A k-NN Method for Large Scale Hierarchical Text Classification at LSHTC3

Xiaogang Han[1], Shaohua Li[1], and Zhiqi Shen[1]

School of Computer Engineering, Nanyang Technological University,
Nanyang Ave, Singapore 639798
hanx0009@e.ntu.edu.sg
shaohua@gmail.com
zqshen@ntu.edu.sg

Abstract. In this paper, we propose a classification method for the LSHTC3 *Large Scale Hierarchical Classification* track and the *Multi-task Learning* track. Our method integrates N-grams into the computation of k nearest neighbors, and the category hierarchy into the candidate category ranking. Firstly, we enhance the Bag-of-Words model with N-grams to represent documents. Secondly, two k-NN algorithms, using $TF \cdot IDF$-weighted cosine similarity and $BM25$ respectively, are performed to select a small number of candidate categories. Furthermore, several features of the candidate categories are extracted. A log-transformed linear model aggregates these feature values with different weights, which are tuned through cross-validation. The model outputs a ranking score for each candidate category, and finally the top few candidate categories are chosen as the predicted categories. The experiments performed on both tracks show that our method can gain high accuracy.

Keywords: Hierarchical Text Classification, k-NN, Candidate Category Ranking

1 Introduction

Hierarchies are becoming more and more important for the management of web documents. There is a need for automated classification of new documents into the categories in a given hierarchy to help search, recommend, and filter information for the user. Wikipedia[1] and ODP[2] provide hierarchical categorization information for a wide range of topics. Efficient and accurate classification of new documents into the categories in such web hierarchies is a challenging task to the text mining community.

In this paper, we study the problem of large scale hierarchical text classification based on Wikipedia and ODP data. Our approach is a multiple stage classification model which is motivated by the intuition that similar documents

[1] http://www.wikipedia.org
[2] http://www.dmoz.org/

often belong in the same categories, and the most likely categories a query document belongs in are usually among the categories of the documents most similar to the query document. Based on this intuition, a k nearest neighbor (k-NN) classification algorithm along with various enhancement is proposed.

Firstly, we use the N-gram extension of the Bag-of-Words (BoW) model to represent a document. Secondly, each of two k-NN algorithms, using $TF \cdot IDF$-weighted cosine similarity and $BM25$ respectively, is performed to select the most similar k documents to a query document. We collect the categories of each set of k documents and get two sets of categories. These categories are referred to as *candidate categories* of the query document. The two sets of candidate categories are then ranked respectively, and combined afterwards. Since only a small number of categories become candidates, the prediction among them becomes very efficient. In order to find the most plausible categories of the query document, we assign a ranking score to each candidate category using a log-transformed linear model of four predefined features. The weights of these features are estimated through cross-validation on a training set. Finally, the candidate categories with the largest ranking scores are chosen as the predicted categories. The competition results show that our approach is one of the best performers. Moreover, our approach is general and can potentially be applied to various hierarchical text categorization tasks beyond these two web hierarchies.

The rest of the paper is structured as follows. Section 2 recaps the basic concepts and notations in our method. Section 3 presents the k-NN based candidate category retrieval, the features of the candidate categories, and the candidate category ranking model. Section 4 presents the evaluation results. Section 5 reviews related work. Section 6 concludes the paper with possible future work.

2 Hierarchies and Classification

In this section, we define the terms and notions that will be used in the description of the classification algorithm. A hierarchy H of categories is a collection of superior categories (*superiors* or *parents*), each of which subsumes a collection of subordinate categories (*subordinates* or *children*). Each subordinate could have its own subordinates, until the most specific categories (*leaf categories*) are reached. Legal categories for the classification task are only those leaf categories in the hierarchy which do not have any subordinates. In the classification task, each leaf category has a collection of documents in the training set, which are compiled by human experts. The structure of the hierarchy is shown in Figure 1.

We denote the set of m leaf categories as $C = \{c_1, c_2, ..., c_m\}$. C is the category space for the classification task. The set of n training documents is denoted as $D = \{d_1, d_2, ..., d_n\}$, and the set of u terms (i.e. N-grams with $N \leq 3$) appearing in D as $T = \{t_1, t_2, ..., t_u\}$. In our current approach, only the immediate superiors of the leaf categories are used in the category prediction. We refer to these categories as the parent category space, denoted by $P = \{p_1, p_2, ..., p_s\}$.

