Proceedings of the Ninth International Conference on Machine Learning and Cybernetics, Qingdao, 11-14 July 2010

A COMPARATIVE STUDY ON TWO LARGE-SCALE HIERARCHICAL TEXT CLASSIFICATION TASKS' SOLUTIONS

JIAN ZHANG[1], HAI ZHAO[1], BAO-LIANG LU[1, 2]

[1]Center for Brain-Like Computing and Machine Intelligence，Department of Computer Science and Engineering, Shanghai Jiao Tong University, 800 Dong Chuan Road, Shanghai, China, 200240
[2]MOE-Microsoft Key Laboratory for Intelligent Computing and Intelligent Systems, Shanghai Jiao Tong University, 800 Dong Chuan Road, Shanghai, China, 200240
E-MAIL: jianzhang@sjtu.edu.cn, zhaohai@cs.sjtu.edu.cn, bllu@sjtu.edu.cn

Abstract:

Patent classification is a large scale hierarchical text classification (LSHTC) task. Though comprehensive comparisons, either learning algorithms or feature selection strategies, have been fully made in the text categorization field, few work was done for a LSHTC task due to high computational cost and complicated structural label characteristics. For the first time, this paper compares two popular learning frameworks, namely hierarchical support vector machine (SVM) and *k* nearest neighbor (*k*-NN) that are applied to a LSHTC task. Experiment results show that the latter outperforms the former in this LSHTC task, which is quite different from the usual results for normal text categorization tasks. Then this paper does a comparative study on different similarity measures and ranking approaches in *k*-NN framework for LSHTC task. Conclusions can be drawn that *k*-NN is more appropriate for the LSHTC task than hierarchical SVM and for a specific LSHTC task. BM25 outperforms other similarity measures and ListWeak gains a better performance than other ranking approaches. We also find an interesting phenomenon that using all the labels of the retrieved neighbors can remarkably improve classification performance over only using the first label of the retrieved neighbors.

Keywords:

Large-scale text classification; Hierarchical text classification; Text classification; comparative study; *k*-NN; Hierarchical SVM; Similarity measure; Ranking approach

1. Introduction

In recent years, along with the rapid development of the Internet, there is an increasing need for solutions to large-scale hierarchical text classification (LSHTC) tasks. By 'large scale', it includes two levels of meaning. One is that there are tens of thousands of categories in the class label taxonomy. The other is that there are millions of training samples. By 'hierarchical', it refers to the class label taxonomy's hierarchical structure. In traditional text classification tasks, there are only several or at most dozens of classes which are predefined in the class label taxonomy and there are usually tens of thousands or at most hundreds of thousands of training samples in the dataset. Thus a question emerges that whether the effective framework in traditional text classification tasks will still work in LSHTC tasks. It is known to us that SVM remains the best classification framework for traditional text classification task in which the scale of the class label taxonomy is not large, such as in [1] and it usually outperforms *k*-NN, which is a classical and effective framework in text classification. LSHTC tasks' characteristics such as 'large scale' and 'hierarchical' bring challenges to the aforementioned classification frameworks. Comparative study shows that *k*-NN framework outperforms SVM framework in the LSHTC task described in Section 4. And comparative study is conducted to investigate the selection of the value of *k*, decay factor r, similarity measure and ranking approach in *k*-NN framework for LSHTC task.

The rest of this paper is organized as follows. In Section 2 the *k*-NN framework is described. Section 3 gives a description of hierarchical SVM framework. Section 4 introduces two LSHTC tasks. In Section 5, comparisons between experiments results are conducted. Section 6 concludes this paper and discusses future work.

2. *k*-NN framework

K nearest neighbor (*k*-NN) is a traditional, effective and a kind of lazy learning approach which is proposed in [2]. Because of its easiness to implement and satisfied performance, *k*-NN remains a popular learning scheme. As the LSHTC issue becomes a hot topic in text classification, *k*-NN is found very suitable for LSHTC's characteristics. For example, when the class label taxonomy consists of quite a large number of categories, *k*-NN does not need to train enormous classifiers which are sometimes intractable due to the extremely high computing complexity. In other words, *k*-NN is not sensitive to the scale of the class label taxonomy. The second advantage of *k*-NN lies in that it possesses an intrinsic attribute to deal with multiple label classification tasks for multiple labels are ready to obtain from the retrieved nearest neighbors. On the one hand, a retrieved neighbor with multiple labels will directly provide with multiple labels. On the other hand, multiple neighbors together contribute to multiple labels. The third advantage is that *k*-NN framework directly deals with training samples and it can directly retrieve the final or the lowest level of label in the class label hierarchy without the level-by-level retrieval process. Level-by-level hierarchical framework usually leads to error propagation.

