ICL at NTCIR-7: An Improved KNN Algorithm for Text Categorization

Wei Wang
Inst. of Computational Linguistics, Peking University
wwei@pku.edu.cn

Sujian Li
Inst. of Computational Linguistics, Peking University
lisujian@pku.edu.cn

Chen Wang
Inst. of Computational Linguistics, Peking University
goldeneagle@pku.edu.cn

Abstract

This paper describes our system for the NTCIR-7 Patent Mining Task which sought to make automatic text classification pragmatic. Our system employs an improved KNN algorithm which makes trade-off between effectiveness and time complexity. We have tried two distance metrics in our algorithm: cosine similarity and Euclid distance. Evaluation results on NTCIR-7 test data shows that the former one is slightly better.

1. Introduction

The NICIR Patent Mining Task (MT) aims to develop techniques for technique trend analysis and mining. In those techniques, text categorization is a primary step for further processing. In NTCIR-7 MT English subtask, the participant is required to categorize research papers (abstracts) written in English into the International Patent Classification (IPC). IPC has a hierarchical structure, each IPC code is composed of five parts: section, class, subclass, group and subgroup. For example, the IPC code "H04H 1/06" can be divided as follows:

Table 1: An example of IPC code

	Code	Meaning
H	Section	Electricity
04	Class	Electric communication technique
H	Subclass	Broadcasting
1	Group	Broadcast distribution system
06	Subgroup	Having frequencies in two or more frequency bands

MT is basically a text categorization task. A variety of methods have been developed for text categorization. The most common way is to use VSM (vector space model) to represent text and statistic learning models to do the classification job. SVM is reported to achieve best performance over a wide range of tasks [1]. However, the training of SVM is time-consuming, especially when there are too many categories and features [2]. For example, if we use one-against-rest SVM, we will need about 30000(the number of IPC codes used in our task) classifiers. Tie-Yan liu [2] reports that it takes 102 hours to train flat SVMs for just 94 categories over OOSUMED [3]. The computational load will be unacceptable if we use such models. To tackle this problem, we turn to KNN model. For simplicity, we use an improved nearest neighbor searching algorithm which calculates the distance between the unlabeled node and "centroid" node instead of all nodes. The "centroid" node represents all nodes belonging to the same category. Experiments show that our method achieves promising results on the NTCIR-7 test data.

The rest of this paper is organized as follows. Section 2 briefly introduces our system. Section 3 describes the basics of our KNN algorithm including complexity analysis and detailed steps. Section 4 illustrates the experimental results and section 5 concludes the paper.

2. System Overview

Our system uses PAJ (Patent Abstracts of Japan) in 1993-2002 as training data. The following is a patent file instance:

```
<PATDOC>
<B110>05000001</B110>
<B511>  A01B 63/10   </B511>
<B512>  A01C 11/02   </B512>
<B542>ROLLING CONTROLLER OF WORKING MACHINE</B542>
<SEC>
<P>PURPOSE: To improve following performance of working machine to ground by restraining rolling control to local unevenness on the ground. </P>
…
<P>COPYRIGHT: (C)1993,JPO&amp;amp;Japio</P>
</SEC>
</PATDOC>
```

Figure 1: An example of patent file

"05000001" in <B110> tag is the patent number, which is unique for each patent. "A01B 63/10" and "A01C 11/02" are the IPC codes for this patent (patent

can have more than one IPC code). The < B542> tag contains the patent title, and <SEC> tag contains the main content.

First we extract texts from<B542> and <SEC> tags. The copyright declaration in <SEC> is discarded. Then we calculate IDF for each word contained in patents. After that, we merge patent files sharing the same IPC code. Merged files and the topic file are represented with term vectors respectively. Then we find 1000 nearest neighbors (most possible IPC codes) for each topic vector using two different distance metrics. The detailed algorithm will be illustrated in section 3.

