**Text Classification based on TF-IDF and Cosine Similarity**

Abstract

Text classification is the most fundamental task for text mining. Text classification performance determines the accuracy of text processing applications. Many text classification algorithms have been presented in previous literatures, such as KNN, Naïve Bayes, Support Vector Machine, and some improved algorithms based on these three. However, these algorithms perform unstably in the current big data environment. The performance of these algorithms depends on the data set and they do not have self-learning functions. This paper proposes an effective approach for text classification. The three points of the approach are extracting the keywords of category (KWC) of labeled texts based on the TF-IDF approach, classifying unlabeled text by the relevancy of category and unlabeled text, improving the performance of the approach by updating the KWC in the process of classification. Experiments compare the three of most popular methods (KNN, Naïve Bayes, Support Vector Machine) and the new approach on three benchmark corpuses (bbc,Reuters-21578,20 Newsgroups Data). Empirical results show that the new approach can improve the accuracy of text classification to reach 90% and even up to 95% when the data volume is large enough. Although for big data processing, the training time for this method may increase, the classification time declines to one tenth of that of other algorithms. It is valuable in practical application that the KWC in the new approach can be updated automatically to ensure the efficiency of classification.

Index Terms

Text classification; KNN text classification; TF-IDF; Cosine Similarity

1 INTRODUCTION

Text classification is the most important part of grouping documents into different categories or classes. With the advent of the era of big data, the requirements for reliable automatic text classification have increased. Text classification techniques are used, for example, to classify news stories, to find interesting information on the web, and to guide a user search through hypertext.

Three kinds of algorithms have frequently been used for text categorization. They are KNN, SVM, and Naïve Bayesian. These three methods are commonly used algorithms in the datamining field and often applied to text categorization. These three methods have obvious advantages in their respective application fields. People present some improved algorithm based on these three methods with some special classification requirements.

The first kind of algorithm is KNN. The k-nearest neighbor approach has been well used in the literature and shown to be a powerful nonparametric technique for density estimation and classification [19]. The KNN algorithm is easy to understand. This algorithm determines the category of a text according to the statistics about the k-nearest neighbors. The KNN method is easy to understand. It is also very easy to implement, very accurate, and not sensitive to outliers. Many researchers have proposed improved methods based on the KNN method, such as an improved k-nearest-neighbor algorithm for text categorization [2], the use of the KNN model for automatic text categorization [3], a simple KNN algorithm for text categorization [4], a two-level hierarchical combination method for text classification [5], text categorization based on k-nearest neighbor approach for website classification [6], FSKNN multi-label text categorization based on fuzzy similarity and k-nearest neighbors [7], and text categorization using weight-adjusted k-nearest neighbor classification [8]. However, their computational complexity is high, and they are not suitable for large-scale data classification problems. KNN algorithms do not have the mechanisms required to automatically enhance the efficiency of the classification process during the classification process. Therefore, the KNN algorithm cannot be applied in the field of big data.

The second kind of algorithm is SVM. SVMs are learning methods that were introduced by V. Vapnik [10][9]. They are well-founded in terms of computational learning theory and very open to theoretical understanding and analysis. Many researchers have proposed improved methods based on the SVM method, such as transductive inference for text classification using support vector machines [11], support vector machine active learning with applications in text classification [12], dimension reduction in text classification with SVMs [13], feature selection in SVM text categorization [14], text classification using SVM with a mixture of kernels [15], text categorization with SVMs learning with many relevant features [16], learning to classify text using SVMs methods, theories, and algorithms [17], and combining supervised term-weighting metrics for SVM text classification with extended term representation [18]. This method is often used for small sample classification and nonlinear and high-dimensional data sets. However, now is the age of big data, and most people are not concerned with acquiring data but improving classification performance. SVM algorithms do not have the mechanism required to automatically enhance the efficiency of the classification in the classification process. Thus, the SVM classification algorithm is not suitable for large amounts of text data.