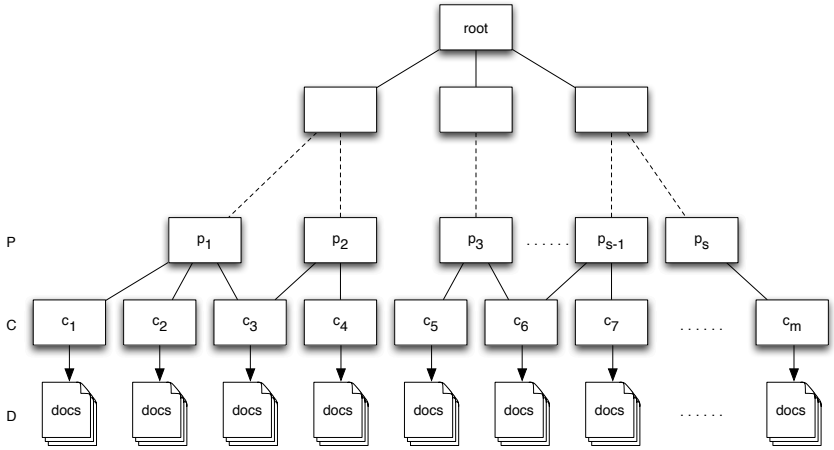

Fig. 1. Hierarchical Categorization Structure.

Based on these definitions, we denote the set of parent nodes of category c by $Parents(c)$, as one category can be subsumed by more than one parent category. $Children(p)$ denotes the set of children categories of category p. For instance, in Figure 1, $Parents(c_1) = \{p_1, p_2\}$, and $Children(p_1) = \{c_1, c_2, c_3\}$. The set of documents in categories c is denoted by $D(c)$, the set of categories of document d is denoted by $Cat(d)$, and the set of terms in document d is denoted by $T(d)$.

One natural property of the category hierarchy is, if a document d belongs in a category c, it also belongs in each of c's parent categories, i.e., $\forall p \in Parents(c), d \in D(p)$. The classification problem can be stated as follows: given query document d, find the leaf categories in C that d belongs in.

3 A *k*-NN Based Hierarchical Classification Algorithm

3.1 Overview

In text classification, k-NN method classifies documents based on the votes of its most similar training documents (k nearest neighbors). We use k-NN to retrieve the most similar k documents for each query document. Rather than ranking the candidate categories by votes (weighted or not) and predict the categories of the document as the most voted categories, we use a log-transformed linear model to rank the categories more reasonably based on four critical features. The weights of these features are tuned through cross-validation. The top ranked categories are chosen as the predicted categories. The overall algorithm is shown in Fig. 2.

3.2 Document Representation

The first step is to prepare the training data and get the tuple set $DS = \{(d_i, \vec{d_i}, c_j) | 1 \leq i \leq n, c_j \in Cat(d_i)\}$ in which $\vec{d_i}$ is the term vector represen-

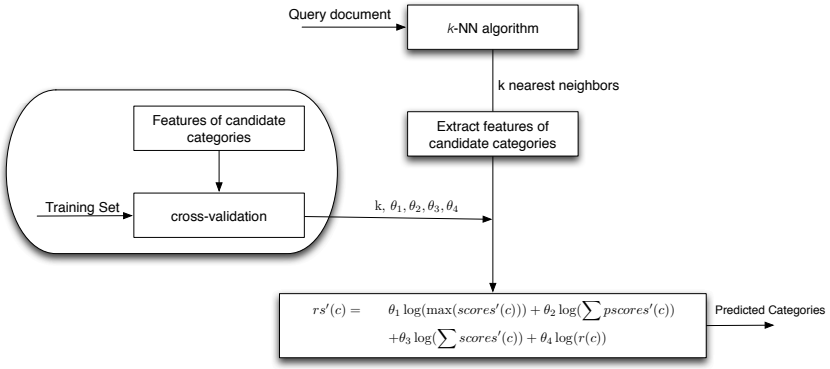

Fig. 2. Algorithm Workflow

tation of document d_i and c_j is one of d_i's category. Note if d_i belongs in multiple categories, i.e. $Cat(d_i) = \{c_{j_1}, c_{j_2}, \cdots\}$, then each category $c_{j_k} \in Cat(d_i)$ corresponds to a tuple $(d_i, \vec{d_i}, c_{j_k})$. In the BoW representation, in addition to unigrams, we also extract N-grams (N is up to 3) to expand the feature space[3].