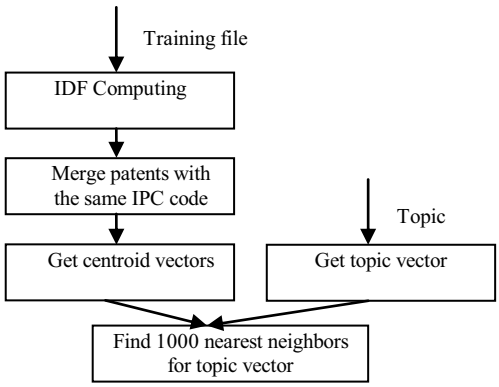

Figure 2: The processing steps of our system

3. The Improved KNN Algorithm

In this section we will describe the algorithm employed in our system, including complexity analysis and detailed steps. Our algorithm is improved mainly on the time cost. Whether the categorization result benefits from it remains to be resolved.

3.1. Complexity analysis of our algorithm

Tradition KNN calculates distance between each labeled node and node to be classified. Suppose there are N labeled nodes, and M different labels. The complexity of finding the most possible label for an unlabeled node from K nearest neighbors is N*logK. When N is large, the computation work will be time consuming. In NTCIR-7, there are about 5 million training patent files, and 879 topics. We have to assign each topic 1000 most possible IPC codes. Using the original KNN model, the computational expense is too large to afford. So we come up with an improved algorithm which makes trade-off between effectiveness and time cost. Our algorithm uses the "centroid node" to represent all nodes that have the same IPC code, and we search 1000 nearest neighbors for each topic from "centroid" nodes instead of the 5 million nodes. The computational cost of this method for a single topic is M*log(1000). If we use the original KNN algorithm, since we have to find 1000 most possible labels from K nearest neighbors, K must be very large to contain enough nodes of different labels. Normally K will be N. So the time cost is almost ($N \approx 5*10^6$, $M \approx 3*10^4$):

$$\frac{N*\log(N)}{M*\log(1000)} \approx 7.5*10^2$$

times of our improved algorithm. In the following we will show more details of our algorithm.

3.2. Detailed steps of our algorithm

Assume the words are w_1 , w_2 … w_t with IDF idf_1 , idf_2 … idf_t respectively, the topic vector is V_{topic} =< idf_1 * $topictf_1$, idf_2 * $topictf_2$ … idf_t * $topictf_t$ > . First we will have to denote the "centroid node" for the i[th] merged text. Suppose the i[th] merged text is composed of c_i independent patent files, which means there are totally c_i different patents labeled with the i[th] IPC code. We define the "centroid" vector for it as V_i =< idf_1 * $tf_{i,1}/c_i$, idf_2 * $tf_{i,2}/c_i$ …, $idf_t * tf_{i,t}/c_i$ >, while $tf_{i,j}$ is the term frequency of the j[th] word in the i[th] merged text. After that, we can calculate the distance or similarity between the centroid vector V_i and the topic vector V_{topic} . The first metric is Euclid distance:

$$Euclid(V_i, V_{topic}) = |V_i - V_{topic}|$$

The Euclid distance metric prefers long topics because they have more words than short ones. The cosine similarity metric can avoid such problems since it's normalized.

$$Cosine(V_i, V_{topic}) = \frac{V_i . V_{topic}}{|V_i| * |V_{topic}|}$$

Notice that smaller Euclid distance means higher possibility of correlation between topic and IPC code while cosine similarity is on the contrary.

Here are the detailed steps of our algorithm:

1. Extract patent file contents from raw data, compute the IDF for each word using the formula $IDF_{term-i} = log(N/D_i)$. N is total number of patent files, and D_i is the number of documents containing term i. In order to remove noise features we discard words appearing in less than 3 patents.
2. Merge patents with the same IPC code into one file. Count the merged files' term frequency and get the "centroid" vector V_i for the i[th] merged file ($i = 1, 2, ...M$).
3. Get the term vector V_{topic} for each topic. Then calculate the distance or similarity between V_i

and V_{topic} . Construct a heap of size 1000 in the memory which records the 1000 vectors with minimum distances/maximum similarity and their corresponding IPC codes while computing the distance/similarity between V_{topic} and each "centroid" vector. After the computing is done, get the IPC codes in the heap. Their rank is determined by the corresponding distance or similarity score.

