ICIC Express Letters
Volume **10**, Number **1**, January **2016**

ICIC International ©2016 ISSN 1881-803X
pp. **235–241**

A SENTIMENT ANALYSIS PARALLEL ALGORITHM BASED ON MAPREDUCE FOR NETWORK INFORMATION

Qicheng Liu and Ying Cong

School of Computer and Control Engineering
Yantai University
No. 32, Qingquan Road, Laishan District, Yantai 264005, P. R. China
{ytliuqc; congying8010}@163.com

Received July 2015; accepted October 2015

Abstract. *A sentiment analysis parallel algorithm based on MapReduce for network information with an improved I/O method is designed to improve the performance of the traditional algorithm when dealing with massive data. Considering the characteristics of network information, especially the microblog information, we adopt the sentiment dictionary, tf-idf and weighted Naive Bayes algorithms. The results show that the algorithm is effective and in the condition of large amounts of data, the parallel algorithm is efficient against sentiment analysis for microblog information.*
Keywords: Microblog information, Parallel sentiment analysis, MapReduce model, Improved I/O method

1. **Introduction.** Network information which is represented by the microblog information contains rich emotion information. Sentiment analysis for network information, especially for microblog information has become a new direction. Currently, sentiment analysis methods mainly involve two methods based on machine learning [1] and semantic knowledge [2]. Pang et al. applied the machine learning algorithms to sentiment analysis firstly [3]. Zhu et al. proposed semantic similarity and semantically related field to analyze the sentiment tendencies [4]. And their work is representative.

After lots of research, some scholars found the results of sentiment analysis are affected by the feature representation algorithms, weighting formulas and classifiers [5-7]. Some researchers regarded nouns, adjectives, adverbs and n-gram as features respectively and compared the analysis results [8]. Paper [9] pointed out that the feature weighting algorithm affects the precision and recall rate directly. Meanwhile, in order to improve the efficiency, many scholars have used MapReduce model to deal with massive data [10]. For example, the algorithms based on vector space model on Hadoop platform [11] and the parallel support vector machine method [12] have been proposed. About sentiment analysis, parallel Rocchio and KNN (K-Nearest Neighbor) algorithms based on MapReduce have been designed.

During the previous studies of sentiment analysis, the researchers rarely considered the differences of microblog information and normal text. And all the key steps of sentiment analysis are implemented in parallel which is not common. In this paper, we consider the characteristics of microblog information and innovatively design a sentiment analysis parallel algorithm based on MapReduce with an improved I/O method which is different from our previous method and all the key steps are implemented in parallel.

In Section 1, we introduce some related work. And in Section 2, the parallel algorithm is designed. And further we verify the effectiveness and efficiency of the parallel algorithm in Section 3. At last, we make a conclusion.

2. Sentiment Analysis Parallel Algorithm for Microblog Information.

2.1. Problem statement.

Assuming that the predefined set of emotion category of texts is

$$C = \{c_1, c_2\}$$

in which c_1 and c_2 indicate positive and negative emotion respectively. The text set is

$$D = \{d_1, d_2, \ldots, d_n\}$$

The task of sentiment analysis is analyzing the emotion category of d_i $(i = 1, 2, \ldots, n)$ in text set and assigning an emotion category which is labeled as c_1 or c_2.

Compared to the normal text, network information has its own characteristics. In this paper, we selected algorithms purposefully and cut massive microblog information into splits to calculate respectively based on MapReduce.

2.2. Parallel algorithm design of sentiment analysis based on MapReduce.

2.2.1. *Emotional feature extraction parallel algorithm.* Emotional feature extraction is extracting the feature words. Different from normal texts, microblog information is shorter. So we designed parallel feature extraction algorithm based on the expanded HowNet sentiment dictionary. The input and the output of the parallel algorithm based on MapReduce in this paper are different from our previous work. In the following parallel algorithm, the emotion words are just extracted without being counted.

Algorithm 1.

Step 1. Read the words in the sentiment dictionary and save them in *emotmap*. Texts are segmented to words and the emotion words are extracted in Map process.

