Active Learning for Hierarchical Text Classification

Xiao Li, Da Kuang and Charles X. Ling

Department of Computer Science,
The University of Western Ontario, London, Ontario, N6A 5B7, Canada
{xli485,dkuang,cling}@csd.uwo.ca

Abstract. Hierarchical text classification plays an important role in many real-world applications, such as webpage topic classification, product categorization and user feedback classification. Usually a large number of training examples are needed to build an accurate hierarchical classification system. Active learning has been shown to reduce the training examples significantly, but it has not been applied to hierarchical text classification due to several technical challenges. In this paper, we study active learning for hierarchical text classification. We propose a realistic multi-oracle setting as well as a novel active learning framework, and devise several novel leveraging strategies under this new framework. Hierarchical relation between different categories has been explored and leveraged to improve active learning further. Experiments show that our methods are quite effective in reducing the number of oracle queries (by 74% to 90%) in building accurate hierarchical classification systems. As far as we know, this is the first work that studies active learning in hierarchical text classification with promising results.

1 Introduction

Hierarchical text classification plays an important role in many real-world applications, such as webpage topic classification, product categorization and user feedback classification. Due to the rapid increase of published documents (e.g., articles, patents and product descriptions) online, most of the websites (from Wikipedia and Yahoo! to the small enterprise websites) classify their documents into a predefined hierarchy (or taxonomy) for easy browsing. As more documents are published, more human efforts are needed to give the hierarchical labels of the new documents. It dramatically increases the maintenance cost for those organization or companies. To tackle this problem, machine learning techniques such as hierarchical text classification can be utilized to automatically categorize new documents into the predefined hierarchy.

Many approaches have been proposed to improve the performance of hierarchical text classification. Different approaches have been proposed in terms of how to build the classifiers [2, 19], how to construct the training sets [1, 6] and how to choose the decision thresholds [15, 1] and so on. As a hierarchy may contain hundreds or even tens of thousands of categories, those approaches often

require a large number of labeled examples for training. However, in real-world applications, such as webpage topic classification, the *labeled* documents are very limited compared to the total number of *unlabeled* documents. Obtaining a large size of labeled documents for training requires great amount of human efforts. How can we build a reliable hierarchical classifier from a relatively small number of examples? Can we reduce the number of labeled examples significantly?

To tackle the lack of labeled examples, active learning can be a good choice [16, 12, 18]. The idea of active learning is that, instead of passively receiving the training examples, the learner actively selects the most "informative" examples for the current classifier and gets their labels from the oracle (i.e., human expert). Usually, those most informative examples can benefit the classification performance most. Several works have successfully applied active learning in text classification [16, 5, 20]. However, to our best knowledge, no previous works have been done in hierarchical text classification with active learning due to several technical challenges. For example, as a large taxonomy can contain thousands of categories, it is impossible to have one oracle to provide all labels. Thus, similar to DMOZ[1], multiple oracles are needed. What would be a realistic setting for multiple oracles for active learning in hierarchical text classification? How can we leverage the hierarchical relation to further improve active learning?

In this paper, we study how active learning can be effectively applied to hierarchical text classification so that the number of labeled examples (or oracle queries) needed can be reduced significantly. We propose a new setting of multiple oracles, which is currently in use in many real-world applications (e.g., DMOZ). Based on this setting, we propose an effective framework for active learning in hierarchical text classification. Moreover, we explore how to utilize the hierarchical relation to further improve active learning. Accordingly, several leveraging strategies and heuristics are devised. According to our experiments, active learning under our framework significantly outperforms the baseline learner, and the additional strategies further enhance the performance of active learning for hierarchical text classification. Compared to the best performance of the baseline hierarchical learner, our best strategy can reduce the number of oracle queries by 74% to 90%.

2 A Novel Multi-Oracle Setting

When active learning is applied to text classification, as far as we know, all previous works (e.g., [5, 20]) explicitly or implicitly assume that given a document that might be associated with multiple labels, there always exist oracles who can perfectly answer all labels. In hierarchical text classification, it is very common that the target hierarchy has a large number of categories (e.g., DMOZ has over one million categories) across various domains, and thus it is unrealistic for one oracle (expert) to be "omniscient" in everything. For example, an expert in "Business" may have less confidence about "Computer", and even less about

[1] It is often called Open Directory Project (`http://www.dmoz.org`).