4. Evaluations

The following evaluation is two-fold. First we will compare the performance of our system with other participants. Each participant can submit at most 3 results. Of the 20 results submitted by nine participants, our best result ranks 12th. Second we will compare the performance of the two distance metrics: Euclid distance and cosine similarity. Evaluation results show that the cosine similarity metric is better on average.

4.1. Comparison of our system and other systems

The NTCIR-7 task contains 879 topics. Each user can submit at most 1000 IPC codes for a single topic. Usually there are 1-3 IPC codes given by the organizer for each topic as the answer. There are totally 2051 answers for 879 topics. Our system retrieves 1888 of them. Here is a comparison of our system's retrieved answer number with other participants:

Table 2: Comparison of retrieved answer number

Participant	Retrieved IPC (Relevant 2051)
NEUN1_S1	1975
xrce_e2j2e	1932
KECIR	1892
ICL07_1	1888
nttcs2	1848
BRKLY-PM-EN-02	1488
AINLP04	1455
rali1	953
PI-5b	895

The IPC numbers showed in Table 2 are the maximum numbers retrieved by each participant. From this table we can see that the top 5 results (including ours) show no significant difference with each other. In fact, our system's gap with other top systems lies in the precision indicators. There are mainly three kinds of precision measurements for this task. The first is the average interpolated recall precision (I-precision). Suppose there are N topics, for topic i, the maximum precision when the recall is bigger than an interpolated value, say 0.50, is P_i . Then the I-precision at 0.50 is defined as $\sum_{i=1}^{N_{topic}} P_i / N_{topic}$ (N_{topic} is the number of topics). Here is the detailed average interpolated recall precision of our system and three other top systems.

Table 3: Comparison of I-precision

Interpolated Value	ICL07_1	NEUN1_S1	xrce_e2j2e	KECIR
0.00	0.2118	0.5965	0.5318	0.3973
0.10	0.2118	0.5965	0.5318	0.3973
0.20	0.2068	0.5936	0.5302	0.3949
0.30	0.1922	0.5718	0.5075	0.3721
0.40	0.1613	0.5308	0.4658	0.3300
0.50	0.1587	0.5254	0.4555	0.3201
0.60	0.1142	0.4522	0.3821	0.2507
0.70	0.1021	0.4183	0.3536	0.2212
0.80	0.0980	0.4085	0.3469	0.2113
0.90	0.0962	0.4029	0.3424	0.2062
1.00	0.0961	0.4027	0.3424	0.2062

The micro average interpolated recall precision (micro I-precision) is similar with average interpolated recall precision. The only difference is that the recall and precision is calculated using all topics. For example, the micro I-precision at 0.50 is defined as $max\{\frac{1}{N_{topic}}\sum_{i=1}^{N_{topic}} c_{i,k} / k \ , 1 \le k \le 1000\}$ when the overall recall $\sum_{i=1}^{N_{topic}} c_{i,k} / N_{answer}$ is bigger than 0.50($c_{i,k}$ is the number of correct answers in top k results for topic i, N_{answer} is the number of correct answers for all topics). Here is the detailed micro I-precision of our system and three other top systems.

Table 4: Comparison of micro I-precision

Interpolated Value	ICL07_1	NEUN1_S1	xrce_e2j2e	KECIR
0.00	0.1024	0.4664	0.4107	0.2708
0.10	0.0846	0.4664	0.4107	0.2708
0.20	0.0556	0.3874	0.3305	0.1862
0.30	0.0417	0.3874	0.2704	0.1486
0.40	0.0312	0.3201	0.2392	0.1090
0.50	0.0230	0.2353	0.1669	0.0744
0.60	0.0163	0.1770	0.1007	0.0468
0.70	0.0112	0.1097	0.0609	0.0252
0.80	0.0062	0.0519	0.0293	0.0124
0.90	0.0027	0.0149	0.0075	0.0038
1.00	0.0000	0.0000	0.0000	0.0000

From Table 4 we can see that all the four systems' micro I-precision at 1.00 are zero, since no system has retrieved all answers.

The third precision (doc-precision) is defined as the precision at a certain number of retrieved answers. For example, the doc-precision at 10 means the average precision of the top 10 answers for all topics. Due to limited pages we don't present detailed evaluation

results. The situation is similar: our system is significantly lower than the top systems.

