J. Cent. South Univ. (2019) 26: 1−12
DOI: https://doi.org/10.1007/s11771-019-3978-x

Springer

Parallel naive Bayes algorithm for large-scale Chinese text classification based on spark

LIU Peng(刘鹏)[1, 2], ZHAO Hui-han(赵慧含)[3], TENG Jia-yu(滕家雨)[4], YANG Yan-yan(仰彦妍)[3], LIU Ya-feng(刘亚峰)[1, 2], ZHU Zong-wei(朱宗卫)[5]

1. Internet of Things Perception Mine Research Centre, China University of Mining and Technology, Xuzhou 221008, China;
2. National and Local Joint Engineering Laboratory of Internet Application Technology on Mine, Xuzhou 221008, China;
3. School of Information and Control Engineering, China University of Mining and Technology, Xuzhou 221116, China;
4. Communication Division, NARI Technology Co., Ltd., Nanjing 211106, China;
5. Suzhou Institute of University of Science and Technology of China, Suzhou 215123, China

Abstract: The sharp increase of the amount of Internet Chinese text data has significantly prolonged the processing time of classification on these data. In order to solve this problem, this paper proposes and implements a parallel naive Bayes algorithm (PNBA) for Chinese text classification based on Spark, a parallel memory computing platform for big data. This algorithm has implemented parallel operation throughout the entire training and prediction process of naive Bayes classifier mainly by adopting the programming model of resilient distributed datasets (RDD). For comparison, a PNBA based on Hadoop is also implemented. The test results show that in the same computing environment and for the same text sets, the Spark PNBA is obviously superior to the Hadoop PNBA in terms of key indicators such as speedup ratio and scalability. Therefore, Spark-based parallel algorithms can better meet the requirement of large-scale Chinese text data mining.

Key words: Chinese text classification; naive Bayes; spark; hadoop; resilient distributed dataset; parallelization

Cite this article as: LIU Peng, ZHAO Hui-han, TENG Jia-yu, YANG Yan-yan, LIU Ya-feng, ZHU Zong-wei. Parallel naive Bayes algorithm for large-scale Chinese text classification based on spark [J]. Journal of Central South University, 2019, 26(1): 1–12. DOI: https://doi.org/10.1007/s11771-019-3978-x.

1 Introduction

Currently, the amount of Internet Chinese text data is showing exponential growth. Among the rapid increasing data, web document is the most common class of data with the widest coverage. The implementation of automatic classification of large-scale web document can improve the situation of messy text information, keep what is valuable and reject what is worthless, so as to reduce query time and improve search quality [1].

Naive Bayes (NB) classification algorithm, which shows good performance in many applications including text classification, is a simple and effective method among many probabilistic classification algorithms. However, when dealing with large-scale Chinese text sets, this algorithm

Foundation item: Project(KC18071) supported by the Application Foundation Research Program of Xuzhou, China; Projects(2017YFC0804401, 2017YFC0804409) supported by the National Key R&D Program of China
Received date: 2017-07-16; **Accepted date:** 2018-03-02
Corresponding author: ZHU Zong-wei, PhD, Assistant Researcher; Tel: +86-18896751046; E-mail: zzw1988@mail.ustc.edu.cn

will experience many problems such as low efficiency in training the classifier and obvious classification errors [2]. Therefore, with the rapid growth of data amount and feature space dimension under the background of big data, the parallelization of traditional Chinese text classification algorithms will significantly improve its operating efficiency. Thus, this will be a valuable research area.

Traditional parallel algorithms such as message passing interface (MPI) [3] and grid computing [4] cannot meet the growing demand for processing large-scale Internet data, owning to their development complexity, poor scalability and other problems. With the maturity of cloud computing [5], MapReduce (MR) has become one of the key technologies drawing the most attention. Hadoop [6], an open source distributed computing model written in Java language, provides application programming interfaces (APIs) with the core of MapReduce and Hadoop distributed file system (HDFS) which enable Hadoop to handle data of one million nodes and ZB magnitude. For example, ZHOU et al [1] and LIU et al [2] improved Naive Bayes text classification using Hadoop MapReduce.

However, more and more researches [7, 8] have proved that Hadoop MapReduce is so simple that it is incapable of handling complicate jobs like real-time response, interactive analysis and iterative processing, etc. Instead, Spark, which is based on memory computing, greatly avoids high-latency materialization of disk, effectively satisfying the time requirement in the aforesaid jobs. Spark is a new generation of parallel processing platform based on memory computing developed by AMPLab in University of California, Berkeley [9]. The core purpose of Spark is to provide a large-scale-oriented, low latency and developer-friendly data process platform. By introducing RDD [10], Spark can distribute data set in memory of various nodes during the whole cluster computing process, thus saving a large number of disk I/O operations and significantly reducing calculating time. JIANG et al [11] introduced the implementation of several machine learning algorithms on Spark platform and compared the running results with Hadoop-based version, which shows that Spark has good memory bandwidth utilization, stable memory access and high frequency input and output requests. REYES-ORTIZ et al [12] made a comparison between Spark and MPI/OpenMP in terms of big data analysis, comparing algorithms of KNN and SVM respectively. LIU et al [13] designed and implemented parallel multinomial naive Bayes multi-class classification, SVM and binary-class classification algorithms and has compared them with Hadoop-based version.

The text analysis based on Spark is one of the research hotspots. YAN et al [14] has implemented SVM microblog classification algorithm based on Spark which possesses better execution speed compared with that based on Hadoop. LIU et al [15], presented by our team, have implemented K-means text clustering algorithm based on Spark which has a better performance in speed-up ratio and expansibility and so forth compared with that based on Hadoop. TOMAS et al [16] has implemented Spark-based application of logistic regression for multi-class text classification, and experimental results have shown that applied multi-class classification method for Amazon product-review data has higher classification accuracy.

Yet, there is still no research on PNBA for large-scale text classification, so we strive to do some useful work in this area.

