A Refined Weighted K-Nearest Neighbors Algorithm for Text Categorization

Fang Lu
College of Mathematics and Computer Science
Fuzhou University
Fuzhou, China
luf_hi@126.com

Qingyuan Bai *
College of Mathematics and Computer Science
Fuzhou University
Fuzhou, China
baiqy@fzu.edu.cn

Abstract—**Text categorization is one important task of text mining, for automated classification of large numbers of documents. Many useful supervised learning methods have been introduced to the field of text classification. Among these useful methods, K-Nearest Neighbor (KNN) algorithm is a widely used method and one of the best text classifiers for its simplicity and efficiency. For text categorization, one document is often represented as a vector composed of a series of selected words called as feature items and this method is called the vector space model. KNN is one of the algorithms based on the vector space model. However, traditional KNN algorithm holds that the weight of each feature item in various categories is identical. Obviously, this is not reasonable. For each feature item may have different importance and distribution in different categories. Considering this disadvantage of traditional KNN algorithm, we put forward a refined weighted KNN algorithm based on the idea of variance. Experimental results show that the refined weighted KNN makes a significant improvement on the performance of traditional KNN classifier.**

Keywords-text categorization; vector space model; weight calculation; KNN

I. INTRODUCTION

Today, with the rapid development of the Internet, textual information available grows rapidly. So text mining which aims to find and organize relevant information in text collections is needed. While text categorization is one important task of text mining, aiming to automatically give pre-defined labels to large numbers of previously unlabelled documents. Since 1950s text categorization has been studied, and now there are many useful techniques and methods for it. While the dominant approach to text categorization is based on machine learning techniques: a general inductive process automatically builds a classifier by learning from a set of pre-labeled documents. There are many well-known and useful algorithms for text categorization, such as Support Vector Machine (SVM), Naïve Bayes (NB), K-Nearest Neighbor (KNN), Neural Network (NNet), Linear Least Squares Fit (LLSF) and so on. Among all these algorithms, KNN is a widely used text classifier for its simplicity and efficiency. In its training-phase, it does nothing more than storing all the training examples as a classifier, so it is often called as lazy learner since 'it defers the decision on how to generalize beyond the training data until each new query instance is encountered'[1].

For text classification, one document is often represented as a vector. The elements of the vector are a series of selected important words called feature items. This method is called the vector space model and KNN is a text classifier based on this model. However, traditional KNN is based on the assumption that the importance of each feature item in various categories is equal. Obviously, this is not so reasonable. For each feature item may have different importance and distribution in different categories. So, how can we find a method to get the different weight of each feature item in different categories? Then a refinement strategy for KNN called weighted KNN has been proposed. The key point of this problem is how to get a weight value system for weighted KNN. Some researches have been done for this problem[2][3][4], but most of them are too complicated and often need too much running time. Then we propose a new easy method for the weighted KNN algorithm and get the weight value system based on the idea of variance. It holds that if one feature item distributes more evenly in one category and less evenly among all categories, then this feature item is more important for this category and the weight value of this feature item is higher. Experiments show that our refined weighted KNN algorithm has better performance in text categorization.

The rest of this paper is organized as follows. Section 2 briefly introduces the vector space model for text categorization. Section 3 describes the traditional KNN algorithm for text categorization. Our proposed refined weighted KNN classifier is presented in Section 4. Experimental results are reported in Section 5. Finally, Section 6 concludes this paper.

II. THE VECTOR SPACE MODEL

In this section, we give a brief introduction of vector space model for text categorization. In vector space model, one document is often represented as a vector like $\vec{d} = (d_1, d_2, ..., d_n)$.

* Corresponding author.

Each element d_i in $\vec{d}$ represents a word t_i called as feature item and can be calculated with many methods.

One of the useful calculation methods is the combination of *term frequency* (*tf*) and *inverse document frequency* (*idf*), i.e., $d_i = tf(\vec{d}, t_i) * idf(t_i)$. In this paper, we adopt the formula of *tf* and *idf* used in the Cornell SMART system as follows[5]:

$$tf(\vec{d}, t_i) = \begin{cases} 0, & if \quad freq(\vec{d}, t_i) = 0 \\ 1 + \log(1 + \log(freq(\vec{d}, t_i))), & others \end{cases} . \qquad (1)$$

Where $freq(\vec{d}, t_i)$ is the number of times that word t_i occurs in document $\vec{d}$.

$$idf(t_i) = \log \frac{1 + |D|}{|df(t_i)|}, \qquad (2)$$

Here $|D|$ is the total number of documents and $df(t_i)$ is the number of the documents where word t_i occurs at least once.