3.3 Similarity Measures

We introduce both $TF \cdot IDF$-based cosine similarity (referred to as *tfidf*) and $BM25$ as measures to calculate the similarity between a query document and each document in the training dataset.

tfidf We use a variant of $TF \cdot IDF$ model ([12]) to represent the weights of each term in a document, as this variant can improve the accuracy significantly, compared to standard $TF \cdot IDF$ model. The $TF \cdot IDF$ variant is defined as:

$$\omega_{t,d} = \log(tf(t,d) + 1) \cdot idf(t) \tag{1}$$

where $tf(t,d)$ is the frequency count of term t in document d, n is the number of documents in the training corpus, and

$$idf(t) = \log(\frac{n}{n_t}) \tag{2}$$

is the *inverse document frequency* of term t, in which n_t is the number of documents containing t.

The vector space model for calculating the similarity between the query document d and a training document d_i can be expressed as:

$$cossim(d_i, d) = \frac{d_i \cdot d}{\|d_i\| \cdot \|d\|} = \frac{\sum_{k=1}^{u} \omega_{k,i}\omega_k}{\sqrt{\sum_{k=1}^{u} \omega_{k,i}^2}\sqrt{\sum_{k=1}^{u} \omega_k^2}} \tag{3}$$

[3] As the original text is only available for the medium-size Wikipedia dataset, we can only apply N-grams to the medium-size Wikipedia task of the *Large Scale Hierarchical Classification* track

BM25 In BM25 [8] weighting scheme, suppose the query document d contains terms $\vec{t} = \{t_1, ..., t_m\}$. The BM25 score of a training document d_i is:

$$bm25sim(d, d_i) = \sum_{j=1}^{m} idf(t_j) \cdot \frac{tf(t_j, d_i) \cdot (k_1 + 1)}{tf(t_j, d_i) + k_1 \cdot (1 - b + b \cdot \frac{|d_i|}{avgdl_D})} \cdot \frac{tf(t_j, d) \cdot (k_1 + 1)}{tf(t_j, d) + k_1 \cdot (1 - b + b \cdot \frac{|d|}{avgdl_Q})} \tag{4}$$

where $tf(t_j, d)$ and $tf(t_j, d_i)$ are t_j's term frequency in document d and d_i, respectively. $|d|$ and $|d_i|$ are the length (count of terms) of document d and d_i, and $avgdl_D$ and $avgdl_Q$ are the average document length in the training data set D and test data set Q, respectively. k_1 and b are free parameters. In our experiment, we set $k_1 = 1.9$ and $b = 0.9$. $idf(t_j)$ is the inverse document frequency of the term t_j in D, and computed as:

$$idf(t_j) = log(\frac{n - n_t + 0.5}{n_t + 0.5}) \tag{5}$$

where n and n_t are defined the same as in the $TF \cdot IDF$ model. Please note the two $idf(t_j)$ definitions in 2 and 5 are different by convention.

3.4 *k*-NN Algorithm

Suppose a query document d in the test set is given. **Algorithm** 1 depicts our method to calculate its k nearest neighbors in D and the corresponding candidate category tuples. $sim(d_i, d)$ in **Algorithm** 1 could be $tfidf$ or $BM25$.