The rest of the paper is arranged as follows. Section 2 introduces the overall process of text classification and the principle of NB classification algorithm. Section 3 designs and implements PNBA based on Hadoop and Spark. Section 4 compares two parallel algorithms based on Hadoop and Spark respectively on main performance indicators, such as accuracy of classification, speedup ratio, scalability, operation time, etc. Section 5 goes to a conclusion.

2 Overall process of text classification

As shown in Figure 1, the overall process of text classification mainly consists of three steps. The first step is corpus preprocessing which includes word segmentation, stop words removal, word frequency statistics, feature selection, etc. The second step is text representation which transforms the text into a form that can be recognized and handled by computers. And the last step is text classification based on some kinds of machine learning algorithm. This paper is focused on improving the efficiency of large-scale text classification algorithm, and before this, the main

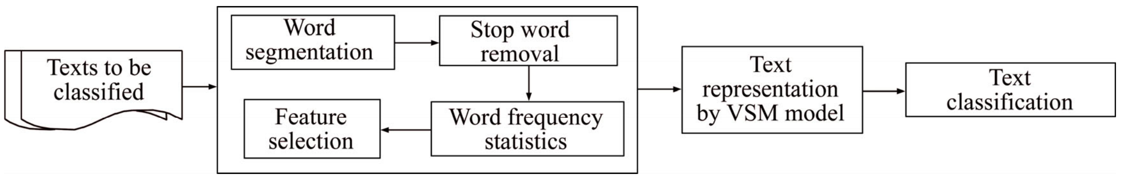

Figure 1 Overall process of text classification (VSM, vector space model [17])

content of Corpus preprocessing and text representation will be introduced.

2.1 Corpus preprocessing

Text data is some kinds of unstructured data, it cannot be directly processed using data mining algorithms. Hence, it is necessary to preprocess the corpus, including work such as word segmentation, Stop word removal, word frequency statistics, feature selection, etc. The quality of corpus preprocessing has a great effect on text classification accuracy. Toolkit for Chinese corpus preprocessing adopted in this paper is NLPIR (nature language processing & information retrieval) Chinese word segmentation system, one of the mainstream products receiving high praise within the industry. It is researched and developed by Institute of Computing Technology, Chinese Academy of Sciences [18].

The first step of Chinese corpus preprocessing is word segmentation [14]. Unlike English text, the Chinese words do not contain continuous space between words. Therefore, word segmentation is required first in Chinese text classification, namely, segmenting text according to the meaning of words and separating words by space.

The segmented word sets still cannot represent text as feature sets, because they contain a large number of function words, such as “the”, “that” and “those” in English or “的”, “吗” and “在” in Chinese. These words are usually called stop words. It is required to establish a stop word list so that stop words in text can be filtered out according to the list. In this way, not only can storage space be saved, but also the dimension of feature space can be reduced and processing can be accelerated.

The basic principle of word frequency statistics is to count how many times a given word appears in a given document. The method which uses the number of a given word to determine the probability of this word becoming a feature reflects the correlation between document and feature words. Feature selection is to select a feature subset from all features. It is aimed at reducing the dimension of vector space and computational complexity through removing as many features which cannot reflect contents of document as possible among the original feature sets.

2.2 Text representation

Text representation is to express text as a form that can be recognized and handled by a computer. In this paper, the Chinese text is represented by VSM, with each dimension of vector composed of feature item and its weight. The weight of feature item is calculated with term (word) frequency inverse document frequency (TFIDF) [19, 20] as follows.

$$w(t,d)=\frac{tf(t,d)\cdot \lg\left(\frac{N}{n_i}+0.01\right)}{\sqrt{\sum_{t_i\in d}\left[tf(t_i,d)\cdot \lg\left(\frac{N}{n_i}+0.01\right)\right]^2}}$$

where $w(t_i,d)$ denotes the weight of feature item t_i in document d; $tf(t_i,d)$ denotes the word frequency of feature item t_i in document d; N denotes the total number of training documents; n_i denotes the number of documents in which feature item t_i appears; and the denominator is the normalization factor. The vectorization of document d_j is represented as $d_j=(t_{j1}:w_{j1}, t_{j2}:w_{j2}, \cdots, t_{ji}:w_{ji}, \cdots, t_{jm}:w_{jm},)$, where t_{ji} represents the ith feature item of the jth document; w_{ji} represents the weight of t_{ji}; m represents the number of feature item in vector. In this paper, Chinese and English documents are both represented as vectors by Mahout [21], while the related parameters are set as follows. –wt tfidf: using TFIDF to calculate word weight; -lnorm: using logarithm to normalize vectors.

2.3 Text classification

The text classification is an important part of text mining. It refers to process of determining a classification for each text in the document collection according to the predefined classification.

Text classification is a typical supervised machine learning method. The frequently-used text classification algorithm include naive Bayes, KNN (K-nearest neighbor), SVM (support vector machines), Rocchio, etc [22].

The classification process is generally divided into two phases, namely, training phase and prediction phase. Training phase is mainly to conduct training on the training text set to generate classification model, while prediction phase first tests the accuracy of the classification model and then makes classification on the prediction text set if the accuracy meets the requirement.

3 Naive Bayes text classification algorithm

As a probability statistical classification algorithm based on Bayesian equation and feature conditional independence assumptions, naive Bayes classification has a relatively simple principle, that is, for a prediction item, first calculating each probability that this item belongs to each class and then selecting the most probable one as its classification result.

The mathematical definition of naive Bayes is as follows:

1) Let $x=\{a_1, a_2, \cdots, a_m\}$ be an item to be classified, and a_i is a feature of $\boldsymbol{x}$.

2) Let there be n classes of training samples, and they are denoted as $C=\{y_1, y_2, \cdots, y_n\}$.

3) Calculate each probability that $\boldsymbol{x}$ belongs to each class: $P(y_1|x), P(y_2|x), \cdots, P(y_n|x)$.

4) If $P(y_k|x)=\max\{P(y_1|x), P(y_2|x), \cdots, P(y_n|x)\}$, the class of $\boldsymbol{x}$ is y_k.

The critical phase is to calculate each conditional probability in step 3, where the following method is adopted (training + prediction):

1) Obtain the training samples with the known classification results.