"Programming". If the expert in "Business" has to label "Programming", errors can occur. Such error introduces noise to the learner.

Therefore, it is more reasonable to assume that there are multiple oracles who are experts in different domains. Each oracle only gives the label(s) related to his or her own domains. Thus, the labels provided by multiple oracles will be more accurate and reliable than the labels given by only one oracle. Although previous works have studied active learning with multiple oracles [4, 10], as far as we know, their settings are quite different from ours as their oracles provide labels for all examples for only one category, while in our case, different oracles provide labels for examples in different categories in the hierarchy.

Our setting of multiple oracles is actually implemented in DMOZ. As far as we know, DMOZ holds a large number of categories. Each category is generally maintained by at least one human editor whose responsibility is to decide whether or not a submitted website belongs to that category.[2] We adopt the similar setting of DMOZ. In our setting, each category in the hierarchy has one oracle, who decides solely if the selected document belongs to the current category or not (by answer "Yes" or "No").

3 A New Framework of Hierarchical Active Learning

In this paper, we mainly discuss pool-based active learning where a large pool of unlabeled examples is available for querying oracles. Figure 1 shows the basic idea of our hierarchical active learning framework. Simply speaking, at each iteration of active learning, classifiers on different categories *independently* and *simultaneously* select the most informative examples from the unlabeled pool for themselves, and ask the oracles on the corresponding categories for the labels. The major steps of our hierarchical active learning algorithm are as follows:

1. We first train a binary classifier (C) on each category to distinguish it from its sibling categories. The training set (D^L) is constructed by using the positive examples from the training set of the parent category [14].[3]
2. Then, we construct the local unlabeled pool (D^U) for each classifier (see Section 3.1), select the most informative examples from the local unlabeled pool for that classifier, and query the corresponding oracle for the labels.
3. For each query, the oracle returns "Yes" or "No" to indicate whether the queried example belongs to that category or not. Based on the answers, the classifier updates its classification model (see Section 3.2).
4. This process is executed simultaneously on all categories at each iteration and repeats until the terminal condition is satisfied.

There are two key steps (step two and three) in the algorithm. In step two, we introduce the local unlabeled pool to avoid selecting *out-of-scope* (we will define it later) examples. In step three, we tackle how to leverage the oracle answers in the hierarchy. We will discuss them in the following subsections.

[2] See `http://www.dmoz.org/erz/` for DMOZ editing guidelines.
[3] On the root of hierarchy tree, every example is positive.

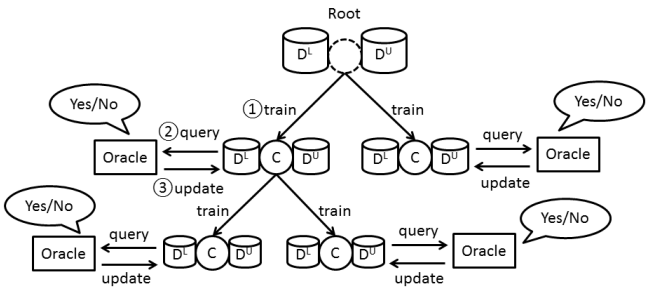

Fig. 1. The hierarchical active learning framework. The typical active learning steps are numbered 1, 2, 3 in the figure.

3.1 Unlabeled Pool Building Policy

From step one of our algorithm, we know that the training examples for a deep category (say c) must belong to its ancestor categories. However, it is likely that many unlabeled examples do not belong to the ancestor categories of c. We define those examples as *out-of-scope* examples. If those out-of-scope examples are selected by c, we may waste a lot of queries. Thus, instead of using one shared unlabeled pool [5] for all categories, we construct a local unlabeled pool on each of the categories. To filter out these out-of-scope examples, we use the predictions of the ancestor classifiers to build the local unlabeled pool. Specifically, given an unlabeled example x and a category c, only if all the ancestor classifiers of c predict x as positive, then we will place x into the local unlabeled pool of c.

3.2 Leveraging Oracle Answers

For the two answers ("Yes" or "No") from oracles, there are several possible ways to handle them. We give a brief overview here and discuss the detailed strategies in Section 5.

If the answer is "Yes", we can simply update the training set by directly including the queried example as a positive example. To better leverage the hierarchical relation, we can even add the positive example to all the ancestor categories. Furthermore, since the positive example is possibly a negative example on some of the sibling categories, we may consider including it as a negative example to the sibling categories.