III. The KNN Algorithm for Text Categorization

KNN is a lazy learning instance-based method that it just stores all the training examples as a classifier. Then to classify an unknown document $\vec{x}$, the KNN algorithm finds the document's k nearest neighbors from the training documents. With the class labels of these k nearest neighbors, the class of the document can be predicted. The formal KNN algorithm describes as follows:

1) Storing all the training documents ($\vec{d}_1, \vec{d}_2, ..., \vec{d}_m$). $\vec{d}_i$ denotes one training document.

2) When an unknown document $\vec{x}$ comes, rank the training documents by the similarity of each one with document $\vec{x}$. Then we get the k most similarity documents of $\vec{x}$. The similarity is calculated with cosine distance as follows:

$$sim(\vec{x}, \vec{d}_i) = \frac{\sum_{k=1}^{N} x_k \times d_{ik}}{\sqrt{(\sum_{k=1}^{N} x_k^2)(\sum_{k=1}^{N} d_{ik}^2)}} . \qquad (3)$$

Where N is the total number of feature items.

3) With the k most similarity documents of $\vec{x}$, the class weight of $\vec{x}$ for each class C_j can be calculated as follows:

$$p(\vec{x}, C_j) = \sum_{\vec{d}_i \in KNN(\vec{x})} sim(\vec{x}, \vec{d}_i) y(\vec{d}_i, C_j) . \qquad (4)$$

Here $KNN(\vec{x})$ denotes the set of k most similarity documents of $\vec{x}$. While $y(\vec{d}_i, C_j)$ indicates the classification of the document $\vec{d}_i$ for class C_j as follows:

$$y(\vec{d}_i, C_j) = \begin{cases} 1, & \vec{d}_i \in C_j \\ 0, & \vec{d}_i \notin C_j \end{cases} . \qquad (5)$$

4) Finally, compare the class weight of $\vec{x}$ for all classes, and $\vec{x}$ is classified to the class with the maximal class weight $p(\vec{x}, C_j)$. That is,

$$C = \arg \quad \max_{C_j} (p(\vec{x}, C_j)) . \qquad (6)$$

Supposing that S is the number of the training documents, T is the number of the testing documents, and N is the number of the feature items, the time complexity of KNN can take O(TSN). So for large-scale training set, KNN will need quite much running time.

IV. The Refined Weighted KNN Algorithm

As discussed in Section 1, traditional KNN takes the assumption that the importance of each feature item in various categories is equal. Obviously, this is not so reasonable. Traditional KNN often introduces inductive biases for in fact each feature item may have different importance and different distribution in different categories. So when the assumption is violated the traditional KNN often has poorer performance. Then the weighted KNN algorithm is proposed which introduces the weight of each feature item in every category. We introduce a weight vector $\vec{w}_j$ for each class C_j, $\vec{w}_j = (w_{j1}, w_{j2}, ..., w_{jn})$ where w_{ji} stands for the weight of feature item t_i in class C_j. Then the modified similarity measurement formula can be derived as follows:

$$sim(\vec{x}, \vec{d}_i, \vec{w}_j) = \frac{\sum_{k=1}^{N} w_{jk} \times x_k \times d_{ik}}{\sqrt{(\sum_{k=1}^{N} x_k^2)(\sum_{k=1}^{N} d_{ik}^2)}} . \qquad (7)$$

And then we should also amend the formula of the class weight of the test document $\vec{x}$ as follows:

$$p(\vec{x}, C_j, \vec{w}_j) = \sum_{\vec{d}_i \in KNN(\vec{x})} sim(\vec{x}, \vec{d}_i, \vec{w}_j) y(\vec{d}_i, C_j) . \qquad (8)$$

Now, how can we get the weight vector $\vec{w}_j$ for each class C_j? Considering the different importance and distribution of each feature item in every class, we get the calculation method of the weight vector based on the idea of variance in mathematics. This idea holds that if one feature item distributes more evenly in one category and less evenly among all categories, then this feature item is more important for this category and the weight value of this feature item for this class is higher. For example, one feature item t_i occurs in the documents of class C_j very frequently, while occurring in documents of other classes rarely; then we can believe that feature item t_i is important for class C_j, and the weight value w_{ij} should be given a high value. Based on the idea of variance, our weight measurement formulas are listed as follows:

$$v_{ji} = \frac{\sum_{\vec{d}_k \in C_j} (d_{ki} - E_{ji})^2}{|C_j|}, \quad i \in [1, n]. \tag{9}$$

Here v_{ji} denotes the variance value of the feature item t_i in class C_j, while d_{ki} is the value of the feature item t_i in document $\vec{d}_k$, $\vec{d}_k \in C_j$. Where E_{ji} stands for the value of the feature item t_i in center vector of class C_j, that is,

$$E_{ji} = \frac{\sum_{\vec{d}_k \in C_j} d_{ki}}{|C_j|}, \quad i \in [1, n]. \tag{10}$$

Where $|C_j|$ is the number of documents of class C_j.