However, *k*-NN framework has its own shortcomings. One of them is that the computational cost is largely decided by the sample scale in training dataset. Thus we use parallel *k*-NN framework to alleviate this difficulty, in which the training dataset is divided into parts and nearest neighbors are retrieved from these parts in parallel and a final result is synthesized from these part results. In this way, we pay computing resource cost in return for general efficiency and make parallel *k*-NN framework applicable and tractable to LSHTC tasks.

Although *k*-NN algorithm is easy to implement, it has quite a lot of variations. Mainly the variations involve two important aspects. One is about similarity measures or distance metrics and the other is related to ranking approaches for a multiple label classification task. In addition to that, the value of k should be determined using a developing or evaluation dataset.

2.1. Similarity measures

In this paper, only the bag of words (BOW) is used to model the text. In BOW model, we only pay attention to occurring frequency of a term in a text and the sequence or order of terms in a text is neglected. Upon BOW model, the text is further represented in the vector space model (VSM), which is prevalent in data mining domain. Thus, in the VSM, a text is mapped into a vector. In a *k*-NN scheme, a test sample's *k* nearest neighbors in the training set are retrieved by their similarity or distance to the test sample. The samples with the larger similarity or shorter distance to the test sample are given higher priority in the retrieving queue. There are a variety of similarity or distance measures, such as Euclidean distance, cosine distance and BM25.

Euclidean distance is defined as follows,

$$Eucliean_{v_m v_n} = \sqrt{\sum_i (v_{mi} - v_{ni})} \quad , \qquad (1)$$

where v_m and v_n represent two text vectors whose distance is to be calculated. v_{mi} and v_{ni} are the counterparts in these two vectors, respectively.

In a text vector $(w_{1,i}, w_{2,i}, \cdots, w_{m,i})$ representing a document *i*, each term t_j in the term set, which can be computed from the training set, is indexed by term frequency-inverse document frequency (TFIDF) as follows,

$$w_{j,i} = (\log(tf_j) + 1)\log\frac{N}{df_i} \quad , \qquad (2)$$

where tf_j is the term frequency of t_j in text i, df_i is the number of documents containing term t_j, N is the total number of documents in the training set. If $tf_j = 0$, then $w_{j,i}$ will be set to zero.

Cosine distance is commonly used in VSM, which is considered as a measurement of the angle between two vectors. Given two document vectors v_m and v_n, the cosine distance is computed as,

$$Cos_{v_m, v_n} = \frac{v_m \bullet v_n}{\|v_m\| \|v_n\|} \qquad (3)$$

BM25 is a popular weighting approach in information retrieval [3]. In the BM25 weighting scheme, the input document is regarded as a query *q* containing a number of terms $\{t_1, t_2, \cdots, t_m\}$. The BM25 similarity between a query *q* and a document *d* in training set is computed as follows,

$$Sim_{BM25} = \sum_t \frac{tf_t^d (a+1) \bullet w_t \bullet tf_t^d}{tf_t^d + a(1 - b + b \bullet \frac{dl}{avgdl})}, \quad (4)$$

where tf_t^d and tf_t^q are term t's frequencies in document *d* and *q* respectively. $avgdl$ is the average document length in the training set. a and b are term frequency influence parameter and text normalization influence parameter respectively and they are set to 1.5 and 1.0 by default. w_t is the inverse document frequency (IDF) of term t.

3. Hierarchical SVM framework

SVM remains one of the best classification algorithms for its high performance among various text classification tasks. However, when the scale of class label taxonomy is very large, e.g. tens of thousands, the training of SVM classifiers becomes intractable for that in a flat SVM framework, a binary SVM classifier should be trained for each of the class pair so there are tens of thousands of SVM classifiers should be trained, which becomes an intractable task.

To solve the aforementioned problem, a hierarchical SVM is used. As is mentioned in [4], the work flow of a hierarchical SVM framework is that it first performs TFIDF feature indexing on the samples [5], then searches for the optimal parameters of SVM on each node, then trains a decision tree of SVMs, at last uses it to classify test samples.

As is described in [6], when category label is to be predicted for a test sample, it is first tested by the root classifier, which predicts its scores on labels at the first level. Then the test sample is tested by the classifiers which are chosen by former prediction. Thus a test sample goes down through the SVM classifier network until it reaches the last level. And the output queue is sorted by the score sum throughout all levels.