If the answer is "No", we can not simply add the example as a negative example, since we don't know whether the queried example actually belongs to the ancestor categories. Thus, we could simply discard the example. Alternatively, we can also query the oracle on the parent category to see if the example belongs to the parent category, but the extra query may be wasted if the answer is "No".

In the following parts, we will first present our experimental configuration, and then empirically explore whether our framework can be effectively applied

to hierarchical classification and whether different strategies described above can indeed improve active learning.

4 Experimental Configuration

4.1 Datasets

We utilize four real-world hierarchical text datasets (20 Newsgroups, OHSUMED, RCV1 and DMOZ) in our experiments. They are common benchmark datasets for evaluation of text classification methods. We give a brief introduction of the datasets. The statistic information of the four datasets is shown in Table 1.

Table 1. The statistic information of the four datasets. Cardinality is the average number of categories per example (i.e., multi-label datasets).

Dataset	Features	Examples	Categories	Levels	Cardinality
20 Newsgroups	61,188	18,774	27	3	2.202
OHSUMED	12,427	16,074	86	4	1.916
RCV1	47,236	23,049	96	4	3.182
DMOZ	92,262	12,735	91	3	2.464

The first dataset is *20 Newsgroups*[4]. It is a collection of newsgroup documents partitioned evenly across 20 different newsgroups. We group these categories based on subject matter into a three-level topic hierarchy which has 27 categories. The second dataset is *OHSUMED*[5]. It is a clinically-oriented MEDLINE dataset with a hierarchy of twelve levels. In our experiments, we only use the sub-hierarchy under subcategory "heart diseases" which is well-studied and usually taken as a benchmark dataset for text classification [8, 13]. The third dataset is *RCV1* [9]. It includes three classification tasks: topic, industrial and regional classification. In our experiments, we focus on the topic classification task.[6] The last dataset is *DMOZ*. It is a human-edited web directory with webpages manually organized into a complex hierarchy. *DMOZ* is extracted from a sub collection rooted at "Science" and it has three-level category hierarchy.[7]

4.2 Performance Measure

To evaluate the performance in hierarchical classification, we adopt the hierarchical F-measure, which has been widely used in hierarchical classification for evaluation [17, 3, 14]. The definition the hierarchical F-measure is as follows,

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

[4] http://people.csail.mit.edu/jrennie/20Newsgroups/
[5] http://ir.ohsu.edu/ohsumed/
[6] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/
[7] http://olc.ijs.si/dmozReadme.html

where hP and hR are the hierarchical precision and the hierarchical recall, $\hat{P}_i$ is the set consisting of the most specific categories predicted for test example i and all its (their) ancestor categories and $\hat{T}_i$ is the set consisting of the true most specific categories of test example i and all its (their) ancestor categories.[14]

4.3 Active Learning Setup

In our experiment, linear Support Vector Machine (SVM) is used as the base classifier on each category in the hierarchy, since the high dimensionality of text data usually results in the dataset being linearly separable [16]. Specifically, LIBLINEAR [7] package is used as the implementation of linear SVM. For LIBLINEAR, there are primarily two parameters C and W that will affect the performance. C is the penalty coefficient for training errors and W balances the penalty on the two classes. In our experiment, we set $C = 1000$ and W as the negative class proportion. For example, if the class ratio of positive and negative class in the training set is 1:9, then $W = 0.9$. The purpose is to give more penalty to the error on the minority class.

For active learning, due to the simplicity and effectiveness of *Uncertainty Sampling*[8], we adopt uncertainty sampling as the strategy to select the informative examples from the unlabeled pool. It should be noted that our hierarchical active learning framework is independent of the specific active learning strategy. Other strategies, such as expected error reduction [12] and representative sampling [18] can also be used. We will study them in the future.

We split all the four datasets into labeled (1%), unlabeled (89%) and testing (10%) parts. As we already know the labels of unlabeled examples, we will use the simulating oracles instead of the real human oracles (experts). We set a query limit (see Section 5.1). The training process is decomposed into a sequence of iterations. In each iteration, each category simultaneously selects a fixed number of examples[9] from its local unlabeled pool and queries the oracles (one query will be consumed when we ask one oracle for one label). After each category updates its training set, we recompute the parameter W and update the classification model. The entire training process terminates when the number of queries consumed exceeds the query limit. To reduce the randomness impact of the dataset split, we repeat this active learning process for 10 times. All the results (curves) in the following experiments are averaged over the 10 independent runs and accompanied by error bars indicating the 95% confidence interval.