Algorithm 1 Calculating k-Nearest Neighbors and Candidate Category Tuples

Require: query document d
1: **for** $d_i \in D$ **do**
2: $score_i = sim(d_i, d)$
3: **end for**
4: Sort documents in D by $score_i$ in descending order. Denote the top k documents as KNN(d).
5: Retrieve tuples from the tuple set DS for each document in KNN(d), and get $CD = \{(d_i, \vec{d_i}, c_j) | d_i \in \text{KNN}(d)\}$.
6: $CS = \phi$.
7: **for** $(d_i, \vec{d_i}, c_j) \in CD$ **do**
8: $CS = CS \cup (d_i, c_j, score_i)$
9: **end for**
10: **return** CS

3.5 Combination of Two Similarity Measures

Since the k nearest neighbors selected using different similarity measures capture different aspects of the semantics and may complement with each other, we combine the two sets of nearest neighbors, along with their categories, to form the input of our prediction model. The details are trivial and omitted here.

3.6 Candidate Category Feature Extraction and Ranking

A known characteristic of *voting* for k-NN classification is that since documents are not uniformly distributed in all categories, the categories with more documents tend to come up more often in the k nearest neighbors, leading to a bias favoring frequent categories. But in LSHTC3, for each category, one test document is sampled from all documents in this category, and therefore such a bias is not desirable. Moreover, for the hierarchical classification, the hierarchical relationships between different categories expose rich information helpful for the categorization of a document, but a simple voting scheme leaves the hierarchical information unutilized. Therefore we use a ranking scheme incorporating both the distributions of the candidate categories and the category hierarchy to improve the classification accuracy.

In this subsection, we will explore the features of the candidate categories, and explain how to assign reasonable weights to the features using cross-validation. Note for a query document d, we will use both $tfidf$ and $BM25$ as similarity measures, get two different sets of candidate category tuples. The following discussion applies to the candidate category tuples under either similarity measure.

Initial Scheme of Ranking Score Evaluation. Let $CS = \{(d_i, c_j, score_i)\}$ denote the set of tuples representing the k nearest neighbors, their categories and similarity scores with the query document d. Let $CC = \{c_j | (d_i, c_j, score_i) \in CS\}$ denote the set of categories in CS. Based on the analysis on CS, we have found the following quantities are important in our classification task:

1. For each category c, the set of similarity scores of the documents belonging in c in CS, denoted by $scores(c) = \{score_i | (d_i, c_j = c, score_i) \in CS\}$:

   As each category may correspond to multiple documents in the k nearest neighbors (with different similarity scores), it is important to aggregate these scores for better classification. We mainly use two ways to aggregate these scores: $\max(scores(c))$ and $\sum scores(c)$, which denote the maximum and the sum of the scores in $scores(c)$, respectively.

   $|scores(c)|$ denotes the count of scores in $scores(c)$, which is also the number of documents belonging in c;

   The definition of $scores(c)$ naturally extends to a set of categories S: $scores(S) = \bigcup_{c \in S} scores(c)$.

2. The count of training documents in each category c, denoted by $|D(c)|$:

   It is observed that the total number of training documents in a category c positively affects the count of documents in c in CS.

3. The similarity scores of the nearest neighbors in the parents of a category c, denoted by $pscores(c)$:

   In order to utilize the hierarchical structure of the categories, we incorporate into our model the documents belonging in the parents of c. The intuition is, if more nearest neighbors of the query document d belong in $Parents(c)$, the chance that d belongs in c is also larger. The similarity scores of those document to d will positively affect category c's ranking. These scores are formally defined as the scores of a subset of CS:

$$\begin{aligned} pscores(c) &= scores(\, Children(\, Parents(c)\,)\,) \\ &= \{score_i | (d_i, c_j, score_i) \in CS, c_j \in Children(Parents(c))\}. \end{aligned}$$

   $c \in Children(Parents(c))$, and therefore $scores(c) \subseteq pscores(c)$. Currently we only utilize the immediate parents of the leaf categories in the hierarchy, as the higher is the category in the hierarchy level, the less specific is it, and the less informative for the classification.

From the above quantities, we derive four features which are the input of our prediction model:

1. $\max(scores(c))$, the strongest evidence supporting category c;
2. $\sum pscores(c)$, aggregating all the evidence supporting the parents of category c, thus supporting c indirectly;
3. $\sum scores(c)$, aggregating all the evidence supporting category c;
4. The ratio $r(c) = \frac{|scores(c)|}{|D(c)|}$, intended to reduce the effect of $|D(c)|$ on $|scores(c)|$.