4. Two LSHTC Tasks

We consider two LSHTC tasks. The first one is from NTCIR-8 patent mining task's Japanese subtask, which is used to compare the aforementioned two classification frameworks, namely parallel *k*-NN framework and hierarchical SVM framework. The second one is a self-composed task, which is used to compare different similarity measure and ranking approaches and to investigate the value selection of k for k nearest neighbors and the value selection of decay factor in the ListWeak ranking approach.

4.1. NTCIR-8 patent mining task's Japanese subtask

In the NTCIR-8 (Japanese National Information Institutes' Testing Corpus for Information Retrieval) patent mining task's Japanese subtask [7], the train set includes approximately 3.5 million patent documents over 10 years from 1993 to 2002 and each patent's document has a title and three fields: abstract, claim and description. The test set consists of 600 or so academic paper abstracts. The corpus is publicly available for research purpose. The class label taxonomy here uses IPC (International Patent Classification) scheme. IPC is a complex hierarchical classification system, which divides all the technological fields into 8 sections, 120 classes, 630 subclasses and approximately 69000 groups.

The task is to use patent documents in the train set to predict one or a set of IPC labels for research paper abstracts in the testing set. Seen from Table 1, there are 3496137 and 549 examples in train and test respectively. And there are altogether 57913 classes involved in the data set.

TABLE1. EXAMPLES AND CLASS SCALES IN TASK1 AND TASK2

	train set scale	test set scale	class scale
in task 1	3496137	549	57913
in task 2	3487511	8626	57913

4.2. Japanese patent classification task

Because in the aforementioned task, the test and train examples are from different genres, namely patent documents and research paper abstracts, additional genre factor is mixed into LSHTC task. To alleviate the genre influence and focus on LSHTC task, we compose another LSHTC task, i.e. Japanese patent classification task. The different between this task and the aforementioned task lies in that patent documents are used to predict IPC labels for patent documents instead of research paper abstracts.

The original patent set has 3496137 examples. We randomly select 8626 examples to form the test set and the left 3487511 examples become the train set. Same as in task 1, there are altogether 57913 classes involved in the data set.

5. Experiments and results

All of the experiments were performed on the Super Cubic of Shanghai Supercomputer Center, a cluster-based mainframe computer system. Two nodes are used, with each node having 16 AMD Barcelona 1.9 GHz CPU cores and 64 Gbytes share memory. The operating system used is SuSE Linux Enterprise Server (SLES) 10 SP2. We implement the parallel *k*-NN framework on the first LSHTC task. And the hierarchical SVM framework's results can be obtained through [4] and [5]. We compare the above results.

5.1. Preprocessing

To prepare proper data for our experiements, we conduct several steps to transform the Japanese patent documents into vectors. The first step is tokenization. In this step, nouns, verbs and adjectives are extracted from four fields of original structured patent document –Title, Abstract, Claim and

Description as isolated words by using the software Chasen [8]. The tokenized words are called *term*s in the research domain of text classification. The second step is indexing. In this step, the tokenized words, namely terms, are transformed to numerical value and thus a patent document is represented by a vector. We also delete 115 patent documents without class labels from the original 3496252 patent documents.

5.2. Ranking approaches

Ranking refers to the process in which labels of the *k* nearest neighbors retrieved will be processed to predict one label or labels for the test example. After *k* nearest neighbors are retrieved, a score is calculated for each category c. The larger the score is, the higher the probability of assigning the corresponding category label to the test sample becomes. Finally, category labels are sorted according to their scores and output as a ranked list. The difference between ranking approaches lies in that how scores of retrieved labels are computed.

Some most popular ones are listed as follows:

(1) Simple ranking approach

In this approach, ranking is implemented by simple voting by retrieved k nearest neighbors.

$$Score_{simple}(c)=\sum_{i=1}^{k} f(c,n_i) \quad ,(5)$$

where $f(c,n_i)$ denotes neighbor i in the k retrieved nearest neighbors. $f(c,n_i)$ is a $0-1$ function that takes value as 1 if c doesn't exist in n_i and takes value as 0 if c exists in n_i respectively.

(2) First Emerge ranking approach

In this approach, ranking is only decided by the class order in which the class first emerge in the retrieved k nearest neighbors.

$$Score_{firstemerge}(c)=\frac{1}{f(c,\{n_1,n_2,\cdots,n_k\})} \quad ,(6)$$

where $\{n_1,n_2,\cdots,n_k\}$ is the sorted sequence of the *k* retrieved nearest neighbors. If c first emerges in the neighbor n_i, then $f(c,\{n_1,n_2,\cdots,n_k\})$ takes value as i.