2) Count the class conditional probabilities with each feature, namely, $P(a_1|y_1)$, $P(a_2|y_1)$, $\cdots$, $P(a_m|y_1)$; $P(a_1|y_2)$, $P(a_2|y_2)$, $\cdots$, $P(a_m|y_2)$; $\cdots$; $P(a_1|y_n)$, $P(a_2|y_n)$, $\cdots$, $P(a_m|y_n)$. Where, the data may either be discrete or continuous.

3) If the features are conditionally independent, the following equation can be deducted according to Bayes' theorem:

$$P(y_i \mid x) = \frac{P(x \mid y_i) \cdot P(y_i)}{P(x)}$$

As can be seen from the above equation, since $P(x)$ is fixed for y_i, it is only necessary to calculate which class can generate the biggest numerator. And because the features are conditionally independent, Eq. (1) can be deducted as the following:

$$P(x \mid y_i)P(y_i) = P(a_1 \mid y_i)P(a_2 \mid y_i) \cdots P(a_m \mid y_i)P(y_i)$$
$$=P(y_i)\prod\nolimits_{j=1}^{m} P(a_j \mid y_i) \quad (1)$$

Based on the above analysis, the entire classification process is divided into three phases, as shown in Figure 2.

In the preparation phase, the main task is to preprocess training and test text sets. As this phase has a major impact on the subsequent phases, sufficient attention needs to be paid to it. In the training phase, the task is to generate a classifier and the main work is to calculate priori probabilities as well as the conditional probabilities that each feature belongs to each class, and to store the results as classifier. The prediction phase contains relatively simple work, which is to use the classifier formed through training to test which class the test data belongs to.

4 Hadoop-based parallelization of naive Bayes text classification

This section parallelizes NB algorithm by using Hadoop distributed system which provides HDFS and MapReduce. According to the principle

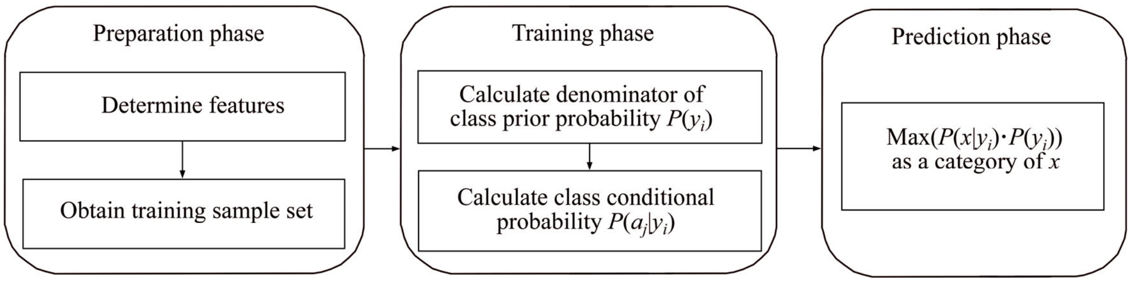
Preparation phase
Determine features
Obtain training sample set
Training phase
Calculate denominator of class prior probability $P(y_i)$
Calculate class conditional probability $P(a_j|y_i)$
Prediction phase
Max($P(x|y_i)\cdot P(y_i)$) as a category of x

Figure 2 Three phases of naive Bayes classification

and analysis of NB algorithm as above, the design and implementation of PNBA on Hadoop are also divided into two phases, namely, training and prediction.

4.1 Hadoop-based parallelization of classifier's training process

The training process of Hadoop-based classifier contains two sequential MapReduce jobs [1], in which the output of the first MapReduce job is the input of the second job, as shown in Figure 3.

The first MapReduce job achieves the following tasks. In Map phase, each Mapper receives a portion of the data block from the training TFIDF text vector, splits each record into a specific key-value pair <class, text feature vector> and aggregates all the vector weights for each class. In Reduce phase, each key-value pair is aggregated globally by class and the model attribute vector ScoresFeatureAndLabel is obtained and stored as an intermediate result.

The second MapReduce job achieves the following tasks. In Map phase, each Mapper receives a block of data from the previous MapReduce output and splits each record into two key-value pairs as <class, sum of class feature vector> and <feature, sum of feature vector>. Then the Mapper aggregates all the features locally for each class and aggregates all the features in all classes. In Reduce phase, each key-value pair is globally aggregated and the two attribute vectors, WeightsPerLabel and WeightsPerFeature are obtained and stored as intermediate results.

Finally, according to the three obtained vectors, ScoresFeatureAndLabel, WeightsPerLabel, and WeightsPerFeature, the classification model is