The Naïve Bayes classier makes up the other group of algorithms. Naïve Bayes has been studied extensively since the 1950s. It was introduced under a different name into the text retrieval community in the early 1960s and remains a popular (baseline) method for text categorization [20]. Based on the bag-of-words representation, this algorithm uses probabilistic models to estimate the likelihood that a given document is in a particular class. They use this probability estimate for decision making. Many researchers have proposed improved methods based on the Naïve Bayes method, such as some effective techniques for Naïve Bayes text classification [21], Naïve Bayes for text classification with unbalanced classes [22], adapting the Naïve Bayes tree for text classification [J], knowledge and information systems [23], tackling the poor assumptions of Naïve Bayes text classifiers [24], sentiment analysis of blogs by combining lexical knowledge with text classification [25], and a comparison of event models for Naïve Bayes text classification [26]. Naïve Bayes classification is a simple and effective classification method based on statistical learning theory, but it cannot handle changes based on the characteristics of the combination results. Naïve Bayes algorithms do not have the mechanisms required to automatically enhance the efficiency of the classification during the classification process, and their classification performance is lower than that of other algorithms. Thus, the Naïve Bayes classification algorithm is unable to guarantee high efficiency classification for large amounts of textual data.

In this paper, we propose an effective approach for text classification. It is divided into three stages, namely extracting the keywords of category (KWCs) of labeled texts based on an TF-IDF approach, classifying unlabeled text by the relevancy of the category and unlabeled text, and improving the performance of the approach by updating the KWC in the process of classification.

This paper is divided into four chapters. Chapter One is the introduction. In Chapter One, we provide a description the research status of text categorization and point out the problems with existing text classification. In Chapter Two, we describe a new text categorization method based on TF-IDF and give a detailed description of the process of the parameter optimization experiments and self-updating mechanism of this method. In Chapter Three, we show the results of experiments that compare the four most popular methods (KNN, Naïve Bayes, SVM, and TSVM) and the new approach on three benchmark corpuses (BBC, The Reuters Transcribed Subset Data Set, 20 Newsgroups Data and Crawled 6586 news webpages from BBC News in 2016). In Chapter Four, we summarize the results of the whole paper.

2 A New Approach for Text Classification Based on TF-IDF and Cosine Similarity

2.1 TF-IDF

A new approach for text classification is proposed based on the well-known TF-IDF term weights [27][28][29][30]. TF-IDF is a weighting factor used in information retrieval and text mining. It represents a numerical statistic that shows how important a word is in a document in a collection or corpus. Term frequency (TF) is a value that represents how often a particular word appears in a document. The more frequent the word is, the more important it is in the document. For example, “debug” could be the keyword in a software magazine because it does not appear in normal documents but frequently appears in software documents. However, if the word appears too frequently in the corpus, this means that the word is common in the corpus. For a certain document, it is not so valuable. “The” cannot be a keyword in any document because it appears frequently in every document. This kind of word often appears in various texts. We call them “stop-words,” which include “the,” “that,” “a,” “an,” “those,” and so on. So we use inverse document frequency (IDF) to evaluate the value of a word in a document. Document frequency (DF) can be obtained by dividing the total number of documents by the number of documents containing the term. IDF is the inverse value of DF.

TF-IDF is the product of TF and IDF. A high TF-IDF weight indicates a high term frequency and a low document frequency of the term in the whole collection of documents. Hence, the weights tend to filter out common terms. TF-IDF can be successfully used for stop-word filtering in various subject fields, including text summarization and classification [31].

The specific process of computing TF-IDF values is as follows:

1) Computing TF.

After obtaining an article, word segmentation is the first step. In this study, we use an English corpus, so we remove the special characters, and then we split the text by spaces to segmentation. Then we have all of the words in this text. In order to improve calculation efficiency, we remove the stop-words according to the stop-words list. We calculate the occurrences of the other words and divided them by the total number of non-stop-words.

(Ti,Di) means that the TF of term Ti is Di. Get the following results:

2) Computing IDF.

We use the number of documents that include a particular word and the total number of documents in the corpus to compute the IDF of the word.

If a word is common, the denominator is large. Inverse document frequency is closer to zero. (Ti,Fi) means that the TF of term Ti is Di. Get the following results:

3) Computing TF-IDF.

TF measures the frequencies of words in a text. IDF is a measure of the frequency of documents in a corpus that contain a particular word. TF-IDF is influenced by both TF and IDF.