If v_{ji} is smaller it explains that the feature item t_i distributes evenly in class C_j. So the feature item t_i may be important for class C_j. But v_{ji} just stands for the distribution of the feature item t_i in class C_j. Then we also need to know the distribution of the feature item t_i in all classes, thus another weight factor is defined as follows:

$$v_i = \frac{\sum_{j=1}^{|C|} (E_{ji} - E_i)^2}{|C|}, \quad i \in [1, n]. \tag{11}$$

Here v_i is the variance value of the feature item t_i in all classes and $|C|$ is the total number of classes. E_i is the average value of all E_{ji}, that is,

$$E_i = \frac{\sum_{j=1}^{|C|} E_{ji}}{|C|}, \quad i \in [1, n]. \tag{12}$$

If v_i is bigger it shows that the feature item t_i distributes unevenly in all classes. Then the feature item t_i may play an important role in distinguishing different classes.

Now, with the above two weight factors we can give a final weight measurement formula as follows:

$$w_{ji} = \frac{v_i \times 10^{2h}}{(10^{v_{ji}})^h}, \quad i \in [1, n]. \tag{13}$$

Here w_{ji} is the final weight value of the feature item t_i in class C_j. While h is a variable argument and in this paper we get 5 according to the value of the two weight factors v_i and v_{ji}. From this formula, we can derive that when v_{ji} is smaller and v_i is bigger the final weight value w_{ji} is bigger and this is the very weight we want. In other words, the proposed weight measurement conforms to the former idea that if one feature item distributes more evenly in one category and less evenly among all categories, this feature item is more important for this category and the weight value of this feature item for this class is higher.

For the weighted KNN, the running time of the weight measurement needs O(*SN*). So the running time of weighted KNN is O(*SN*)+ O(*TSN*), which is just a little more than that of traditional KNN.

V. Experimental Results

A. The Performance Measure

In our experiment, we use the popular *F1* measure. *F1* measure takes into account both recall and precision to evaluate a text classification system. They are defined as follows[7]:

$$\mathrm{Re}call = \frac{number\ of\ correct\ positive\ predictions}{number\ of\ positive\ examples}, \tag{14}$$

$$\mathrm{Pr}ecision = \frac{number\ of\ correct\ positive\ predictions}{number\ of\ positive\ predictions}, \tag{15}$$

$$F1 = \frac{2 \times \mathrm{Pr}ecision \times \mathrm{Re}call}{\mathrm{Pr}ecision + \mathrm{Re}call}. \tag{16}$$

For evaluating performance average across categories, there are two forms of *F1* average measure: Macro-average and Micro-average of *F1* measure.

Macro-F1 = average of within-category *F1* values. (17)

Micro-F1 = *F1* over categories and documents. (18)

Then in our experiment, we use *Micro-F1* as the performance measure.

B. Experiment Design and Result Analysis

In our experiment, we use TanCorp-12 dataset of TanCorp corpus[6][7]. The dataset contains 12 categories and the total documents amount to 14150. We use three-fold cross-validation in our experiment. Then we randomly select 150 documents from each category amounting to 1800 documents as our experimental dataset. The dataset is split into three parts, then two parts are used for training and remaining third is for testing. So in our experiment the training-testing procedure is conducted for three times and the average of the three performances is used as the final result.

For feature selection method, we employ three different methods that are Document Frequency(DF), Expected Cross Entropy(ECE) and Correlation Coefficient(CC)[8][9]. Among these feature selection methods, DF doesn't perform as well as the other two in most cases. With different feature selection methods, we want to prove that our proposed refined weighted KNN can still perform better than traditional KNN.

In our experiment, we find that if *k* gets 5 the text classifier performs well, so we set *k*=5.

Tables I, П and Ш show the performance of traditional KNN and refined weighted KNN text classifier with three different feature selection methods, that is DF, ECE and CC.

The first column in Table I i.e. DF(#) is the minimum number of documents where one feature item occurring at least once. The second column is the number of feature items and the last two columns are the *Micro-F1* value of traditional KNN and refined weighted KNN text classifiers. The columns in Tables П and Ш are like the same.