5 Empirical Study

In this section, we will first experimentally study the standard version of our active learning framework for hierarchical text classification, then propose several improved versions and compare them with the previous version.

[8] Uncertain sampling in active learning selects the unlabeled example that is closest to the decision boundary of the classifier.

[9] We heuristically use logarithm of the unlabeled pool size to calculate the number of selected examples for each category.

5.1 Standard Hierarchical Active Learner

In order to validate our active learning framework, we will first compare its standard version (we call it standard hierarchical active learner) with the baseline learner. The standard hierarchical active learner uses intuitive strategies to handle oracle answers (see Section 3.2) in deep categories. If the oracle answer is "Yes", the standard hierarchical active learner directly includes the example as a positive example; if "No", it simply discards the example. On the other hand, the baseline learner is actually the non-active version of the standard hierarchical active learner. Instead of selecting the most informative examples, it selects unlabeled examples randomly on each category.

Empirical Comparison: We set the query limit as $50 \times |C|$ where $|C|$ is the total umber of categories in the hierarchy. Thus, in our experiments the query limits for the four datasets are 1,350, 4,300, 4,800 and 4,850 respectively. We denote the standard hierarchical active learner as AC and the baseline learner as RD. Figure 2 plots the average learning curves for AC and RD on the four datasets. As we can see, on all the datasets AC performs significantly better than RD. This result is reasonable since the unlabeled examples selected by AC are more informative than RD on all the categories in the hierarchy. From the curves, it is apparent that to achieve the best performance of RD, AC needs significantly fewer queries (approximately 43% to 82% queries can be saved)[10].

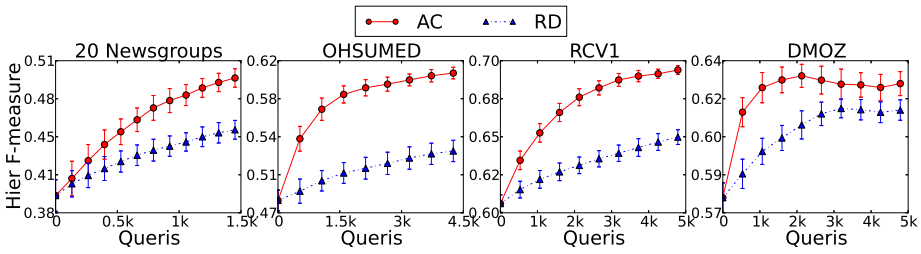

Fig. 2. Comparison between AC and RD in terms of the hierarchical F-measure. X axis is the number of queries consumed and Y axis is the hierarchical F-measure.

Although the standard hierarchical active learner (AC) significantly reduces the number of oracle queries compared to the baseline learner (RD), we should note that there is no interaction between categories in the hierarchy (e.g., each category independently selects examples and queries oracle). Our question is: can we further improve the performance of the standard active learner by taking

[10] In *20 Newsgroups*, RD uses 1,350 queries to achieve 0.46 in terms of the hierarchical F-measure, while AC only uses 750 queries. Thus, $(1350-750)/1350 = 44.4\%$ of the total queries are saved. The savings for other datasets are 82.5%, 72.9% and 43.3%.

into account the hierarchical relation of different categories? We will explore several leveraging strategies in the following subsections.

5.2 Leveraging Positive Examples in Hierarchy

As mentioned in Section 3.2, when the oracle on a category answers "Yes" for an example, we can directly include the example into the training set on that category as a positive example. Furthermore, according to the category relation in a hierarchy, if an example belongs to a category, it will definitely belong to all the ancestor categories. Thus, we can propagate the example (as a positive example) to all its ancestor categories. In such cases, the ancestor classifiers can obtain *free* positive examples for training without any query. It coincides with the goal of active learning: reducing the human labeling cost!

Based on the intuition, we propose a new strategy *Propagate* to propagate the examples to the ancestor classifiers when the answer from oracle is "Yes". The basic idea is as follows. In each iteration of the active learning process, after we query an oracle for each selected example, if the answer from the oracle is "Yes", we propagate this example to the training sets of all the ancestor categories as positive. At the end of the iteration, each category combines all the propagated positive examples and the examples selected by itself to update its classifier.