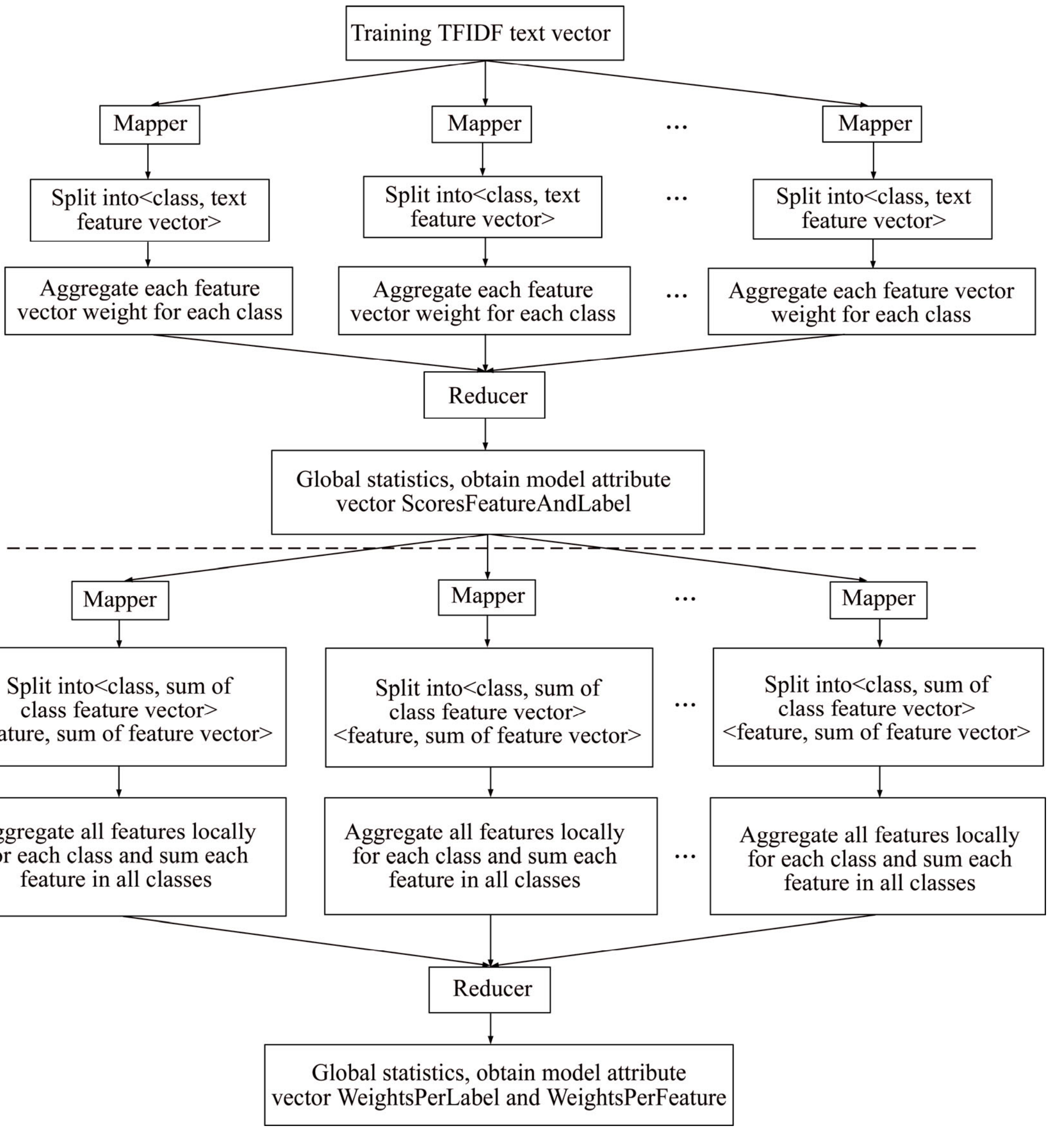
Training TFIDF text vector
Mapper
Mapper
…
Mapper
Split into<class, text feature vector>
Aggregate each feature vector weight for each class
Reducer
Global statistics, obtain model attribute vector ScoresFeatureAndLabel
Mapper
Split into<class, sum of class feature vector> <feature, sum of feature vector>
Aggregate all features locally for each class and sum each feature in all classes
Reducer
Global statistics, obtain model attribute vector WeightsPerLabel and WeightsPerFeature

Figure 3 Hadoop-based parallelization of classifier's training process

created and stored in HDFS.

4.2 Hadoop-based parallelization of classifier's prediction process

The classifier's prediction process, which can be implemented with the operation of only one MapReduce job, is relatively simple. First, each Mapper receives partial data blocks from test TFIDF text vector and calls the classification model constructed in the training process. Then, the class label of test document is obtained through calculation and comparison in accordance with Eq. (1). Finally, the calculation results of each Mapper are merged by Reducer and meanwhile the prediction accuracy of test sets is obtained.

5 Spark-based parallelization of naive Bayes text classification

Similar to parallelization based on Hadoop MapReduce, Spark-based parallelization is also implemented through two phases, namely, training and prediction. The biggest difference is that implementation of the Spark-based version is focused on the algorithm design of resilient distributed datasets (RDD).

As a parallel data structure with distributed memory, RDD is the core mechanism of Spark. It stores user data in the memory and optimizes the data distribution by controlling the partition. The parallel operation of RDD consists of two types, namely, transformation and action. Transformation means creating a new RDD on the basis of present RDD while action is to perform the actual computation on a RDD and then to return a simple-datatype value or to output data in RDD to the memory [13].

Polynomial NB text classification model is adopted in this paper. After the training set is given, class conditional probability $P(a_j|y_i)$, the maximum likelihood estimation of word frequency in the training set, is computed as follows:

$$\hat{P}(a_j \mid y_i) = \frac{N_{y_i a_j}}{N_{y_i}} \tag{2}$$

where N_{y_i} is sum of all feature words' frequency in training set of samples y_i, and $N_{y_i a_j}$ is determined by the word frequency including the feature word a_j in the training set which belongs to class y_i with the dependence as follows:

$$N_{y_i a_j} = \sum tf(a_j \mid y_i) \tag{3}$$

To avoid dilution of the classifier's precision when computing estimation of word frequency $P(a_i|y_j)$=0, additive smoothing [23] is introduced. When adding λ ($\lambda \geq 0$) to Eq. (2), the improved equation is as follows:

$$\hat{P}(a_j \mid y_i) = \frac{N_{y_i a_j} + \lambda}{N_{y_i} + \lambda n} \tag{4}$$

where n is the number of class/classes. When the training set is large enough, this method not only avoids the probability of 0 but also keeps the classification results not be affected. When λ=1, it is named as Laplace smoothing, and while λ<1, it is called Lidstone smoothing.

Equation (1) is multiplied by several conditional probability values (from 0 to 1), which would result in the underflow of floating number. By taking logarithm on both sides of the equal sign and changing the multiple multiplication to a chain addition, the following NB classifier is obtained:

$$\begin{aligned} \lg(P(x \mid y_i)P(y_i)) &= \lg(\hat{P}(y_i)\prod_{j=1}^{m}\hat{P}(a_j \mid y_i)) \\ &= \lg(\hat{P}(y_i)) + \sum_{j=1}^{m}\lg(\hat{P}(a_j \mid y_i)) \end{aligned} \tag{5}$$

5.1 Parallelization of classifier's training process

After vectoring the text, it is necessary to use training samples which have been marked categories to set up text categorization model. From Eq. (4), it can be shown that for this algorithm, during the training process, both the probability for each class $P(y_i)$ and the probability for each feature in every class $P(a_j|y_i)$ must be calculated. And that means to show the number of each class and the total number of each feature value in each class.

