A Distributed Chinese Naive Bayes Classifier Based on Word Embedding

Mengke Feng[1, a], Guoshi Wu[1, b]

[1]Beijing University of Post and Telecommunications, Beijing 100876, China;

[a]fengmengke@163.com, [b]guoshiwu@bupt.edu.cn

Keywords: Naive Bayes, Word Embedding, Distributed Classifier.

Abstract. The Naive Bayes classifier is built on the assumption of conditional independence between the attributes in a given class. The algorithm has been shown to be successful in text classification. But when calculating the conditional probability these methods take two different words as two different feature, no matter how close their meanings are.In this paper we proposed an algorithm to improve the calculation of probability that a word belonging to a class by using its related words base on word embedding and we named this model NBCBWE (Naive Bayes classifier based on word embedding). Word embedding provides a way of applying Deep Learning to solve natural language processing problems. In this way every word can be represented by a vector, and we can get the related word by calculate the similarity of two words. What's more, as the data set grows larger, it can be very time consuming to store and classify text in a single computer. To decrease the consuming time, we parallel Bayes classification algorithm using Map-Reduceto implement the model on Hadoop. Our experiments shows that our model improves the precision in text classification and also processes more efficiently.

Introduction

With the rapid development of the Internet, it's increasingly important to use automated processing to deal with large-scale data resources. Automatic document classification identifies unknown category, through natural language and machine learning methods. The common classification methods are support vector machine (SVM), K Nearest Neighbor (KNN), neural networks, Bayesian (Bayes) and decision tree. Bayesian classification method is based on Bayesian learning methods, although its principle is relatively simple, but its very success in practical applications.

Most of the Bayesian classification method only use the occurrence of the word to calculate the conditional probability. The tendency of the expression of Chinese is turned to be diversified gradually. It means many phrase has different presentation with similar meaning. If we can accumulate the related words or phrases in a certain of weight to represent the conditional probability of a word or a phrase, it will turn the string matching method into meaning matching, and the conditional probability will be more precise. Word embedding provides a way of applying Deep Learning to solve natural language processing problems. In this way every word can be represented by a vector, and we can easily get the related word by calculate the similarity between two vectors. The word2vec software of Tomas Mikolov has gained a lot of traction lately, and provides state-of-the-art word embedding. So we chose word2vec as the tool to train a large-scale corpus and got word embedding.

As we brought in word embedding to calculate the conditional probability of words belonging to a class, the scale of computation increased very much. Map-Reduce is a parallel and distributed computing model that presented by the Google Labs, and it's mainly for massive data processing. In view of this, we design and implement a parallel configuration Bayesian text classifier based on Hadoop.

Naive Bayes Classifier Based on Word Vectors

The overall classifier introduced. Flowchart of the overall classifier shown in Figure 1. The

input data consists of three parts, data for word2vec, training set and testing set. Word2vec is designed to train a large-scale of no-tagging corpus without supervision and get the word vectors. So the data for word2vec is different from other two corpus because of it's scale.

In this paper, the Naive Bayes classifier is designed and tested in Chinese corpus, so each data set should be preprocessed by Chinese Word Segmentation. And for training set and testing set, the method of feature selection based on word frequency differentia and TF-IDF was proposed to improve the quality of feature selection, the speed and accuracy of categorization. After training process, every class is represented by a map which key is the feature of this class and value is the conditional probability of this feature. Finally, we use word vectors to calculate the probability of a document to be classified. The formula of Weighted Naive Bayes and how we brought word embedding into text classification will be explained in the next section.