TF-IDF is often used to extract the keywords of a text. In this paper, we use TF-IDF to not only to extract the keywords of a text but also extract the KWCs. The whole process is very similar. KWC can be described as a feature of category information, so we can use the KWC to represent a category. We extract the keywords of all documents in a certain category and summarize the KWCs. So we can get the KWC. In section 2.3, we use the relevancy of KWC and the keywords of unlabeled texts to classify the unlabeled texts.

2.2 Cosine Similarity

Cosine similarity is a measure of similarity between two vectors of n dimensions by finding the cosine of the angle between them. It is often used to compare documents in text mining. The cosine of 0° is 1, and it is less than 1 for any other angles. It is, thus, a judgment of orientation and not magnitude; two vectors with the same orientation have a cosine similarity of 1, two vectors at 90° have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, regardless of magnitude. Cosine similarity is particularly used in positive space, where the outcome is neatly bound in [0,1].

Note that these bounds apply to any number of dimensions, and cosine similarity is most commonly used in high-dimensional positive spaces. For example, in information retrieval and text mining, each term is notionally assigned a different dimension and a document is characterized by a vector where the value of each dimension corresponds to the number of times that term appears in the document. Cosine similarity then gives a useful measure of how similar two documents are likely to be in terms of their subject matter.[32]

The cosine of two vectors can be derived by using the Euclidean dot product formula:

Given two vectors of attributes, A and B, the cosine similarity, cos(θ), is represented using a dot product and magnitude.

In the formula, and are components of vector A and B, respectively. The resulting similarity ranges from [-1,1],-1, meaning exactly opposite, to 1, meaning exactly the same, with 0 indicating orthogonality (decorrelation) and in-between values indicating intermediate similarity or dissimilarity. In the case of information retrieval, the cosine similarity of two documents will range from 0 to 1, since the term frequencies (TF-IDF weights) cannot be negative. The angle between two term frequency vectors cannot be greater than 90°.

Cosine similarity is often used to measure the similarity between two texts. In this paper, we use cosine similarity to measure not only the similarity of texts but also the similarity between the text and the category. The whole process is very similar. The numbers of text keywords and category keywords are always different, but for cosine similarity calculation, similarity is not affected. In section 2.2, we get the KWC to represent every category. We get the KWC. Then we calculate the similarity of the text and the KWC in each category. We classify the text into the category with the highest cosine similarity.

2.3 A New Approach for Text Classification Based on TF-IDF and Cosine Similarity

This section presents a new approach to text classification based on TF-IDF and cosine similarity. The three phases of the approach are extracting the KWCs of labeled texts based on the TF-IDF approach, classifying unlabeled text by the relevancy of the category and unlabeled text, and improving the performance of the approach by updating the KWC in the process of classification. The overall architecture diagram of the approach is as follows:

There are three phases in the new approach: 1) training phase; 2) testing phase; and 3) automatic updating phase. Each phase contains some submodules.

The first phase is the training phase. The objective of this phase is to get the KWCs from labeled texts based on the TF-IDF approach. First, we put the labeled texts for text preprocessing work. In this step, we read all the labeled texts in a category. Then we remove all the special characters in the text, such as \n,,,(.),. $, %, etc. In this study, the data set is all in English, so we use spaces to split the whole text. Then we filter the words in the stop-words list and ensure that the remaining words are the KWCs. Then we do the same work on all categories. We can get the KWCs of every category, but under the background of big data, there will be a lot of training texts. Meanwhile, there will be lots of keywords in every category. So we weighted all keywords using the TF-IDF method for subsequent sorting and screening of keywords. We describe the method in detail in section 2.4.

Here we algorithms also consider two special classifications:

The first is the noise problem. It means there are some incorrectly labeled texts in the training data set. Many classification algorithms are very sensitive to noise. This means that if there is some noise in the training set, it will greatly affect the efficiency of the classification algorithms. The new classification algorithms have a very strong ability to inhibit noise. From the algorithm design mechanism, we will extract the KWCs from a number of known texts. Noise text is very little comparing with a lot of testing text, so noise does not impact the extraction of KWCs. Even noise is extracted keyword, the TF-IDF weights of the noise are very small. Therefore, the noise problem does not affect classification efficiency.