Input: Text of microblog information

Output: $< textName, w >$

(1) Text is segmented and the segmented words are saved in *segArray*

(2) **for** each word **in** *segArray* **do**

  if the word is the name of the text **then**

    textName = word

  else if word in *emotmap* **then**

    wordlist. add (word)

(3) **for** i=0 **to** *wordlist*. length **do**

  Output $< textName, wordlist.\text{get}(i) >$

Step 2. The results are merged in Reduce process.

Input: $< textName, \text{list}(w) >$

Output: The global $< textName, w >$

(1) **for** each w **in** list(w) **do**

  global-wordlist. add (w)

(2) **for** $j = 0$ **to** *global-wordlist*.length **do**

  Output $< textName, global\text{-}wordlist.\text{get}(j) >$

2.2.2. *Feature vector weighting parallel algorithm.* Feature weighting is calculating the weight of the feature word. In this paper, parallel *tf-idf* algorithm based on MapReduce with the improved I/O method which is different from our previous work is designed. The value of *tf-idf* is the weight of feature word and it is calculated as Formula (1).

$$tf\text{-}idf = tf * idf \tag{1}$$

In Formula (1), *tf* (term frequency) represents word frequency, which is calculated in Formula (2). And the *idf* (inverse document frequency) is calculated as Formula (3).

$$tf(w, d) = \frac{n(w, d)}{\sum_i n(w_i, d)} \tag{2}$$

In Formula (2), $n(w, d)$ is the number of the emotion term w appearing in text d and the $\sum_i n(w_i, d)$ is the total number of all the feature words appearing in text d.

$$idf(w) = \log\left(\frac{|D|}{|D(w)|}\right) \tag{3}$$

In Formula (3), $|D|$ represents the total number of texts, and $|D(w)|$ indicates the number of texts which contain w, the emotional feature word.

The feature weighting parallel algorithm is described in Algorithm 2.

Algorithm 2.

Step 1. Map reads the output of the test texts processed by Algorithm 1. Output the global $< textName, w >$.

Step 2. The counting part is processed in Reduce. The total number of feature words in the text and the number of each emotional feature word appearing in the text are counted. So the value of *tf* can be calculated according to Formula (2).

Input: $< textName, \text{list}(w) >$
Output: $< textName, (w, tf) >$
(1) $length = 0$; HashMap *map*
(2) **for** each w **in** list(w) **do**
(2.1) *length* ++
(2.2) **if** w not in *map* **then**
map. put (w, 1)
else *map*. put (w, count(w)= count(w) + 1)
(2.3) tf = count $(w)/length$
(3) Output $< textName, (w, tf) >$

Step 3. The second Map selects the texts which contain the feature word. The input is the output of the training texts processed by Algorithm 1.

Input: The global $< textName, w >$
Output: $< w, textName\ (w) >$
(1) **for** each w **in** the global $< textName, w >$ **do**
(1.1) textList [*textName*]. add (w)
(1.2) **if** w is not in *wordList* **then**
wordList. add (w)
(2) **for** each w **in** *wordList* **do**
if w in textList [*textName*] **then**
Output $<$ w, *textName* (w) $>$

Step 4. Reduce counts the number of the texts which contain the emotional feature word. The number of training texts has been known, so the value of *idf* can be calculated according to Formula (3).

Input: $< w, \text{list}\ (textName\ (w)) >$
Output: $< w, idf >$
(1) $countText = 0$
(2) **for** each *textName* **in** list (*textName* (w)) **do**
countText ++
(3) $idf = \log(trainCount/countText)$
(4) Output $< w, idf >$

Step 5. The pairs $< w, idf >$ are stored in the classifier file.

Step 6. The third Map reads $< textName, (w, tf) >$. Analyze the name of the text, emotional feature word in this text and its *tf* value. Output $< w, (textName, tf) >$.

Step 7. Reduce reads *tf* and *idf* of the word and calculates *tf-idf* according to Formula (1).