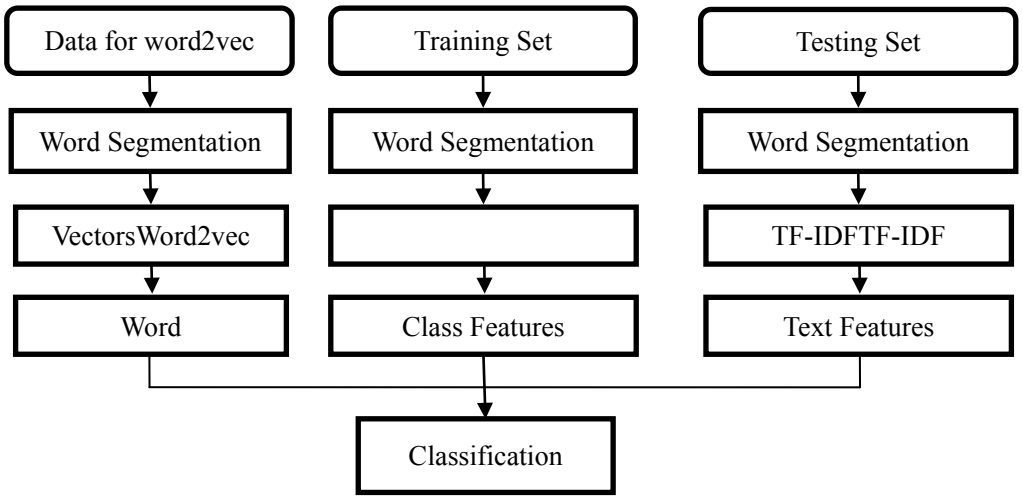


Fig. 1 Flowchart of the classifier

Naive Bayes. Naive Bayes classifier based on a simple assumption: every property values is independent of another property when the target value was given. For text classification, it is assumed that every word w_i and w_j is independent of each other. Based on this assumption, the document d belongs to the class d can be expressed as:

$$P(d \mid Ci) = P\left((w_1, w_2, \ ..., \ w_n) \mid C_i\right) = \prod_{j=1}^{n} P\ \left(w_j \ \mid C_i\right) \qquad (1)$$

There are two ways to calculate $P(w_j|C_i)$, namely document type formulas and word frequency type formulas. In this paper, we choose frequency type. Frequency type consider the word appears in the document frequency, according to the formula (2).

$$P(w_{j,k} \mid C_i) = \frac{1 + TF(w_{j,k}, C_i)}{|V_{C_i}| \ + \sum_{k=1}^{|V_{C_i}|} TF(w_k, C_i)} \qquad (2)$$

$|V_{ci}|$is the total number of characteristic item in class C_i. $TF(w_{j,k},C_i)$ is the occurrence number of characteristic item $w_{j,k}$ in all of the documents belonging to class C_i.

Improve Naive Bayes using word embedding. In this paper, we use word2vec as the tool to train corpus and get word embedding.Word2vec takes a large amounts of training text as input. Word2vec using the distributed representation of word vector and every word feature can be turned into a K-dimensional real-valued vectors after training. We can easily calculate the similarity between words by their corresponding real-valued vectors. There are many methods to calculate the distance between words. In this paper we adopted cosine similarity, according to foumula(3).

$$sim(w_i, w_j) = \cos(\overrightarrow{w_i}, \overrightarrow{w_j}) \quad (3)$$

While researching the features of document to be classified and class that have trained, we have found two phenomenon. One of them is that, there are some words in the document to be classified that have the same meaning with one or more words in the class feature set, but they are not exactly the same words. This phenomenon is especially common in the web pages recently. These new phrase lead to some confusion when classifying text. Word Embedding is a proper way to solve this problem, because every word is training by it's context. For two different phrase, if they are similar with each other in semantics, they will share the similar context. The other phenomenon is that, in the feature set of every class, the occurrence of those words related to this class is more than those unrelated. What's more those words related have gotten higher weight than others. So if we add the probability of those words that is extremely similar with the word w, we can increase the discrimination between those words related to this class and those unrelated. Consider the related words, the conditional probability of a word belonging to a class can be expressed as:

$$p(w_i \mid c) = \sum_{k=1}^{|V|} (f(sim(w_i, w_k)) \times p(w_k \mid c)) \quad (4)$$

f(x) is a function to transform the similarity between words and filter out those words with weak relationship with word w_i. Compared to the calculation of the document type and the type of word frequency, this method of calculation captures more information of the correlation between feature items and categories.

Details of Chinese Naive Bayes Text Classifier. In this section, the details of process of our classifier in a single computer will be expounded. The process is consisted of three parts, how to prepare data for word2vec, how to train Naive Bayes model in Chinese corpus, and how to classify testing text using word embedding.