Spark supports the operation of HDFS to conduct row processing of text data. So we first present training samples as key-value pairs with each row of class labels and feature vectors and store them in HDFS. Then the training process is manifested in the transformation process of RDD as shown in Figure 4.

The explanation of Figure 4 is as follows (note: Spark APIs are written in italics for easy understanding).

1) Firstly, read all text vectors and form RDD.

2) Then, conduct the map operation for RDD

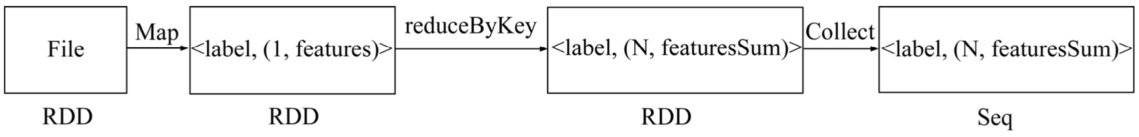

Figure 4 RDD transformation in training process

and map each row in the form of (label, (1, features)), that is (key, value), where label is the class number, 1 represents the document number and features indicate feature value vectors of certain text. Local partial merge is conducted at the same time.

3) Take label as key to conduct reduceByKey. Add the value for the same key and obtain (label, (N, featuresSum)). Here N is the document number which belongs to certain label class and featuresSum represents adding all the value of features with the same label. The number of final label is also the class number.

Hereto we can count the frequency and number of each class as well as the frequency of each feature for each class. Next the training process is presented as the following.

Algorithm 1: Parallelization of NB classifier's training process based on Spark

```
Input: the preprocessed training set
Begin
    Step 1. Define ZeroCombiner [class, (text number, conditional probability in this
            class of each feature)] for map data structure
    Step 2. Initialize the value of ZeroCombiner and calculate the total number and
            feature vector sum for local samples
            for each class $i$ do
                Calculate the total number and feature vector sum for global samples
            end for $i$
    Step 3. Obtain class number C and total number of training samples $N$.
            Calculate the denominator of class prior probability $P(y_i)$ and take the
            logarithm, piLogDenom = math.log ($N$+$C$ * lambda)
    Step 4. for each class $i$ do
            Obtain the number of samples in every class $n$, calculate class prior probability,
            pi(i) = math.log ($n$ + lambda) – piLogDenom and store it in a one-dimensional vector pi.
            for each feature vector $v$ in the class do
                Calculate the denominator of class conditional probability $P(a_j|y_i)$ and
                take the logarithm, thetaLogDenom = math.log (Sum(featuresSum) + numFeatures * lambda).
                Calculate the class conditional probability, theta($i$)($v$) =
                math.log(featuresSum(v) + lambda)– thetaLogDenom and store it in
                two-dimensional matrix theta.
            end for $v$
          end for $i$
End
Output: the training model made up of matrix theta and vector pi
```

After the training process, the NB classification model, mainly represented by sparse matrix of class prior vector and class conditional probability, is obtained.

5.2 Parallelization of classifier's predicting process

Next, we conduct the parallelization of predicting process based on the established model. Firstly, the test samples are input to form the RDD. Secondly, for the text vector in RDD, we use the map function based on the trained model to calculate the probability of text samples for each class. Then we take the class with the maximum probability as the class mark. Finally, the classification results for the test samples are stored in HDFS.

Algorithm 2: Parallelization of NB classifier's predicting process based on Spark

```
Input: the preprocessed test set
Begin
    Step 1. Present the test text in the matrix form as dataMatrix.
    Step 2. Calculate the probability belonging to
        each class, i.e.,
        pi.add(theta.mmul(dataMatrix))
    Step 3. Take out the maximum value, i.e.,
        result.argmax()
Output: the classification results
```

6 Experiment design and analysis

The cluster of this experimental platform is comprised of one master and nine slaves (slave 1–9). The serial version and the Hadoop-based version are written in Java while Spark in Scala. The node deployment in the cluster is shown in Table 1.

The Chinese corpus used in this paper is Sogou Corpus (SogouC) provided by Sogou Labs which includes 17910 documents classified by the following 9 categories namely, education, culture, sports, finance and economics, IT, health, recruit, traveling and military. The English corpus adopts Twenty Newsgroup text dataset in UCI machine learning repository. The dataset is made up of 19996 documents which composed of 20 different categories of news. Since the given dataset is unity in scale, we copied the original dataset to make up datasets of different scale. Table 2 indicates datasets used in the experiment. The feature of computing time for NB algorithm in the prediction process is the same as training phase. Hence, this section only lists the comparison of training time in the experiment.

6.1 Relationship between training time and partition number

The partition number is a supposition of parallelism granularity. In order to evaluate the influence of different partition numbers on the training time of text set at various scales, experiments are conducted for the former 4 text sets with different partition numbers in Table 1 when setting the slave number as 9 in Spark cluster platform. Test results are presented in Figure 5.

According to Figure 5, it can be seen that with the increase of partition numbers, training time for each text set decreases at first and then increases. This is because that the number of partition increases will result in the increase of parallelism degree and decrease of training time for each partition, meanwhile the network communication cost of collecting data from each partition will increase. When the partition reaches certain degree, computation time approaches to a definite value and the network communication cost is continuously increasing. Thus, it will demonstrate a trend of decreasing at first and then increasing. For text sets at different scale, the corresponding partition number for the optimal training time is also different from each other. For text sets TN-5000 and TN-10000, the optimum partition number is 20–25 while for TN-40000, it is 35–45. This is because the increase of partition number decreases the data size of each partition for text sets at large scale.

6.2 Relationship between node number and training time

In this experiment, training time for text sets of different scale with various slave numbers is tested in Spark cluster platform. The result is shown in Figure 6.