Our system's low precision indicates that our rank function is not good enough. It's partly because of the model we use. Our KNN model is "flat", that is, it treats all categories equally. However, the IPC has a hierarchical structure. By using flat KNN, the structure information is lost. Besides, currently we haven't done much work on feature selection and term weight adjusting. We just simply use words as features, and IDF as term weights. This may not be appropriate sometimes, for example, words in patent titles are usually more informative than words in claims, so maybe we should assign such words higher weights.

4.2. Comparison of two distance metrics

We have submitted two results, using cosine similarity and Euclid distance as the distance metric respectively. The former one retrieves 1888 answers, while the other retrieves 1277. The precision of cosine similarity is also higher than that of Euclid distance.

Table 5: Comparison of I-precision and micro I-precision for two distance metrics

Recall	I-precision		micro I-precision	
	Cosine	Euclid	Cosine	Euclid
0.00	0.2118	0.2094	0.1024	0.1149
0.10	0.2118	0.2094	0.0846	0.0914
0.20	0.2068	0.2058	0.0556	0.0535
0.30	0.1922	0.1899	0.0417	0.0319
0.40	0.1613	0.1543	0.0312	0.0173
0.50	0.1587	0.1509	0.0230	0.0060
0.60	0.1142	0.0967	0.0163	0.0019
0.70	0.1021	0.0845	0.0112	0.0000
0.80	0.0980	0.0807	0.0062	0.0000
0.90	0.0962	0.0796	0.0027	0.0000
1.00	0.0961	0.0796	0.0000	0.0000

However, for the micro I-precision, the Euclid distance metric is higher than Cosine similarity metric at recall 0.00 and 0.10. This means that for the few top answers, the Euclid distance metric's answer is more accurate. It's more obvious when compared with doc-precisions.

Table 6: Comparison of doc-precision for two distance metrics

Top K answers	Cosine	Euclid
5	0.0710	0.0719
10	0.0536	0.0501
15	0.0447	0.0397
20	0.0388	0.0337
30	0.0312	0.0261
100	0.0147	0.0107
200	0.0087	0.0058
500	0.0040	0.0027
1000	0.0021	0.0015

We can see that for the top 5 answers, the Euclid distance metric is higher. This is possibly because it can take into account negative features, that is, if two words don't appear in two texts, their term vectors' Euclid distance will be smaller, on the contrary, only common words in two texts make contribution to cosine similarity.

5. Conclusion and future work

Much work remains to be done. Our current model is simple, effective, at the cost of information loss. Negative features, and the hierarchical structure of IPC, are ignored by our system (especially for the cosine similarity metric). In the future, we will try to take into account these factors. Also our feature selection method needs to be improved. Currently, we just simply discard words with DF (document frequency) less than 3. We will try more sophisticated methods, like IG, MI and LSI to remove noise features and redundant features. Besides, we will use mixed models to improve the precision of our system. That is, we will first use the KNN model to extract top 1000 IPC codes. Then, we will use other models like ME or SVM to rescore each IPC code.

Acknowledgement

This work is supported by NSFC programs (No: 60603093 and 60875042), and 973 National Basic Research Program of China (2004CB318102).

References

[1] SU Jin-Shu, ZHANG Bo-Feng, XU Xin. Advances in Machine Learning Based Text Categorization. *Journal of Software*, Vol. 17, No.9, September 2006, pp.1848-1859

[2] Tie-Yan liu, Yiming Yang, Hao Wan, Hua-Jun Zeng, Zheng Chen, and Wei-Ying Ma. Support Vector Machines Classification with A very Large-scale Taxonomy. *SIGKDD 05 Explorations*, Volume 7, Issue 1-Page 36-43,

[3] Hersh, W., Buckley, C., Leone, T., and Hickam, D. OHSUMED: An interactive retrieval evaluation and new large test collection for research. *SIGIR*, 192-201, 1994.

[4] Lam W, Lai KY. Automatic textual document categorization based on generalized instance sets and a metamodel. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2003,25(5):628−633.

[5] Tsay JJ, Wang JD. Improving linear classifier for Chinese text categorization. *Information Processing and Management,* 2004,40(2): 223−237.