(3) Total ranking approach

In this approach, ranking is to sum up a class label's similarities among the *k* retrieved nearest neighbors to compute the score for a class label and then sort these scores.

$$Score_{total}(c)=\sum_{i=1}^{k} f(c,n_i)*sim(t,n_i) \qquad (7)$$

Where $sim(t,n_i)$ denotes the similarity score between the test document t and the retrieved neighbor t. The definition of function $f(c,n_i)$ is the same as the one in simple ranking approach.

(4) ListWeak ranking approach

This approach is originated from information retrieval, which is one of the best ranking strategies [9].

$$Score_{listweak}(c)=\sum\nolimits_{i=1}^{k} f(c,n_i)\bullet sim(t,n_i)\bullet r^i \quad ,(8)$$

where the definition of function $f(c,n_i)$ and $sim(t,n_i)$ are same as the aforementioned ones. r^i is a decay factor which penalizes the samples which have lower ranks. r is set to 0.95 by default.

5.3. Evaluation measure

Mean Average Precision (MAP) is adopted to measure how well the ranked list matches the gold standard [10]. The MAP is calculated in the following way.

$$MAP_{value} \frac{\sum_{i=1}^{n} AverP(t_i)}{n} \qquad (9)$$

$$AverP(t_i)=\frac{\sum_{r=1}^{N} P_i(r)*reli(r)}{m} \qquad (10)$$

where *ti* is the *i*-th example in test set, *n* is the total number of the test examples. For a given topic *ti*, *r* is its rank in the list, *Pi(r)* is the precision at *r* which is obtained by dividing the number of correct IPC labels in top-r of the ranked list by *r*. *reli(r)* is a 0-1 function that indicates the relevance of rank r. If the IPC code at rank r is correct, *reli(r)=1*; otherwise *reli(r)=0. m* is the number of correct IPC labels for *ti*, and *N* is the maximal rank that is evaluated. In this task, *N* is set to 1000.

5.4. Comparison results on the NTCIR-8 patent mining task's Japanese subtask

In NTCIR-8 patent mining task, We get 3496117 patent documents in the train set. We use different similarity measures and ranking approaches. With regards to similarity measures, we find that BM25 outperforms the others. Regarding ranking, we find ListWeak performs best among the candidate ranking approaches. We use different parts of patent documents to conduct indexing and find that using the whole text is the most effective. After retrieving k nearest neighbors, we use the first label and all labels of neighbors respectively and we find an interesting phenomenon that using all of the labels of the retrieved neighbors instead of the main label can gain a 6 percents raise in MAP evaluation measure.

The MAP values for *k*-NN framework and hierarchical SVM framework are 42.40 and 29.94 respectively.

5.5. Comparison results on Japanese patent classification task

TABLE 2. COMPARISON OF DIFFERENT SIMILARITY MEASURES

similarity measures	Euclidean distance	Cosine distance	BM25
MAP	41.09	51.37	61.64

To eliminate the genre influence and focus on the LSHTC task itself, experiments are conducted on this task to investigate the selection of the value of k and decay factor, and to compare similarity measures and ranking approaches and different use of retrieved labels.

TABLE 3. COMPARISON OF DIFFERENT RANKING APPROACHES

ranking approaches	Simple	FirstEmerge	Total	ListWeak
MAP	54.00	4.02	55.18	61.64

We compare three similarity measures listed in Table 2, which shows that BM25 outperforms the other two remarkably. We also compare four ranking approaches listed in Table 3, which shows ListWeak is the most effective. Comparison of different use of labels is also conducted and the result shown in Table 4 reveals an interesting phenomenon that it is more effective to use all labels of the retrieved neighbors instead of just using the first label of the retrieved neighbors.

TABLE 4. COMPARISON OF DIFFERENT USE OF LABELS

labels used	the first label	all labels
MAP	54.94	61.64

We also investigate the value selection of k in *k*-NN and decay factor r in ListWeak ranking approach. From Figure 1, we can see that as k increases, MAP increases accordingly. However, the increase becomes weak after k exceeds 100.