From Figure 6, it can be seen that with the

Table 1 Information of cluster node deployment

Node	CPU	Memory	Network	JVM version	Hadoop version	Scala version	Spark version
Master	4 core	8 GB	1 GB/s	1.7.0	1.2.1	2.10.4	1.6.0
Slaves 1–9	4 core	8 GB	1 GB/s	1.7.0	1.2.1	2.10.4	1.6.0

Table 2 Text set used in experiment

Text set	Source of text set	Language	Number of categories	Number of text	Scale
TN-5000	Twenty Newsgroup	English	20	5000	12 M
TN-10000	Twenty Newsgroup	English	20	10000	21.8 M
TN-20000	Twenty Newsgroup	English	20	19996	43.9 M
TN-40000	Twenty Newsgroup	English	20	39992	87.9M
SogouC-Reduced	Sogou Corpus	Chinese	9	17910	48.2M

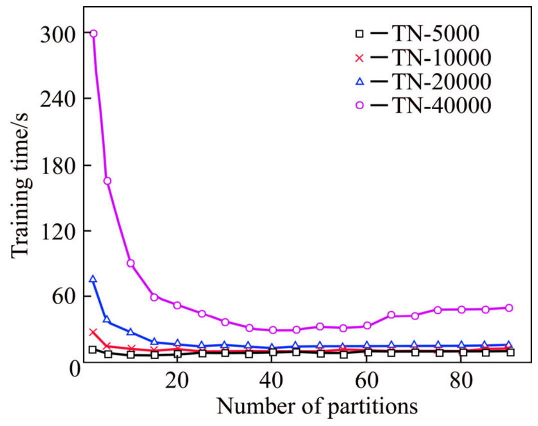

Figure 5 Relationship among partition number, text set scale and training time

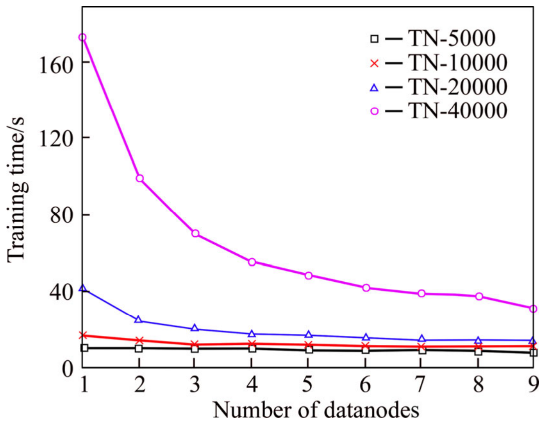

Figure 6 Impact of node number and text set scale on training time

increase of the slave number, the training time decreases gradually. When the slave number fixes, the training time required increases unceasingly with the growing scale of text set. To analyze the performance of Spark, it is necessary to carry out the quantitative calculation for speedup and expansion ration of the data in Figure 6. Experimental analysis is made from the following three aspects.

6.2.1 Analysis of system speedup of naive Bayes algorithm

Speedup represents how much execution speed of parallel algorithm has increased with respect to execution speed of serial algorithm [24]. It is a significant indicator to evaluate the performance of parallel computation and to measure the performance and effect of algorithm parallelization. Suppose the runtime for the serial algorithm (namely single node) is T_s and T_p is the runtime of parallel algorithm in p computational node, then the speedup $S_p=T_s/T_p$. Greater speedup indicates higher parallel efficiency and performance.

Figure 7 describes the speedup of the training time of NB algorithm under the environment of Spark. As the number of slave increases, speedup curve shows an increasing tendency. The speedup of text set TN-5000 and TN-10000 is close to 1 with rather gentle curve while the speedup of text set TN-20000 and TN-40000 is greater than 1 with obvious increasing trend of the curve. When the slave number fixes, the smaller the size of the text set is, the less obvious the speedup is. With the increase of text set scale, the speedup curve for the same data size rises gradually with an increasing number of the slave.

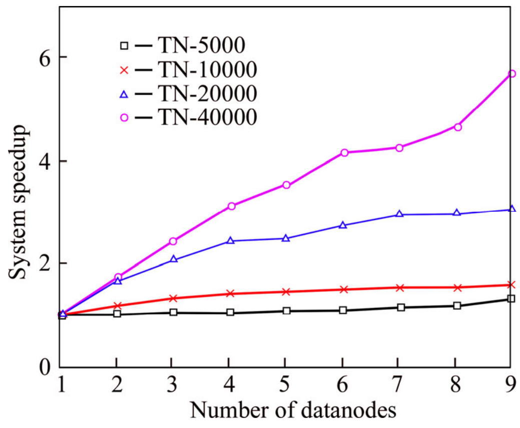

Figure 7 Speedup of training time of different text set

6.2.2 Analysis of system scalability of naive Bayes algorithm

Scalability describes the ability for performance of parallel algorithm scaling up with the increasing number of slave [24]. It indicates the utilization ratio of the cluster in the implementation of parallel algorithm. The equation of scalability is $J=S_p/p$. Here S_p represents speedup; p denotes slave number. In general, J is equal or less than one. When it is close to one, the scalability is better.

Figure 8 presents the scalability curve of naive Bayes algorithm system in Spark cluster. The scalability curve of parallel algorithm shows a downward trend with the increase of the slave. For text sets TN-5000 and TN-10000, their scalability curve decreases rapidly while the scalability curve for text set TN-40000 decreases gently and turns to be relative stable. This illustrates that in a Spark cluster environment, the speedup of PNBA increases obviously. However, with the increasing data size and slave number of the text set, the scalability is gradually stabilizing.

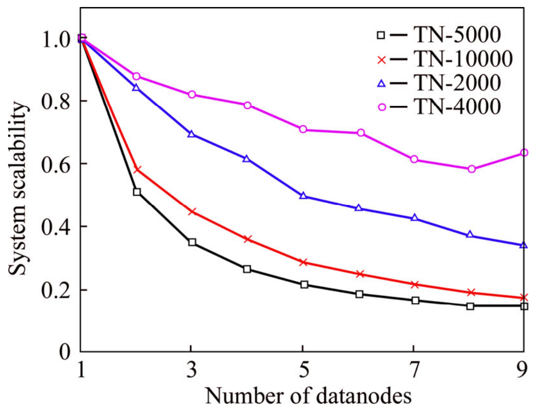

Figure 8 Scalability of training time for different text set

6.2.3 Analysis of data scalability of Naive Bayes algorithm

Data expansion ratio embodies variation trend of training time with the change of text set scale. From Figure 9, it can be seen that when the slave number is nine on both Spark and Hadoop platform, with the increasing text set scale, training time of naive Bayes algorithm increases linearly and it has good scalability.