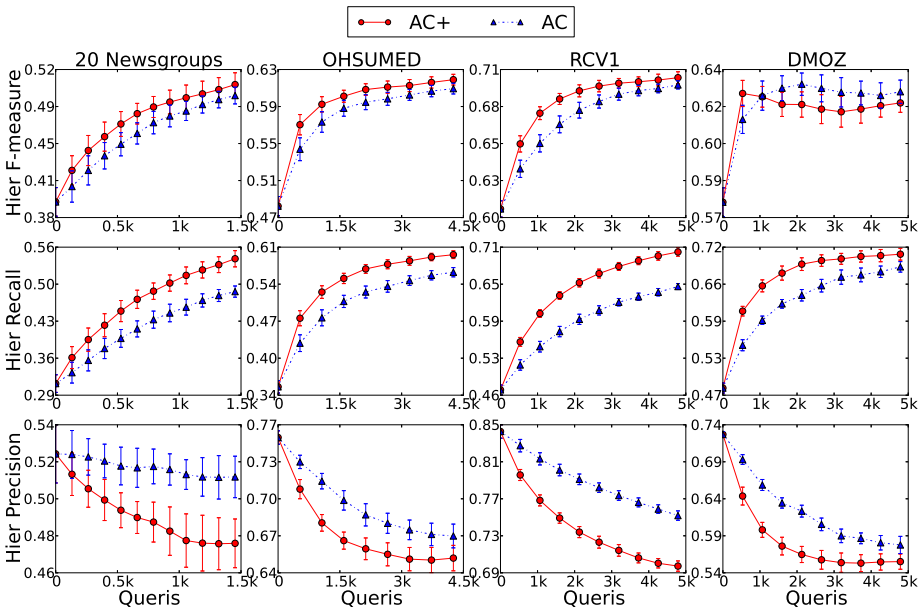

Fig. 3. Comparison between *AC+* and *AC* in terms of the hierarchical F-measure (first row), recall (second row) and precision (third row).

Empirical Comparison: We integrate *Propagate* to the standard hierarchical active learner (we name the integrated version as $AC+$) and then compare it with the original AC. The first row of Figure 3 shows the learning curves of $AC+$ and AC on the four datasets in terms of the hierarchical F-measure. Overall, the performance of $AC+$ is slightly better than that of AC. By propagating positive examples, the top-level classifiers of $AC+$ can receive a large number of positive examples and thus the (hierarchical) recall of $AC+$ increases faster than AC as shown in the second row. This is the reason why $AC+$ can defeat AC on the first three datasets. However, from the third row, we can see the hierarchical precision of $AC+$ actually degrades very sharply since the class distribution of the training set has been altered by the propagated positive examples. It thus weakens the boosting effect in the hierarchical recall and hinders the improvement of overall performance in the hierarchical F-measure.

Since positive examples can benefit the hierarchical recall, can we leverage negative examples to help maintain the hierarchical precision so as to further improve $AC+$? We will propose two possible solutions in the following.

5.3 Leveraging Negative Examples in Hierarchy

We introduce two strategies to leverage negative examples. One is to query parent oracles when the oracle answers "No"; the other is to predict the negative labels for sibling categories when the oracle answers "Yes".

Querying Negative Examples: For deep categories, when the oracle answers "No", we actually discard the selected example in $AC+$ (as well as in AC, see Section 5.1). However, in this case, the training set may miss a negative example and also possibly an informative example. Furthermore, if we keep throwing away those examples whenever oracle says "No", the classifiers may not have chance to learn negative examples. On the other hand, if we include this example, we may introduce noise to the training set, since the example may not belong to the parent category, thus an out-of-scope example (see Section 3.1).

How can we deal with the two cases? We introduce a complementary strategy called *Query*. In fact, the parent oracle can help us decide between the two cases. We only need to issue another query to the parent oracle on whether this example belongs to it. If the answer from the parent oracle is "Yes", we can safely include this example as a negative example to the current category. If the answer is "No", we can directly discard it. Here, we do not need to further query all the ancestor oracles, since the example is already out of scope of the current category and thus can not be included into its training set. There is a trade-off. As one more query is asked, we may obtain an informative negative example, but we may also waste a query. Therefore, it is non-trivial if this strategy works or not.