We chose Chinese Wikipedia as the training data of word2vec. Because the data for word2vec should be both large-scaled and cover many classes. Wikipedia has meet these two requirements very well. We filter out punctuations and replace the line break so that every page is in a single row. We chose the Chinese text segmentation system NLPIR to segment extracted Wikipedia corpus. We chose 200 as the parameter of word2vec, it's the dimension of each word vector. After long-time training, we can finally get the vectors, then we store the vector model as a stream file.

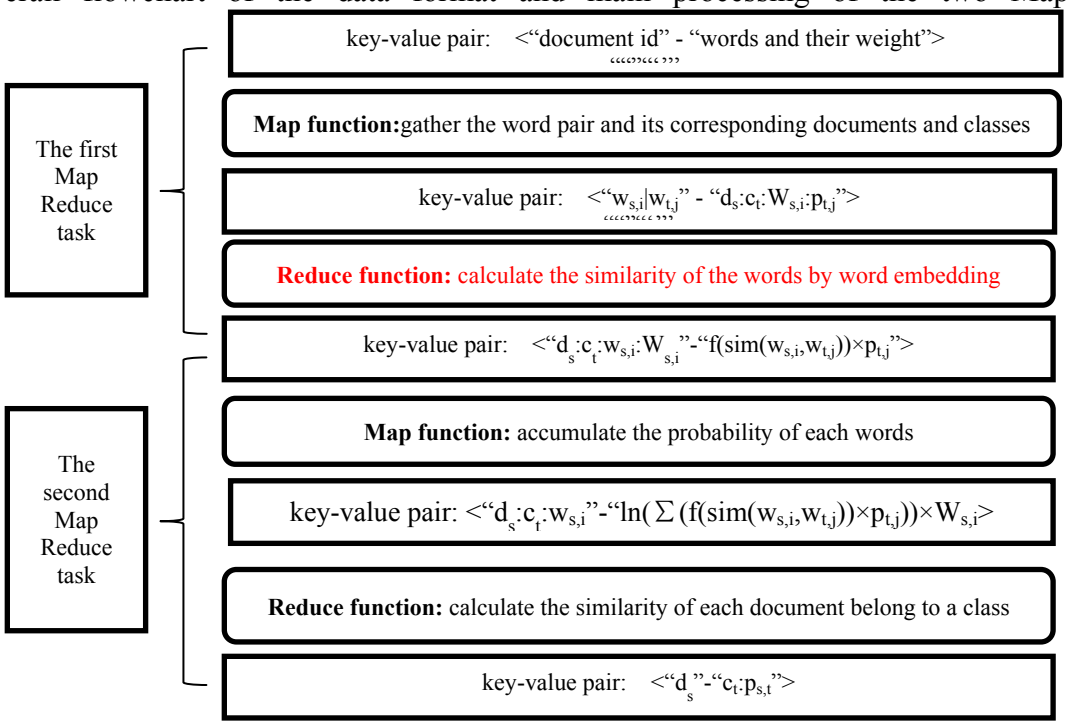
The preprocessing of training set and testing set is similar. First of the process is segmentation as well. Then we filter out the stops words to clean the features. In training process we count the occurrence of all words in training set to calculate the TF-IDF weight. In this paper, we chose word frequency type to calculate the conditional probability. After training process, the Naive Bayes model can be expressed as two Map structure. One is the key-value pair of word and it's TF-IDF weight, the other is the key-value pair of word and it's conditional probability.

The process of traditional Naive Bayes can be simplified as multiplying the conditional probability of each word as formula (1). Compared to formula (1), formula (5) needs much more computation. The conditional probability of word *w* is the sum of its related words, and the similarity between words is cosine similarity between their vectors. But other process of our classifier is same as traditional Naive Bayes. To solve the problem of two much computation, we will introduced a parallel model of Naive Bayes based on Hadoop.

Parallel Naive Bayes using Hadoop. As the increase of training set, it's impossible to store all the documents in a single node when the scale is extremely huge. So we adopted the Map-Reduce programming model to deal with the training phase of large-scale data sets. Map-Reduce is a programming model, which is relevant to processing / generating massive data set. The basic procedure of Map-Reduce is two functions Map and Reduce. The input, output, and intermediate data are all based on <key, value> form. The input of Map function are some set of tuples. Every time when Map function is called, it will generate zero or more intermediate tuple <k2, v2>. The system will collect all the tuple list with the same key tuple, and transfer it to Reduce function to

process. Reduce function is called by every list of different key values at a time, therefore, its input is <k2, list (v2)>.