In addition, our model has a few parameters which need to be tuned to get the optimal classification accuracy:

1. The choice of k, i.e. the size of the nearest neighbors;
2. The weights of the four features, denoted by $\theta_1, \theta_2, \theta_3$ and θ_4. These weights scale the contributions of the features differently, matching with their different correlations with the real categories.

We use a simple log-transformed linear model to calculate the scores of the candidate categories. The candidate categories are then ranked by the scores, and the top few categories are chosen as the predicted categories of the query document. The ranking score of a category c is denoted by $rs(c)$, which is defined as follows:

$$\begin{aligned} rs(c) = \; & \theta_1 \log(\max(scores(c))) + \theta_2 \log(\sum pscores(c)) \\ & + \theta_3 \log(\sum scores(c)) + \theta_4 \log(r(c)) \end{aligned} \tag{6}$$

Optimized Scheme of Ranking Score Evaluation. We tried the following transformation on the quantities $scores(c)$, and a new function in place of $pscores(c)$ to incorporate the parent-children relationships between categories.

The new schemes below are verified to slightly improve the classification performance:

$$scores'(c) = \{\, log(1 + score) \mid score \in scores(c) \,\},$$
$$pscores'(c) = \{\, log(|Children(p)|) \,\big|\, p \in Parents(c) \,\}.$$

$pscores'(c)$ uses the logarithm of the number of children of each parent of c to represent these parents. Parents with more children will have higher scores, and thus their child categories are more favored in the prediction. But if two categories are both the children of a parent, then the scores with regard to this parent will be the same, and other feature values will determine their precedence. The new ranking score is as follows:

$$\begin{aligned} rs'(c) = \; & \theta_1 \log(\max(scores'(c))) + \theta_2 \log(\sum pscores'(c)) \\ & + \theta_3 \log(\sum scores'(c)) + \theta_4 \log(r(c)) \end{aligned} \tag{7}$$

Due to space limitations, our following discussions, especially the reported optimal parameter values and experimental results, are those of the optimized ranking score scheme.

Parameter Tuning by Cross-validation. To tune the parameters, cross-validation is performed on the medium-size Wikipedia training set. Other tasks use the same set of parameters.

It is observed that the test data was sampled evenly from each category, and thus we perform the same sampling procedure on the training data. We randomly selected one document in each category in the training data, into the validation set. The rest documents form the sub-training set. Eventually, we get a sub-training set of 420,386 documents and a validation set of 36,500 documents.

The documents in the sub-training set along with the corresponding category labels are used to train the model. The validation algorithm then tries different combinations of the parameters on the validation set, and chooses the one that maximizes the prediction accuracy as the optimal parameter values. Each individual parameters is tried with different values in a pre-specified range. The step size of the values is fixed to 0.1 for all parameters. The ranges and the optimal parameter values are listed in Table 1.

Table 1. Parameter Tuning Range and Results

Parameter	Range	Optimal
θ_1	1∼5	3.4
θ_2	0∼1	0.6
θ_3	0∼1	0.8
θ_4	0∼1	0.2

3.7 Multiple Category Classification

As each document can be assigned to multiple categories in the hierarchy, we select top-M categories as the predicted categories of a query document d. Note M varies across documents, so one problem is how to decide M for each document.

Let $avgCats$ denote the average number of leaf categories per document within the hierarchy, which is pre-computed from the validation set. For the ranked list of categories $(rs'(c_{i_1}), rs'(c_{i_2}), \cdots, rs'(c_{i_k}), \cdots)$ computed by the candidate category ranking algorithm, we choose all categories whose ranking scores are large enough relative to the largest score $rs'(c_{i_1})$ as d's predicted categories, i.e. $rs'(c_{i_k})/rs'(c_{i_1}) > \alpha$, where $0 < \alpha < 1$ is a constant ratio threshold. In order to tune α, we calculate the predicted average number of categories per document in the validation set, denoted as $avgPredCats(\alpha)$. By iteratively trying different values of α and calculating the $error = |avgPredCats(\alpha) - avgCats|$, the α value with the minimum $error$ is chosen as the ratio threshold.