Predicting Negative Labels: When the oracle on a category (say "Astronomy") answers "Yes" for an example, it is very likely that this example may not belong to its sibling categories such as "Chemistry" and "Social Science". In this

case, can we add this example as a negative example to its sibling categories? In those datasets where each example only belongs to one single category path, we can safely do so. It is because for the categories under the same parent, the example can only belong to at most one category. However, in most of the hierarchical datasets, the example belongs to multiple paths. In this case, it may be positive on some sibling categories. If we include this example as negative to the sibling categories, we may introduce noise.

To decide which sibling categories an example can be included as negative, we adopt a conservative heuristic strategy called *Predict*. Basically, when a positive example is included into a category, we add this example as negative to those sibling categories that the example is least likely to belong to. Specifically, if we know a queried example x is positive on a category c, we choose m sibling categories with the minimum probabilities (estimated by Platts Calibration [11]). We set

$$m = n - \max_{x \in D_L} \Psi_{\uparrow c}(x), \tag{2}$$

where D_L is the labeled set, $\uparrow c$ is the parent category of c, n is the number of children categories of $\uparrow c$, $\Psi_{\uparrow c}(x)$ is the number of categories under $\uparrow c$ that the example x belongs to.

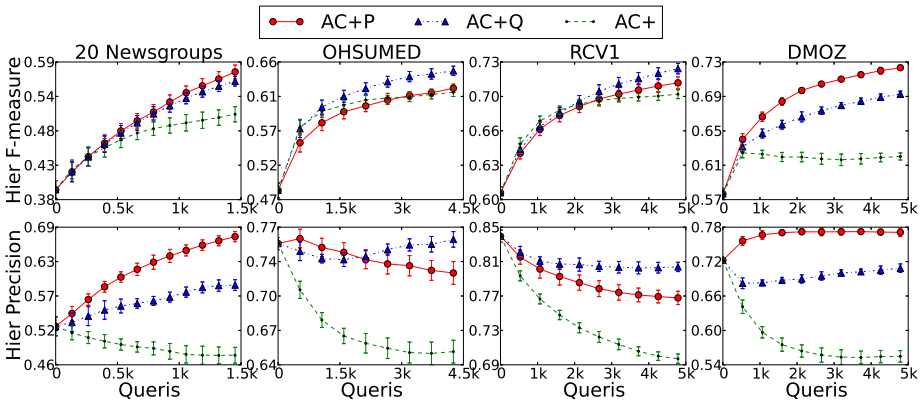

Fig. 4. Comparison between *AC+P*, *AC+Q* and *AC+* in terms of the hierarchical F-measure (upper row) and precision (bottom row).

Empirical Comparison: We integrate the two strategies *Query* and *Predict* discussed above into *AC+* and then compare the two integrated versions (*AC+Q* and *AC+P*) with the original *AC+*. Since in *AC+* positive examples are propagated, we can use this feature to further boost *AC+Q* and *AC+P*. For *AC+Q*, when the parent oracle answers "Yes", besides obtaining a negative example, we

can also propagate this example as a positive example to all the ancestor categories. For *AC+P*, as a positive example is propagated, we can actually apply *Predict* to all the ancestor categories.

We plot their learning curves for the hierarchical F-measure and the hierarchical precision on the four datasets in Figure 4. As we can see in the figure, both *AC+Q* and *AC+P* achieve better performance of the hierarchical F-measure than *AC+*. By introducing more negative examples, both methods maintain or even increase the hierarchical precision (see the bottom row of Figure 4). As we mentioned before, *AC+Q* may waste queries when the parent oracle answers "No". However, we discover that the average number of informative examples obtained per query for *AC+Q* is much larger than *AC+* (at least 0.2 higher per query). It means that it is actually worthwhile to issue another query in *AC+Q*. Another question is whether *AC+P* introduces noise to the training sets. According to our calculation, the noise rate is at most 5% on all the four datasets. Hence, it is reasonable that *AC+Q* and *AC+P* can further improve *AC+*.

However, between *AC+Q* and *AC+P*, there is no consistent winner on all the four datasets. On 20 Newsgroup and DMOZ, *AC+P* achieves higher performance, while on OHSUMED and RCV1, *AC+Q* is more promising. We also try to make a simple combination of *Query* and *Predict* with *AC+* (we call it *AC+QP*), but the performance is not significantly better than *AC+Q* and *AC+P*. We will explore a smarter way to combine them in our future work.

Finally, we compare the improved versions *AC+Q* and *AC+P* with the non-active version *RD*. We find that *AC+Q* and *AC+P* can save approximately 74% to 90% of the total queries. The savings for the four datasets are 74.1%, 88.4%, 83.3% and 90% respectively (these numbers are derived from Figures 2 and 4).