The second problem is the generic classification problem. It means the TF-IDF values of each category keyword are not too prominent. Many classification algorithms cannot deal with such problems or have a great influence on classification efficiency, such as KNN and Naïve Bayes. The new classification algorithms can solve this problem. From the algorithm design mechanism, we will extract the KWCs from a number of known texts and sort the KWCs to filter keywords in section 2.4. Therefore, we only care about the value of a specific keyword ranking rather than weight. Thus, the generic classification problem does not affect classification efficiency.

to the TF-IDF values. It first reads the root path of training text (step 1 to step 3). The directory structure of the training set is as follows, based on the BBC dataset:

Each directory represents a category. Then it reads every file, cleans all content, and merges all text content (step 4 to step 10). Then it calculates the TF-IDF of every word (step 11 to step 13), places the keywords in order for subsequent sorting and screening of keywords (step 14 to step 16).

The second phase is the testing phase. The objective of this phase is to get the keywords of every unlabeled text based on the TF-IDF approach. First, we prepare the unlabeled texts for text preprocessing work. In this step, we read all the unlabeled texts. Then we remove all the special characters in the text, such as \n,,,(.),. $, %, etc. In this study, the data set is all in English, so we use spaces to split the whole text. Then we filter the words according to the stop-words list. Ensure that the remaining words are the keywords of the unlabeled text. Then we do the same work on all texts. The number of keywords of text are few relative to the number of KWCs. Therefore, we weighted all keywords using the TF-IDF method for the subsequent sorting and screening of keywords. We describe the method in detail in section 2.5.

The algorithm test module is applied to get the keywords for every testing text, and then the keywords are sorted according to the TF-IDF values. It first reads the root path of the test text (step 1 to step 9). The directory structure of the unlabeled set is the same as in the training directory. Each test text has a real category (step 3). Then it reads every file and cleans all content (step 2 to step 7). Then it calculates the TF-IDF of every word (step 10 to step 11) and places the keywords in order for the subsequent sorting and screening of keywords (step 13 to step 15).

The last step of the second phase is the classification of texts by cosine similarity.

Algorithm classification by CS is applied to classify test texts by cosine similarity. It first determines the keywords of the test text (step 1). Then it calculates the level of similarity with every category (step 2 to step 6). The result is the max similarity of category (step 7).

The last phase is the automatic updating phase. The objective of this phase is to update the KWC automatically in the process of classification in order to maintain the level of performance. First, we set the threshold function. When the category similarity of the test text reaches the threshold condition, we append the test text to the training corpus and repeat the training phase to update the KWC. It will keep the performance in the process of continuously classification. We describe the threshold function in section 2.6.

2.4 Category Parameter Discussion of Method

In the section 2.3, we propose a new approach for text classification. The first phase is the training phase. The objective of this phase is to get the KWCs from labeled texts based on the TF-IDF approach. After data processing, we can get all the KWCs and TF-IDF values. The KWC of the training set is as follows, based on the BBC dataset (For more information, see BBC corpus 3.1). It is listed for each category in the top 20 keywords:

It is obvious that the relationship between these keywords and categories is very close. The TF-IDF weight determines the degree of relation. The bigger the value is, the closer the relevance is. We can see each category reserved too much keywords in the big data background. This will affect our classification efficiency, and when each category of keywords to reach a certain threshold, the increase of the number of keyword affect the accuracy of classification barely. For instance, the BBC data sets.

In the above experiment, we selected all of the test text keywords, and the percentage of KWC increased gradually from 0% to 1%. The growth rate is very obvious between 0% and 0.16%, but is very gentle above 0.16%. The same is the result of on Reuters-21578 data set and 20 Newsgroups Data set. Therefore, we chose part of KWC instead of all. This reduced our classifier accuracy by 6%, but it saved 90% of the required time. In our experiments, the time of using all keywords is between 100 seconds and 200 seconds, but the time of using 0.16% of the keywords is most 20 seconds. If only care about the precision of classification, application system can constantly improve the proportion of KWC. Taking overall performance into account, we selected just a proportion of keywords for classification.