3.8 Multi-task Learning

Two different but closely-related category systems share a lot of common elements, and the information shared between them can be utilized to improve classification performance on each of the individual tasks. Multi-task learning in the context of text classification aims at learning two classification models at the same time by utilizing the commonality between the two tasks and category systems. However, in our experiments, we ignore the commonality of the two tasks, and treat the them as two independent tasks and use the same trained model to do prediction separately.

4 Evaluation and Results

4.1 Settings

ECML/PKDD 2012 Discovery Challenge[4] consists of three tracks based on two large datasets: one created from the ODP web directory (DMOZ) and one from Wikipedia. The datasets are multi-class, multi-label and hierarchical. The number of categories range between 13,000 and 325,000 roughly and the number of the documents between 380,000 and 2,400,000. The three tracks are:

1. Track1 - Standard large-scale hierarchical classification
   - Task1 - Medium Wikipedia
   - Task2 - Large Wikipedia
2. Track2 - Multi-task learning
   - Task1 - DMOZ
   - Task2 - Wikipedia
3. Track3 - Refinement-learning

The metrics used for evaluating the classification algorithms include accuracy, example-based F-measure, label-based macro F-measure, label-based micro F-measure and multi-label graph-induced error [10].

[4] `http://www.ecmlpkdd2012.net/info/discovery-challenge/`

4.2 Track1 Experimental Results

The experimental results for the 2 tasks in Track1 compared to k-NN baseline algorithm with regards to various measures are shown in Table 2.

Table 2. Evaluation results for Track 1

Task	1		2	
Algorithm	Our algorithm	k-NN baseline	Our algorithm	k-NN baseline
Rank	4	14	2	6
Accuracy	0.412	0.249	0.346	0.272
Example Based F1-measure	0.477	0.318	0.42	0.347
Example Based Precision	0.518	0.283	0.607	0.363
Example Based Recall	0.512	0.416	0.367	0.387
Label Based Macro F1-measure	0.245	0.176	0.17	0.149
Label Based Macro Precision	0.426	0.252	0.538	0.303
Label Based Macro Recall	0.3	0.235	0.176	0.177
Label Based Micro F1-measure	0.419	0.298	0.345	0.302
Label Based Micro Precision	0.394	0.251	0.551	0.326
Label Based Micro Recall	0.447	0.367	0.251	0.281
Hierachical F1-measure	0.677	0.561	0.546	0.519
Hierachical Precision	0.709	0.512	0.743	0.542
Hierachical Recall	0.717	0.691	0.5	0.576

It can be observed that the proposed algorithm can obtain promising accuracy. In particular, the accuracy for Medium Wikipedia data set (task1) is relatively higher than Large Wikipedia data set (task2), which may attribute to the fact that the more the number of categories and number of documents, the harder to distinguish the feature space of each category. Another trend can be observed from the table is that k-NN baseline performs better for Large Wikipedia data sets than Medium Wikipedia data set, which probably due to the fact that the larger the training corpus is, the more likely very similar documents to the query document exist in the corpus, and therefore the more accurate the categories of the nearest neighbors are.

4.3 Results for multi-task learning

The results of our algorithm for the multi-task learning track (Track2) are shown in Table 3. It shows that our algorithm produced high accuracy for the DMOZ data compared with k-NN baseline. However, the performance for the Wikipedia data is relatively lower which might due to the noise in the Wikipedia data set.