In this paper, we use the open source project Hadoop of Map-Reduce framework to program. We use two Map-Reduce task to process training procedures and to implement Bayesian text classification algorithm. The documents and categories are characterized as formal features items and weights. For each document, d_s=<$w_{s,1}$:$W_{s,1}$, $w_{s,2}$:$W_{s,2}$, …, $w_{s,2}$:$W_{s,2}$>.And for each class, c_t=< $w_{t,1}$:$p_{t,1}$, $w_{t,2}$:$p_{t,2}$, …, $w_{t,n}$:$p_{t,n}$>. *s* means the id of document and t means the id of class $w_{s,t}$ represents a word in document d_s, and $W_{s,i}$ represents the TF weight of word $w_{s,i}$. $w_{t,j}$ represents a words in class c_t, and $p_{t,j}$ is the conditional probability of word $w_{t,i}$ in class c_t. Figure 2 shows the overall flowchart of the data format and main processing of the two Map-Reduce task.



The first Map-Reduce task splits page by word pair between document and class. So we can avoid repeated computation by calculating the similarity of the same word pair together. In the first process of Map-Reduce , the data form of input in Mapper function is <key, value> type, which key is correspond to the identification of a document, and value is correspond to a string composed of characteristic item and its weight, like "$w_{s,1}$:$W_{s,1}$, $w_{s,2}$:$W_{s,2}$, …, $w_{s,2}$:$W_{s,2}$", there is space between every item.The process of Mapper function is as follow.

for $w_{s,i}$ in d
    for c_t in C
        for $w_{t,j}$ in c_t
            context.write("$w_{s,i}|w_{t,j}$", "d_s:c_t:$W_{s,i}$:$p_{t,j}$ ")

The capital letter *C* represents all class. After this process of Mapper function, the immediate data is a key-value pair, which key is string combined with two items, and the value is composed of document d_s , class c_t and the weight. The process of reduce function in the first Map-Reduce task is as follow.

r := f(sim($w_{s,t}$, $w_{t,j}$))
    if r > α
        for d_s:c_t:$W_{s,i}$:$p_{t,j}$ in list
            context.write("d_s:c_t:$w_{s,i}$:$W_{s,i}$", "r×$p_{t,j}$")

The first Map-Reduce process collected all the records with the same characteristic item, and compute it an once. The task of the second process Map-Reduce job is to calculate the conditional probability of words and the probability of document in a class. The data form of input in Mapper function is <key, value> type, which key is correspond to a string consisted of the identification of a

document d_s, class c_t ,and item in both d_s and c_t ,$w_{s,i}$. The value is correspond to a list composed of t. The process of Mapper function in the second Map-Reduce task is as follow.

$sum_{s,t,i} := 0$
for $Y_{s,t,i}$ in list
    $sum_{s,t,i} := sum_{s,t,i} + Y_{s,t,i}$
$p_{s,t,i} := \ln(sum_{s,t,i}) \times W_{s,i}$
context.write("d_s:c_t", "$p_{s,t,i}$")

After the process of this Mapper function, the immediate data turns into a key-value pair, which key is a string composed of d_s and c_t, and the value is the weight correspond to the key-value pair $p_{s,t,i}$ is the conditional probability of word $w_{s,i}$ in class c_t .The process of the reduce function in the second is as follow.

$sum_{s,t} := 0$
for $sum_{s,t,i}$ in list
    $sum_{s,t} := sum_{s,t} + sum_{s,t,i}$
context.write("d_s","c_t:$sum_{s,t}$")

Up to here, we get $sum_{s,t}$ the probability of d_s belonging to c_t. The class corresponding to the max probability is the one this text should be classified into.

Experiment Result

In this section, we first compare the precision between traditional Naive Bayes and our model. Then we compare the speed-up ratio when the Hadoop cluster consist of 2, 4, 6, 8 nodes. We use NLPIR System to segment and mark words. In preprocessing, we filter the stop words and converted the digit of the uncommon numbers to a fixed token so as to improve the precious. Our models use 200 as the parameter of dimensions of real-number vector to represent a word and use 10-word windows of document to represent the local context in all of the experiments.