2.5 Text Parameter Discussion of Method

In section 2.3, we propose a new approach for text classification. The second phase is the testing phase. The objective of this phase is to get the keywords for every unlabeled text and classify the text. We got some test texts from the BBC dataset. It is listed for five texts in the top 20 keywords. First of all, we can clearly see that the text keywords are not in order of magnitude with KWC. In the above test text, the maximum number of keywords is 148, and the minimum number of KWCs is 10,217. Therefore, according to our principle of classification, we speculate that the higher the number of text keywords is, the easier it is to classify the text. For instance the BBC data sets.

In the above experiment, we select all KWCs and 0.16% of KWCs. The percentage of testing text is increasing gradually from 0% to 100%. The growth rate is very obvious between 0% and 32% but is fluctuant above 32%. The same is the result of on Reuters-21578 data set and 20 Newsgroups Data set. By comparing the red and blue line, we find that the two lines of the same general trend, and almost merge above 32%. When the number of KWCs is larger, we can confirm the category of the test text by less text keywords. However, it takes plenty of time to calculate all KWCs. Hence, in order to ensure the overall classification performance of approach, we more likely to choose selecting fewer KWC and more testing text keywords.

2.6 Automatic Updating Discussion of Method

Through the descriptions in sections 2.3 to 2.5, we can get a new and better classification model for relative performance. However, the classifier performance cannot be maintained because the KWCs will vary in each era. We crawl 6,586 webpages from BBC News in 2016 and classify them using the new approach. Performance cannot keep, even decline. After investigation, we found that the BBC data set consists of documents from the BBC News website corresponding to stories in five topical areas from 2004 to 2005. The 6,586 webpages we crawled are news reports from 2016. Some words are not the same KWCs from 2004 to 2005, but they are now. Some words are the KWCs from 2004 to 2005, but they are no longer KWCs. For example, AlphaGo is a KWC of the current tech category, but it does not appear any text between 2004 and 2005. Michael Jordan was a KWC in the sports category in 2004 to 2005, but it rarely appears now. We call that cold start. It is a potential problem in computer-based information systems that involve a degree of automated data modeling. Common classification algorithms cannot avoid cold start problems, and the only solution is to re-collect the data to train the model. Therefore, in order to ensure the performance of the new classification approach, there must be a mechanism to ensure the KWCs update automatically.

In this study, we set the threshold function to ensure the performance of the new classification approach. When the category similarity of testing text reaches the threshold condition, we append the testing text to the training corpus and repeated the training phase to update the KWC. By using this this mechanism, we can ensure the performance of the classifier.

Algorithm classification by CS is applied to classify test texts by cosine similarity. Adding step 8-9 for KWC update operations. If the max similarity reaches the threshold function (step 8), it will repeat the training phase to update the KWC automatically. This phase can guarantee the performance of the classifier.

3 EXPERIMENTAL RESULTS

3.1 Dataset Descriptions

**BBC dataset**: We make use of the BBC corpra that have been previously used in document classification and clustering tasks and extract the keywords based on the TF-IDF algorithm. The original BBC corpus contains a total of 2,225 documents with 5 annotated topic labels from the BBC News website corresponding to stories in five topical areas from 2004 to 2005. These five topics are business, entertainment, politics, sports, and tech. Detailed corpus data follows.

**20 Newsgroups:** The 20 Newsgroups data set, which was collected by Ken Lang, consists of 20,017 articles divided almost evenly among 20 different UseNet discussion groups. The task is to classify an article into the one newsgroup (of twenty) to which it was posted. When words from a stop list of common short words are removed, there are 62,258 unique words that occur more than once; other feature selection is not used. The word counts of each document are scaled such that each document has a constant length with potentially fractional word counts. The 20 Newsgroups collection has become a popular data set for experiments in text applications of machine learning techniques, such as text classification and text clustering. Detailed corpus data follows.

**Reuters-21578:** This is a very often used test set for text categorization tasks. It contains 21,578 Reuters News documents from 1987 [1][34]. They were labeled manually by Reuters personnel. Labels belong to 5 different category classes, including people, places, and topics. The total number of categories is 672, but many of them occur very rarely. Some documents belong to many different categories, others belong to only one category, and some have no category. Over the past decade, there have been many efforts to clean the database up and improve it for use in scientific research. In this paper, we use a Reuters transcribed subset data set. This dataset is created by reading out 200 files from the 10 largest Reuters classes and using an automatic speech recognition system to create corresponding transcriptions. Researchers often use this dataset for classification experiments instead of the original Reuters dataset. Detailed corpus data follows.