To summarize, we propose several improved versions (*AC+*, *AC+Q* and *AC+P*) in addition to the standard version (*AC*) of our hierarchical active learning framework. According to our empirical studies, we discover that in terms of the hierarchical F-measure, *AC+Q* and *AC+P* are significantly better than *AC+*, which in turn is slightly better than *AC*, which in turn outperforms *RD* significantly. In terms of query savings, our best versions *AC+Q* and *AC+P* need significantly fewer queries than the baseline learner *RD*.

6 Conclusion

We propose a new multi-oracle setting for active learning in hierarchical text classification as well as an effective active learning framework for this setting. We explore different solutions which attempt to utilize the hierarchical relation between categories to improve active learning. We also discover that propagating positive examples to the ancestor categories can improve the overall performance of hierarchical active learning. However, it also decreases the precision. To handle this problem, we propose two additional strategies to leverage negative examples in the hierarchy. Our empirical study shows both of them can further boost the performance. Our best strategy proposed can save a considerable number of queries (74% to 90%) compared to the baseline learner. In our future work,

we will extend our hierarchical active learning algorithms with more advanced strategies to reduce queries further.

References

1. Ceci, M., Malerba, D.: Classifying web documents in a hierarchy of categories: a comprehensive study. J. Intell. Inf. Syst. 28, 37–78 (2007)
2. DAlessio, S., Murray, K., Schiaffino, R., Kershenbaum, A.: The effect of using hierarchical classifiers in text categorization. In: RIAO '00. pp. 302–313 (2000)
3. Daraselia, N., Yuryev, A., Egorov, S., Mazo, I., Ispolatov, I.: Automatic extraction of gene ontology annotation and its correlation with clusters in protein networks. BMC Bioinformatics 8(1), 243 (Jul 2007)
4. Donmez, P., Carbonell, J.G.: Proactive learning: cost-sensitive active learning with multiple imperfect oracles. In: CIKM '08. pp. 619–628 (2008)
5. Esuli, A., Sebastiani, F.: Active learning strategies for multi-label text classification. In: ECIR '09. pp. 102–113 (2009)
6. Fagni, T., Sebastiani, F.: Selecting negative examples for hierarchical text classification: An experimental comparison. J. Am. Soc. Inf. Sci. Technol. 61, 2256–2265 (2010)
7. 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)
8. Lam, W., Ho, C.Y.: Using a generalized instance set for automatic text categorization. In: SIGIR '98. pp. 81–89 (1998)
9. 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)
10. Nowak, S., Rüger, S.: How reliable are annotations via crowdsourcing: a study about inter-annotator agreement for multi-label image annotation. In: MIR '10. pp. 557–566 (2010)
11. Platt, J.C.: Probabilistic outputs for support vector machines. Advances in Large Margin Classifiers pp. 61–74 (1999)
12. Roy, N., McCallum, A.: Toward optimal active learning through sampling estimation of error reduction. In: ICML '01. pp. 441–448 (2001)
13. Ruiz, M.E., Srinivasan, P.: Hierarchical neural networks for text categorization (poster abstract). In: SIGIR '99. pp. 281–282 (1999)
14. Silla, Jr., C.N., Freitas, A.A.: A survey of hierarchical classification across different application domains. Data Min. Knowl. Discov. 22, 31–72 (2011)
15. Sun, A., Lim, E.P.: Hierarchical text classification and evaluation. In: ICDM '01. pp. 521–528 (2001)
16. Tong, S., Koller, D.: Support vector machine active learning with applications to text classification. J. Mach. Learn. Res. 2, 45–66 (2002)
17. Verspoor, K., Cohn, J., Mniszewski, S., Joslyn, C.: Categorization approach to automated ontological function annotation. In: Protein Science. pp. 1544–1549 (2006)
18. Xu, Z., Yu, K., Tresp, V., Xu, X., Wang, J.: Representative sampling for text classification using support vector machines. In: ECIR '03. pp. 393–407 (2003)
19. Xue, G.R., Xing, D., Yang, Q., Yu, Y.: Deep classification in large-scale text hierarchies. In: SIGIR '08. pp. 619–626 (2008)
20. Yang, B., Sun, J.T., Wang, T., Chen, Z.: Effective multi-label active learning for text classification. In: KDD '09. pp. 917–926 (2009)