Text Classification Evaluations. In order to show that our model performs better in text classification, we choose a corpus named Sougou Corpus and conducted a classification experiment with Naive Bayes and our model. The dataset of Sougou Corpus comes from several news sites from June 2012 - July 2012. The news consists of international, sports, social, entertainment and other 9 classes which is manual classification. We randomly choose 18,000 pages, 2000 pages for each class. We evaluated the result by using methods of recall rate and accuracy. Recall rate and precision are two important evaluating metrics broadly used to evaluate the quality of results in the areas of information retrieval and statistics. Recall rate means the ratio between the number of corresponding documents searched out and all the corresponding documents in the library, while precision is the ratio between the number of corresponding documents searched out and all documents searched out. Accuracy means the ratio between the number of accurate samples and the number of all samples searched out, i.e. accuracy = accurate samples / samples searched out.The result is shown as table 2.

Class	Naive Bayes			NBCBWE		
	Precision	Recall	F1measure	Precision	Recall	F1measure
Economy	77.13%	80.60%	78.83%	78.71%	83.90%	81.22%
IT	79.86%	80.90%	80.38%	83.15%	82.40%	82.77%
Health	82.46%	72.40%	77.10%	83.54%	74.60%	78.82%
Sports	66.92%	96.70%	79.10%	68.46%	96.80%	80.20%
Travel	84.29%	88.50%	86.34%	85.70%	90.50%	88.04%
Knowledge	76.76%	75.30%	76.02%	78.95%	76.90%	77.91%
Work	63.82%	47.10%	54.20%	67.16%	50.10%	57.39%
Culture	81.21%	70.00%	75.19%	83.55%	71.10%	76.82%
Military	74.90%	74.00%	74.45%	78.39%	79.10%	78.75%

Table 1, Precision between Naive Bayes and NBCBWE

We chose 18,000 articles as training set, and 9,000 articles as testing set. As the table 2 shows, 6,855 articles has been classified into right corpus when using traditional Naive Bayes. Our model performs better and 7,054 articles has been classified right. The precision has been improved from 76.17% to 78.38%.

Text Classification Evaluations. In order to test the performance in the Hadoop, we choose a larger data set of 2.8 G, and we run the program in 2, 4, 6 and 8 nodes separately. The relationship between speed and nodes is approximate to linear. As the nodes of computing cluster expand, the speed increase at the same time. At the same time, the effect of classification is also quite good.

Conclusion

In this paper we presented a new text classification model using word embedding and programmed in Hadoop. The advantages of our model is improving the classification precision using related words. The experiment shows our model outperforms Naive Bayes in text classification. Besides it's a parallel Bayesian classifier designed based on Map-Reduce . The classification is capable of large-scale text data automatic classification.

Acknowledgements

This research is supported by the Science and Technology Projects of MOT China (NO. 2014364222100).

References

[1] Christopher G. Atkeson , Andrew W. Moore , Stefan Schaal, Locally Weighted Learning, Artificial Intelligence Review. (1997) 11-73

[2] Nir Friedman , Dan Geiger , Moises Goldszmidt, Bayesian Network Classifiers, Machine Learning. (1997) 131-163

[3] P. Domingos and M. Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. Machine Learning. (1997) 103–130

[4] Zijian Zheng , Geoffrey I. Webb, Lazy Learning of Bayesian Rules, Machine Learning. (2000) 53-84

[5] Dilrukshi, I.; De Zoysa, K.，Twitter news classification: Theoretical and practical comparison of SVM against Naive Bayes algorithms,Advances in ICT for Emerging Regions (ICTer). (2013) 278-278

[6] Baoyi Wang; Shaomin Zhang, A Novel Text Classification Algorithm Based on Naive Bayes and KL-Divergence, Parallel and Distributed Computing, Applications and Technologies. (2005) 913-915

[7] Jie Lin; Jiankun Yu, Weighted Naive Bayes classification algorithm based on particle swarm optimization, Communication Software and Networks (ICCSN). (2011) 444-447

[8] Yuguang Huang; Lei Lij, Naive Bayes classification algorithm based on small sample set, Cloud Computing and Intelligence Systems (CCIS). (2011) 34-39

[9] Guo Qiang, An Effective Algorithm for Improving the Performance of Naive Bayes for Text Classification, Computer Research and Development. (2010) 699-701