(http://archive.ics.uci.edu/ml/datasets/Reuters+Transcribed+Subset)

3.2 Performance Measures

For text classification algorithm evaluation, we often use the following three indicators: accuracy, recall, and F1 value.

Accuracy is defined as the following: For a given set of test data, the ratio of sample of classifier classified correctly and the total number of samples. Accuracy is also used as a statistical measure of how well a classifier testing correctly identifies. The accuracy is the proportion of true results (both true positives and true negatives) among the total number of cases examined. An accuracy of 100% means that the measured values are exactly the same as the given values. On the contrary, precision or positive predictive value is defined as the proportion of the true positives against all the positive results.

Recall in information retrieval is the fraction of the documents that are relevant to the query that are successfully retrieved. In classification, recall is called sensitivity. Therefore, it can be looked at as the probability that a relevant document is retrieved by the query.

In the statistical analysis of classification, the F1 score (also F-score or F-measure) is a measure of a test’s accuracy. It considers both the precision, p, and the recall, r, of the test to compute the score; p is the number of correct positive results divided by the number of all positive results, and r is the number of correct positive results divided by the number of positive results that should have been returned. The F1 score can be interpreted as a weighted average of precision and recall. An F1 score reaches its best value at 1 and its worst value at 0. The F-measure or balanced F-score (F1 score) is the harmonic mean of precision and recall. The traditional F-measure or balanced F-score (F1 score) is the harmonic mean of precision and recall.

Hence, this paper chose three of the most common evaluation indexes (precision, recall, and F1) to compare four methods (k-NN, SVM, Bayes) on the three common datasets (BBC dataset, 20 Newsgroups, Reuters-21578).

3.3 Comparison to Related Work

The experiments compare the three of most popular methods (KNN, Naïve Bayes, Support Vector Machine) and the new approach on three benchmark corpuses (BBC, Reuters Transcribed Subset, 20 Newsgroups Data). For these three data sets, we randomly selected 80% as a training set and 20% as a test set. Each category’s specific training set and testing set are as follows.

The BBC dataset distribution is as follows.

The 20 Newsgroups dataset distribution is as follows.

The Reuters Transcribed Subset dataset distribution is as follows.

In these three datasets, the largest number of texts is the 20, in the news groups dataset, and the smallest number of texts is in the Reuters Transcribed Subset dataset. In order to avoid the influence of the size of the dataset for the classification performance, we adopt the method of cross-validation. Repeat 10 times on each set of experiments, 80% were randomly selected as training set, the rest is the test set. Then the results are obtained by average processing. In order to comprehensive precision and recall, we select F1 value as a comprehensive performance index. The result is as follows.

Through the experiments, we found that the performance of the new approach on the three datasets are well. Specific performance will be influenced by datasets and categories. The greater the number of texts in the dataset is, the better the performance of classification is. The clearer the categories in the dataset are, the better the performance of the classification will be.

In order to test the general performance of this classification approach, we crawl 6586 news webpages from BBC News in 2016 (http://pan.baidu.com/s/1qXNrlTy), artificial classification to them by the category of BBC. In the new experiments, the performance of five kinds of methods are lower than the existing data sets. The performance of the new approach also declines but is relatively stable because the new approach has an automatic KWC update mechanism to maintain the same classification performance for a new text set.

4 CONCLUSION

This paper proposes an effective approach for text classification based on TF-IDF and cosine similarity. Our experimental results clearly show that the new approach has good performance in the BBC dataset, Reuters-21578 dataset, and 20 Newsgroups data et. Specific performance is influenced by the datasets and categories. The greater the number of text datasets is, the better the performance of classification is. The clearer the categories of the datasets are, the better the performance of the classification will be. This accords with the current requirements for internet information processing.

This new classification approach also has the very good extension. It can automatically update the KWCs in the process of classification to enhance classification efficiency or at least ensure the stability of classification accuracy. This suggests that the new approach is a simple and strongly practical strategy for text classification.