much more accurate than the parallel framework. This explains why our top-down active learning approach is more effective than the parallel approach.

We also study how many queries can be saved by our top-down approach. For the three approaches, we record their best performance and the queries needed in Table 2. We find that to achieve the best performance of the parallel approach, our top-down approach needs much fewer queries. About 20% to 37% queries can be saved. For example, on the RCV1-V2 dataset, the parallel approach needs 1,000 queries to achieve 0.606 hierarchical F-measure, while our top-down approach only requires 630 queries. Thus, $(1000 - 630)/1000 = 37\%$ queries are saved. Compared to the random approach, the query reduction is even more significant (about 30% to 56%). It clearly indicates that our top-down approach is more effective in reducing the queries than the parallel approach and the baseline random approach.

Table 2. The best hierarchical F-measure with needed queries on the 20 Newsgroup, OHSUMED and RCV1-V2 datasets. The value in the bracket is the relative query reduction.

	Method	Hier F1	Random	Parallel	Top-down
20 Newsgroup	Random	0.455	1000	850 (15%)	700 (30%)
	Parallel	0.483		1000	800 (20%)
	Top-down	0.518			1000
OHSUMED	Random	0.552	1000	720 (28%)	440 (56%)
	Parallel	0.591		1000	680 (33%)
	Top-down	0.630			1000
RCV1-V2	Random	0.587	1000	660 (34%)	490 (51%)
	Parallel	0.606		1000	630 (37%)
	Top-down	0.661			1000

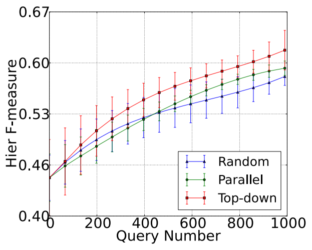
4.5 Experimental Results on ODP Datasets

In the following experiments, we compare the performance of the three approaches on five ODP datasets. From Figure 5, we find that on all datasets, our top-down approach performs consistently better than both the parallel approach and the random approach. The largest improvement occurs on the Math dataset where our top-down approach saves 40% queries to achieve the best performance of the parallel approach.[8] By analyzing the *out-of-domain* ratio in Figure 5f, we find that our top-down approach reduces the ratio of *out-of-domain* queries by 32% on the Math dataset compared to the parallel approach. The similar pattern can also be observed on the Biology and Earth Sciences datasets where about 32% and 23% *out-of-domain* queries can be saved. For the Astronomy and Chemistry datasets, we can see that the parallel approach makes less than 20% *out-of-domain* ratios. This can explain why our top-down approach performs only slightly better than the parallel approach on the Astronomy and Chemistry datasets. However, on some of the ODP datasets, the performance curves of the parallel approach have an obvious large overlap with

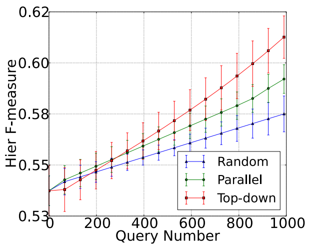
[8] The top-down approach requires 600 queries to achieve 0.4 hierarchical F-measure of the parallel approach which requires 1,000 queries. The saving is $(1000 - 600)/1000=40\%$.

the random approach, while our top-down approach always outperforms the two approaches at the end of active learning.

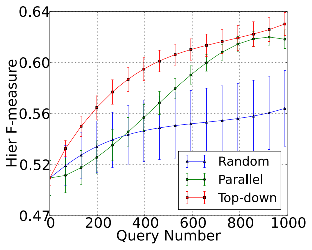
To summarize, from our extensive experiments on eight real-world hierarchical text datasets, we empirically demonstrate that our top-down active learning framework is more effective than the state-of-the-art active learning approaches for hierarchical text classification.



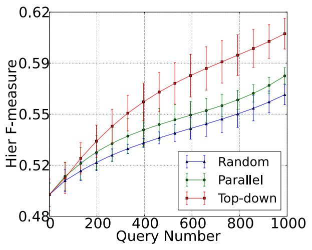
(a) Astronomy



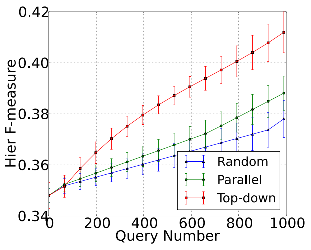
(b) Biology



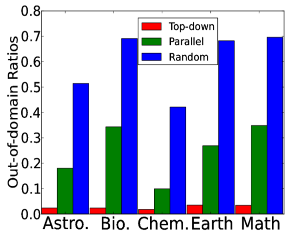
(c) Chemistry



(d) Earth Sciences



(e) Math



(f) Out-of-domain Ratios

Fig. 5. Hierarchical F-measure and the ratios of *out-of-domain* queries on the ODP datasets