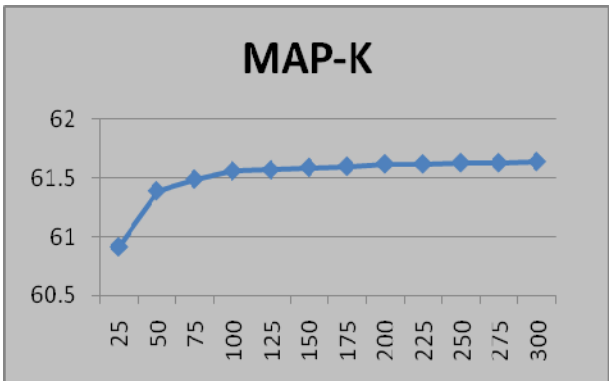

Figure 1. The relation between MAP and k

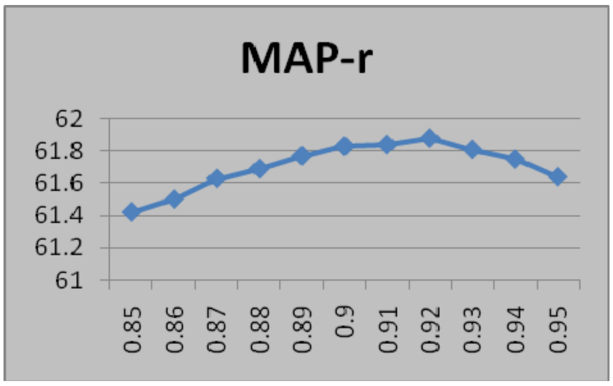

Figure 2. The relation between MAP and r

We also investigate the relation between MAP and decay factor r. From Figure 2, we come to the conclusion that there exists a single global optimal value for r, which makes MAP take the highest value.

6. Conclusions and Future work

In this paper, we conduct some comparative studies using two LSHTC tasks. First we compare two frameworks, namely *k*-NN framework and hierarchical SVM framework to deal with the first LSHTC task, namely NTCIR-8 task. By comparison, we come to the conclusion that to deal with this LSHTC task, further, this multiple label LSHTC task, *k*-NN framework outperforms hierarchical SVM framework, which is quite different from the usual results for normal text categorization tasks. *k*-NN framework is insensitive to the complex hierarchical structure and the scale of class label taxonomy. Moreover, *k*-NN framework is born to deal with multiple label classification tasks. And using parallel implementation of *k*-NN framework overcomes the traditional *k*-NN's shortcoming of being sensitive to the scale of train data. Thus *k*-NN framework becomes an ideal solution to multiple label LSHTC task. Second, from the comparative results on the second LSHTC task, we draw the conclusion that BM25 outperforms other similarity measures and ListWeak is more effective than all other ranking approaches. An interesting found is that using all the labels of the retrieved neighbors can remarkably improve classification performance over using only the first label of the retrieved neighbors.

However, due to page limit, the feature selection, term weighting and sample selection issues are not considered in this paper, which may be studied in the future to further improve the solutions to LSHTC tasks.

Acknowledgements

This work is partially supported by the National Natural Science Foundation of China (Grant No. 60903119, Grant No. 60773090 and Grant No. 90820018), the National Basic Research Program of China (Grant No. 2009CB320901), and the National High-Tech Research Program of China (Grant No. 2008AA02Z315).

References

[1] Joachims T., “Text categorization with support vector machines: learning with many relevant features”, Proceeding of the European Conference on Machine Learning (ECML), pp. 137-142, 1998.

[2] Cover T. M. and Hart P. E., “Nearest Neighbor Pattern Classifications”, IEEE Transaction on Information Theory, Vol 13, No. 1, pp. 21-27, 1967.

[3] Robertson, Stephen E., Steve Walker, and Micheline Hancock-Beaulieu, “Okapi at TREC-7: automatic ad hoc, filtering, VLC and interactive track”, Proceeding of the seventh Text Retrieval Conference, Gaithersburg, USA, pp. 253-264, 1998.

[4] Wang Xiao-Lin and Lu Bao-Liang, “Improved Hierarchical SVMs for Large-scale Hierarchical Text Classification Challenge”, Proceeding of Large-Scale Hierarchical Classification Workshop related to the Pascal2 LSHTC Challenge, 32nd European Conference on Information Retrieval (ECIR), UK, 2010.

[5] Sebastiani F., “Machine Learning in Automatic Text Categorization”, ACM Computing Surveys, Vol. 34, pp. 547, 2002.

[6] Jin G., Kong Q., Zhang J., Wang X., Hui C., Zhao H. and Lu B. L., “Multiple Strategies for NTCIR-08 Patent Mining at BCMI”, Proceeding of NTCIR-8 Workshop Meeting, Tokyo, Japan, 2010.

[7] http://research.nii.ac.jp/ntcir/.

[8] http://chasen.naist.jp/hiki/ChaSen/.

[9] Xiao T., Cao F., Li T., Song G., Zhou K., Zhu J. and Wang H., “KNN and Re-ranking Models for English Patent Mining at NTCIR-7”, Proceeding of NTCIR-7 Workshop Meeting, Tokyo, Japan, pp. 333-340, 2008.

[10] Baeza-Yates R. A. and Ribeiro-Neto B. A., “Modern Information Retrieval”, ACM Press/Addison-Wesley, 1999.