TABLE I. THE RESULT OF TWO ALGORITHMS WITH DF FEATURE SELECTION

DF(#)	**Feature Items**	***Micro-F1* value**	
		Traditional KNN	***Weighted KNN***
60	1067	0.553	**0.657**
30	2106	0.615	**0.732**
20	3053	0.645	**0.747**
15	3966	0.653	**0.758**
11	5165	0.667	**0.755**
9	6115	0.675	**0.747**
7	7451	0.673	**0.75**
6	8458	0.68	**0.752**
5	9778	0.68	**0.75**

From Table I, we can see that with the DF feature selection the refined weighted KNN makes a significant improvement on the performance of traditional KNN. With any different number of feature items the refined weighted KNN classifier performs better obviously.

TABLE II. THE RESULT OF TWO ALGORITHMS WITH ECE FEATURE SELECTION

Feature Items	***Micro-F1* value**	
	Traditional KNN	***Weighted KNN***
1000	0.685	**0.763**
2000	0.688	**0.760**
3000	0.690	**0.750**
4000	0.687	**0.752**
5000	0.688	**0.755**
6000	0.698	**0.745**
7000	0.692	**0.753**
8000	0.69	**0.743**
9000	0.678	**0.743**
10000	0.673	**0.75**

From Table П, we can derive that with ECE feature selection the refined weighted KNN also has a better performance of text classification than tradtional KNN.

TABLE III. THE RESULT OF TWO ALGORITHMS WITH CC FEATURE SELECTION

Feature Items	***Micro-F1* value**	
	Traditional KNN	***Weighted KNN***
1000	0.748	**0.757**
2000	0.720	**0.748**
3000	0.715	**0.773**
4000	0.715	**0.762**
5000	0.708	**0.753**
6000	0.715	**0.752**
7000	0.697	**0.753**
8000	0.705	**0.758**
9000	0.692	**0.757**
10000	0.693	**0.755**

From Table Ш, we can also find that with CC feature selection our proposed weighted KNN is a better text classifier than traditional KNN.

Experimental results show that even with different number of feature items and different feature selection methods the new refined weighted KNN text classifier performs better then traditional KNN in most cases. So we can believe that each feature item has different importance in different categories and our proposed refined weighted KNN is a more effective text classifier than traditional KNN.

VI. CONCLUSION

In this paper, we proposed a new refined weighted KNN algorithm for text classification. We get the weight measurement based on the idea of variance which holds that if one feature item distributes more evenly in one category and less evenly in other classes this feature item is more important for this category and the weight value of this feature item in this category is higher.

Although the refined weighted KNN algorithm needs a little more running time than traditional KNN, experimental results show that the refined weighted KNN could make a significant improvement on the performance of traditional KNN classifier.

ACKNOWLEDGMENT

The work was supported by a grant from the State Key Laboratory of Computer Science of the Institute of Software of Chinese Academy Sciences and by a grant from Scientific Research Foundation for Returned Scholars, Ministry of Education of China.

REFERENCES

[1] Sebastiani and Fabrizio, "Machine learning in automated text categorization," ACM Computing Surveys, Vol.34, No.1, 2002, pp.1–47.

[2] Zhen-zhou Chen, Lei Li, and Zheng-an Yao, “Feature-Weighted K-Nearest Neighbor Algorithm with SVM,” Acta Scientiarum Naturalium Universitatis Sunyatseni, Vol.44, No.1, 2005, pp.17-20.

[3] Xiao-dong Qian and Zheng-ou Wang, “Text Categorization Method Based on Improved KNN,” Information Science, Vol.23, No.4, 2005, pp.550-554.

[4] Song-bo Tan, “An effective refinement strategy for KNN text classifier,” Expert Systems with Applications, Vol.30, 2006, pp.290-298.

[5] Jiawei Han and Micheline Kamber, Data Ming:Concepts and Techniques, 2nd ed, Beijing:China Machine Press, 2007,pp.401-407.

[6] Song-bo Tan and Yue-fen Wang, Chinese corpus of text categorization-TanCorpV1.0, http://www.searchforum.org.cn/tansongbo/corpus.htm.

[7] Song-bo Tan, Xue-qi Cheng, Moustafa M. Ghanem, Bin Wang, and Hong-bo Xu, “A Novel Refinement Approach for Text Categorization,” Proceedings of the 14th ACM international conference on Information and knowledge management, New York, NY, USA: ACM, 2005, pp.469-476.

[8] Yi-ming Yang and Pedersen J O, “A comparative study on feature selection in text categorization,” Proceedings of the 14th International Conference on Machine Learning, 1997, pp.412-420.

[9] Hwee Tou Ng, Wei Boon Goh, and Kok Leong Low, "Feature selection, perceptron learning, and a usability case study for text categorization," Proceedings of the 20th ACM International Conference on Research and Development in Information Retrieval, 1997, pp.67-73.