5 Conclusion and Future Work

In this paper, we study the problem of active learning for hierarchical text classification. A major challenge for effective hierarchical active learning is to form the unlabeled pools to avoid the so-called out-of-domain queries. Previous state-of-the-art approaches tackle this challenge by simultaneously forming the unlabeled pools on all the categories of the hierarchy regardless of the inherited hierarchical dependence of classifiers. In this paper, we propose a novel top-down hierarchical active learning framework which utilizes the top-down tree structure to form the unlabeled pools. Under our framework, we propose several effective strategies to tackle the out-of-domain problem and the other challenges encountered. With extensive experiments on eight real-world hierarchical text datasets, we demonstrate that our top-down framework is highly effective, and it outperforms the state-of-the-art hierarchical active learning methods by reducing 20% to 40% queries.

In our future work, we plan to use crowdsourcing for hierarchical active learning in real-world applications, such as constructing the hierarchical classifiers for search engines with hierarchy.

Acknowledgments. We thank the anonymous reviewers for their valuable suggestions. This work was partially supported by the National High Technology Research and Development Program of China (Grant No. 2012AA011201).

References

1. Brinker, K.: On active learning in multi-label classification. In: From Data and Information Analysis to Knowledge Engineering, pp. 206–213 (2006)
2. Ceci, M., Malerba, D.: Classifying web documents in a hierarchy of categories: a comprehensive study. J. Intell. Inf. Syst. 28, 37–78 (2007)
3. Dumais, S., Chen, H.: Hierarchical classification of web content. In: SIGIR 2000, pp. 256–263. ACM (2000)
4. Esuli, A., Sebastiani, F.: Active learning strategies for multi-label text classification. In: Boughanem, M., Berrut, C., Mothe, J., Soule-Dupuy, C. (eds.) ECIR 2009. LNCS, vol. 5478, pp. 102–113. Springer, Heidelberg (2009)
5. Fall, C.J., Törcsvári, A., Benzineb, K., Karetka, G.: Automated categorization in the international patent classification. SIGIR Forum 37(1), 10–25 (2003)
6. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: Liblinear: A library for large linear classification. J. Mach. Learn. Res. 9, 1871–1874 (2008)
7. Lam, W., Ho, C.Y.: Using a generalized instance set for automatic text categorization. In: SIGIR 1998, pp. 81–89 (1998)
8. Lewis, D.D., Yang, Y., Rose, T.G., Li, F.: Rcv1: A new benchmark collection for text categorization research. J. Mach. Learn. Res. 5, 361–397 (2004)
9. Li, X., Kuang, D., Ling, C.X.: Active learning for hierarchical text classification. In: Tan, P.-N., Chawla, S., Ho, C.K., Bailey, J. (eds.) PAKDD 2012, Part I. LNCS, vol. 7301, pp. 14–25. Springer, Heidelberg (2012)
10. Liu, T.Y., Yang, Y., Wan, H., Zeng, H.J., Chen, Z., Ma, W.Y.: Support vector machines classification with a very large-scale taxonomy. SIGKDD Explor. Newsl. 7, 36–43 (2005)
11. Roy, N., McCallum, A.: Toward optimal active learning through sampling estimation of error reduction. In: ICML 2001, pp. 441–448 (2001)
12. Ruiz, M.E., Srinivasan, P.: Hierarchical neural networks for text categorization (poster abstract). In: SIGIR 1999, pp. 281–282 (1999)
13. Silla Jr., C.N., Freitas, A.A.: A survey of hierarchical classification across different application domains. Data Min. Knowl. Discov. 22, 31–72 (2011)
14. Sun, A., Lim, E.P.: Hierarchical text classification and evaluation. In: ICDM 2001, pp. 521–528 (2001)
15. Tong, S., Koller, D.: Support vector machine active learning with applications to text classification. J. Mach. Learn. Res. 2, 45–66 (2002)
16. Verspoor, K., Cohn, J., Mniszewski, S., Joslyn, C.: Categorization approach to automated ontological function annotation. In: Protein Science, pp. 1544–1549 (2006)
17. Xu, Z., Yu, G., Tresp, V., Xu, X., Wang, J.: Representative sampling for text classification using support vector machines. In: Sebastiani, F. (ed.) ECIR 2003. LNCS, vol. 2633, pp. 393–407. Springer, Heidelberg (2003)
18. Xue, G.R., Xing, D., Yang, Q., Yu, Y.: Deep classification in large-scale text hierarchies. In: SIGIR 2008, pp. 619–626 (2008)
19. Yang, B., Sun, J.T., Wang, T., Chen, Z.: Effective multi-label active learning for text classification. In: KDD 2009, pp. 917–926 (2009)
20. Zhu, J., Wang, H., Hovy, E., Ma, M.: Confidence-based stopping criteria for active learning for data annotation. ACM Trans. Speech Lang. Process 6(3), 3:1–3:24 (2010)