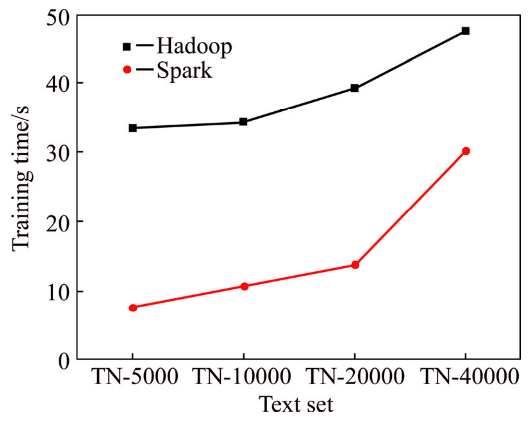

Figure 9 Relationship between training time and text set scale

6.3 Comparison of training time of text classification on various platforms

This experiment has tested the classifier's training time at different scale on serial, Hadoop and Spark platform, respectively. The result is shown in Figure 10. From Figure 10, it can be seen that with the increasing data size of the text set, the training time of these three methods all increases. Through simple calculation, it can be obtained that the ratio of training time between Spark and serial algorithm increases from 8.55 to 17.23. This indicates that the efficiency of parallel algorithm based on Spark is much higher than serial algorithm. Meanwhile, the execution efficiency of Spark based on memory computing is better than Hadoop.

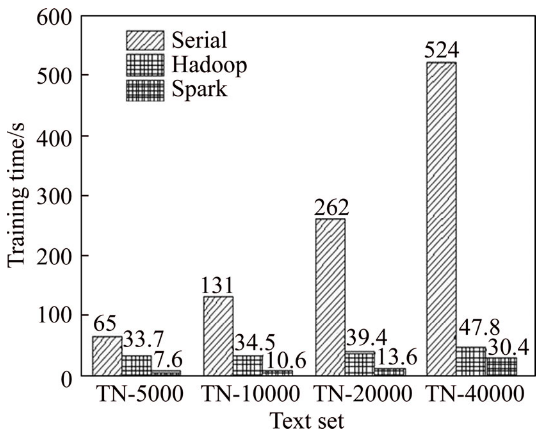

Figure 10 Classifier's training time in three platforms

6.4 Analysis of predicting results of text classification

In order to verify the effectiveness of Spark parallel algorithm, we carried on statistical analysis on the accuracy of the prediction for the training model in serial, Spark and Hadoop platforms, respectively. Firstly, we gave a test on both Chinese and English text sets. The Chinese text set is SogouC-Reduced which is composed of 17910 documents of 9 categories. TN-20000 which is made up of 19996 documents of 20 categories is chosen as the English text set since its scale is close to the Chinese one. The test results of classification accuracy of the above text sets are shown in Table 3.

Then, we test the classification accuracy of four different kinds of English text sets. The test results are presented in Table 4.

As can be seen from the above prediction results in Table 4:

Table 3 Classification accuracy of Chinese and English text sets

Text set	Number of training set tests	Number of test set texts	Number of predicted error texts			Classification accuracy rate/%		
			Serial	Hadoop	Spark	Serial	Hadoop	Spark
TN-20000	11999	7997	801	874	864	89.98	89.07	89.19
SogouC-Reduced	10746	7164	1107	1202	1169	84.55	83.22	83.68

Table 4 Classification accuracy of different kinds of text sets

Text set	Number of training set tests	Number of test set texts	Number of predicted error texts		Classification accuracy rate%	
			Hadoop	Spark	Hadoop	Spark
TN-5000	3000	2000	239	217	88.34	89.15
TN-10000	5982	4018	674	656	82.74	83.67
TN-20000	11999	7997	874	864	89.07	89.19
TN-40000	23905	16095	994	989	93.52	93.86

1) The classification accuracy rate of English text set TN-20000 is 6% higher that Chinese SogouC-Reduced. The reason is that during the word segmentation of Chinese text sets, it is hard to avoid the loss of some key phrases of different categories due to the inaccurate segmentation. Hence, the absence of feature words would result in inaccurate classifier training and then exert influence in the final classification accuracy. However, the classification accuracy rate of Chinese text averages above 83%, which can meet the requirement completely.

2) Generally, the larger the text set, the higher the classification accuracy rate. Because the lager text set means the larger scale of the training text set, the classifier model for training would be more accurate and the classification accuracy rate of test samples would be higher too. In Table 4, the classification result based on Hadoop and Spark is roughly in line with this rule.

3) Whether it is the Chinese or English text set, the classification accuracy rate in single-machine environment is 1% higher than that based on Hadoop and Spark. Moreover, the difference between Hadoop and Spark is within 1%. The above results indicate that the accuracy of the proposed parallel classification algorithm is guaranteed.

7 Conclusions

With the rapid growth of data amount and feature space dimension under the background of big data, the parallelization of traditional Chinese text classification algorithms will significantly improve its running efficiency. For the sake of this, the paper proposed a parallel naive Bayes algorithm based on Spark. According to both theoretical and experiment aspects, the following conclusions are proved. When the proposed algorithm is used to process large-scale texts, it shows good speedup as well as scalability. Specifically, the processing time is greatly reduced compared with serial algorithm while the processing efficiency is significantly improved compared with Hadoop-based version. Therefore, the proposed Spark-based parallel algorithms can better meet the requirement of large-scale text data mining.

References

[1] ZHOU Li-juan, WANG Hui, WANG Wen-bo. Parallel implementation of classification algorithms based on cloud computing environment [J]. Telkomnika Indonesian Journal of Electrical Engineering, 2012, 10(5): 1087–1092.