5 Related Work

Example based classification, especially k-NN algorithm is very popular for text classification [11][4]. However, how to identify important features in the training data to make better prediction is always a concern when applying k-NN to

Table 3. Evaluation results for Track 2

Task	1		2	
Algorithm	Our algorithm	k-NN baseline	Our algorithm	k-NN baseline*
Rank	2	4	2	-
Accuracy	0.5198	0.3345	0.0268	-
F1-measure	0.3136	0.0834	0.004	-
Precision	0.3093	0.0928	0.0033	-
Recall	0.3051	0.1044	0.0029	-
Hierarchical F1-measure	0.7335	0.6148	0.1505	-
Hierarchical Precision	0.7348	0.6144	0.1598	-
Hierarchical Recall	0.734	0.6177	0.1851	-

*The k-NN baseline data for Task 2 was not provided by the competition.

practical data, especially hierarchical classification problems. Selecting the features based on the data [6][2] is very important for robust learning model. The hierarchical relationships [1][9] between different categories in the hierarchical category space expose extra information about the categorization of new document [5]. However, how to utilize the semantic relationships between hierarchical categories is still an open problem.

In participating the *Second Pascal Challenge on Large Scale Hierarchical Text classification*[5], we developed an optimized k-NN algorithm [3], which obtained high accuracy compared with other algorithms.

One of the most important problem for k-NN algorithm is to compute the similarity between two documents. $TF \cdot IDF$ model [7] and $BM25$ [8] are the most widely used weighting schemes. Dependent on the specific problems, various variant of $TF \cdot IDF$ model are developed. We used the one described in [12] in our algorithm.

6 Conclusion and Future Work

6.1 Conclusion

In this paper, we proposed a multiple stages hierarchical text classification method based on k-NN algorithm. Firstly, the k-NN algorithm was performed to select a few most similar documents, whose categories become candidate categories for the classification. Secondly, several important candidate category features were extracted. Finally, the categories prediction algorithm uses a log-transformed linear model to assign scores to the candidate categories, and the top ranked categories are chosen as the predicted categories of the query document. The weights of those features in the log-transformed linear model are estimated through cross-validation.

[5] `http://lshtc.iit.demokritos.gr/LSHC2_workshop`

We found that when calculating the similarity between two document term vectors, a variant of the $TF \cdot IDF$ ([12]) can perform better than the standard $TF \cdot IDF$ scheme. $BM25$ is combined with $TF \cdot IDF$ for better performance.

We also found that for hierarchical classification problem, the hierarchical relationships expose extra information. Incorporating such information as a candidate category feature can improve the classification performance significantly.

6.2 Future Work

There is still lots of work that can be done with the proposed classification model. In the current model, we only utilized two levels of hierarchical relationships. It is possible to identify the impact of higher level hierarchical relationships for the classification. Furthermore, a more general ensemble method can be developed to scale different features into the same space.

References

1. Dumais, S., Chen, H.: Hierarchical classification of web content. In: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval. pp. 256–263 (2000)
2. Forman, G.: An extensive empirical study of feature selection metrics for text classification. The Journal of Machine Learning Research 3, 1289–1305 (2003)
3. Han, X., Liu, J., Shen, Z., Miao, C.: An optimized k-nearest neighbor algorithm for large scale hierarchical text classification. In: Joint ECML/PKDD PASCAL Workshop on Large-Scale Hierarchical Classification. pp. 2–12 (2011)
4. Larkey, L., Croft, W.: Combining classifiers in text categorization. In: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 289–297 (1996)
5. Mladenic, D., Grobelnik, M.: Feature selection for classification based on text hierarchy. In: Proceedings of the Workshop on Learning from Text and the Web (1998)
6. Rogati, M., Yang, Y.: High-performing feature selection for text classification. In: Proceedings of the eleventh international conference on Information and knowledge management. pp. 659–661 (2002)
7. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval* 1. Information processing & management 24(5), 513–523 (1988)
8. Singhal, A., Buckley, C., Mitra, M.: Pivoted document length normalization. In: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 21–29. ACM (1996)
9. Sun, A., Lim, E.: Hierarchical text classification and evaluation. In: Proceedings IEEE International Conference on Data Mining. pp. 521–528 (2001)
10. Tsoumakas, G., Katakis, I., Vlahavas, I.: Random k-labelsets for multi-label classification. IEEE Transactions on Knowledge and Data Engineering (2010)
11. Yang, Y., Chute, C.: An example-based mapping method for text categorization and retrieval. ACM Transactions on Information Systems 12(3), 252–277 (1994)
12. Yang, Y., Liu, X.: A re-examination of text categorization methods. In: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval. pp. 42–49 (1999)