[2] LIU Bing-wei, BLASCH E, CHEN Yu, SHEN Dan, CHEN Gen-she. Scalable sentiment classification for big data analysis using Naive Bayes classifier [C]// IEEE International Conference on Big Data. Silicon. Valley, CA, USA: IEEE, 2013, 194(101): 99–104.

[3] GROPP W, LUSK E, SKJELLUM A. Using MPI: Portable parallel programming with the message-passing interface [M]. Cambridge: MIT Press, 1999.

[4] BERMAN F, FOX G, HEY A. Grid Computing: Making the global infrastructure a reality [M]. Hoboken, NJ, USA: Wiley & Sons, 2003.

[5] ZHANG Qi, CHENG Lu, BOUTABA R. Cloud computing: State-of-the-art and research challenges [J]. Journal of Internet Services and Applications, 2010, 1(1): 7–18.

[6] WHITE T. Hadoop: The definitive guide [M]. Sebastopol, CA, USA: O’Reilly Media, Inc, 2009.

[7] DEAN J, GHEMAWAT S. MapReduce: Simplified data processing on large clusters [J]. Communications of the ACM, 2008, 51(1): 107–113.

[8] SRIRAMA S N, BATRASHEV O, JAKOVITS P, VAINIKKO E. Scalability of parallel scientific applications on the cloud [J]. Scientific Programming, 2011, 19(2): 91–105.

[9] ZAHARIA M, CHOWDHURY M, FRANKLIN M J, SHENKER S, STOICA I. Spark: cluster computing with working sets [C]// Proceeding HotCloud’10 Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing. Boston: USENIX Association Berkeley, 2010: 1–8.

[10] ZAHARIA M, CHOWDHURY M, DAS T, DAVE A, MA J, MCAULEY M, FRANKLIN M J, HENKER S, STOICA I. Resilient distributed datasets: A fault-tolerant abstraction for

in-memory cluster computing [C]// Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation. San Jose: USENIX Association Berkeley, 2012: 1–15.

[11] JIANG Tao, ZHANG Qian-long, HOU Rui. Understanding the behavior of in-memory computing workloads [C]// Proceedings of 2014 IEEE International Symposium on Workload Characterization. Raleigh, NC, 2014: 22–30.

[12] REYES-ORTIZ J L, ONETO L, ANGUITA D. Big data analytics in the Cloud: Spark on Hadoop vs MPI/OpenMP on Beowulf [C]// Proceedings of the INNS Conference on Big Data 2015 Program San Francisco. Francisco, CA, 2015: 121–130.

[13] LIU Zhi-qiang, GU Rong, YUAN Chun-feng, HUANG Yi-hua. Parallelization of classification algorithms based on SparkR [J]. Journal of Frontiers of Computer Science and Technology, 2015, 9(11): 1281–1294. (in Chinese)

[14] YAN Bo, YANG Zi-jiang, REN Yi-tian, TAN Xing, ERIC L. Microblog sentiment classification using parallel SVM in apache spark [C]// Proceeding of 2017 IEEE International Congress on Big Data. Honolulu, USA: IEEE, 2017: 282–288.

[15] LIU Peng, TENG Jia-yu, DING En-jie, MENG Lei. Parallel k-means algorithm for massive texts on Spark. [J]. Journal of Chinese Information Processing, 2017, 31(4): 145–153. (in Chinese)

[16] TOMAS P, VIRGINIJUS M. Application of logistic regression with part-of-the-speech tagging for multi-class text classification [C]// Proceeding of the 2016 IEEE 4th Workshop on Advances in Information, Electronic and Electrical Engineering. Vilnius, Lithuania: IEEE, 2016: 1–5.

[17] CATALDO M. Enhanced vector space models for content-based recommender systems [C]// Proceeding of the Fourth ACM Conference on Recommender Systems. New York, USA: ACM. 2010: 361–364.

[18] Nature language processing & information retrieval [EB/OL] 2016. http://ictclas.nlpir.org/.

[19] LONG Jun, WANG Lu-da, LI Zu-de, ZHANG Zu-ping, YANG Liu. WordNet-based lexical semantic classification for text corpus analysis [J]. Journal of Central South University, 2015, 22: 1833–1840.

[20] ZHANG Wen, YOSHIDA T, TANG Xi-jin. A comparative study of TF*IDF, LSI and multi-words for text classification [J]. Expert Systems with Applications, 2011, 38(3): 2758–2765.

[21] OWEN S, ANIL R, DUNNING T. Mahout in action [M]. Manning Publications, 2010.

[22] VIKAS K V, BINDU K. R, LATHA P. A. Comprehensive study of text classification algorithms [C]// Proceeding of 2017 International Conference on Advances in Computing, Communications and Informatics. Udupi, India: IEEE, 2017: 1109–1113.

[23] RENNIE J D, SHIH L, TEEVAN J. Tackling the poor assumptions of naive Bayes text classifiers [C]// Proceedings of the Twentieth International Conference on Machine Learning (ICML). Washington, DC, 2003: 661–623.

[24] SUN X, ROVER D. Scalability of parallel algorithm-machine combinations [J]. IEEE Trans Parallel and Distributed System, 1994, 5(6): 599–613.

(Edited by HE Yun-bin)

中文导读

面向大规模中文文本分类的朴素贝叶斯并行 Spark 算法

摘要：针对互联网中中文文本数据量激增使得对其作分类运算的处理时间显著延长的问题，提出并实现了一种基于内存计算模型 Spark 的并行朴素贝叶斯中文文本分类算法，主要利用弹性分布数据集编程模型，实现了朴素贝叶斯分类器训练过程和预测过程的全程并行化算法。为便于比较，同时实现了基于 Hadoop-MapReduce 的并行朴素贝叶斯版本。实验结果表明，在相同计算环境下，对同一数据量的中文文本集，基于 Spark 的朴素贝叶斯中文文本分类并行化算法在加速比、扩展性等主要指标上明显优于基于 Hadoop 的实现，因此能更好地满足大规模中文文本数据挖掘的要求。

关键词：中文文本分类；朴素贝叶斯；Spark；Hadoop；弹性分布式数据集；并行化