Input: $< w, \text{list}(textName, tf) >$, classifier file
Output: $< (textName, w), tf\text{-}idf >$

(1) Analyze (*textName*, *tf*) and obtain *textName* and the *tf* of the word
(2) Read *idf* of the feature word from the classifier file
(3) Calculate $tf\text{-}idf = tf * idf$
(4) Output $< (textName, w), tf\text{-}idf >$

2.2.3. *Weighted Naive Bayes parallel algorithm.* Naive Bayes algorithm has been used in the studies of classification. And some scholars have improved the traditional algorithm and proposed weighted Naive Bayes algorithms [14,15] to improve the performance. The classification model is shown as Formula (4).

$$c_{wNB} = \arg\max_{c_i \in C} \left\{ p(c_i) \prod_{j=1}^{n} p(w_j|c_i)^{wt(w_j)} \right\} \tag{4}$$

In Formula (4), $p(c_i)$ represents the priori probability of the text d_i $(w_1, w_2, \ldots, w_n)$ belonging to category c_i, and $p(c_i) = \frac{N(c_i)}{\sum N(c_i)}$ in which $N(c_i)$ is the number of training texts belonging to c_i and $\sum N(c_i)$ is the total number of training texts. $p(w_j|c_i)$ represents the conditional probability of emotional feature word w_j in text d belonging to the category c_i. $p(w_j|c_i) = \frac{N(w_j,c_i)+1}{N(c_i)+|V|}$, in which $N(w_j, c_i)$ is the number of texts containing feature word w_j in category c_i and $|V|$ is the smoothing factor and its value is the total number of categories. $wt(w_j)$ is the weight of feature word w_j.

The posterior probability is calculated through the priori probability multiplied by the conditional probabilities. And the category with the maximum posterior probability will be regarded as the classification result of the text. In this paper, a parallel algorithm of microblog information sentiment analysis based on weighted Naive Bayes is designed. The parallel algorithm is described in Algorithm 3.

Algorithm 3.

Step 1. In the training phase, the input of Map is the output of the training texts processed by Algorithm 1. For each emotional feature word, select the training texts which contain this term. The pseudo-code of Map function is the same as Step 3 in Algorithm 2.

Step 2. The prior probability has been known in advance. The number of the texts which contain the feature word in each category can be counted, and then the conditional probability of the word will be achieved.

Input: $< w, \text{list}(textName\ (w)) >$
Output: $< (w, c_i), (p(c_i), p(w|c_i)) >$
(1) $countPos = 0$; $countNeg = 0$; $p(c_1) = N(pos)/N$; $p(c_2) = N(neg)/N$;
(2) **for** each *textName* **in** list (*textName*) **do**
  if text category is positive **then**
    $countPos$ ++
  else if text category is negative **then**
    $countNeg$ ++
(3) $p(w|c_1) = (countPos + 1)/(N(pos) + 2)$
  $p(w|c_2) = (countNeg + 1)/(N(neg) + 2)$
(4) Output $< (w, c_i), (p(c_i), p(w|c_i)) >$

Step 3. The results of the training phase are stored in the classifier file.

Step 4. In the classification phase, Map reads $< (textName, w), tf\text{-}idf >$ which is the output of Algorithm 2. Output $< textName, (w, tf\text{-}idf) >$.

Step 5. The posterior probabilities are calculated. The category of text will be the category with the largest posterior probability according to Formula (4).

Input: $< textName, \text{list}\ (w, tf\text{-}idf) >$, classifier file
Output: $< textName, c_{wNB} >$
(1) **for** each w **in** list $(w, tf\text{-}idf)$ **do**

Analyze the *tf-idf* and get $p(c_i)$, $p(w|c_i)$ from classifier file

(2) $c_{wNB} = \arg\max_{c_i \in C} \left\{ p(c_i) \prod_{j=1}^{n} p\left(w_j|c_i\right)^{tf\text{-}idf(w_j)} \right\}$

(3) Output $< textName, c_{wNB} >$

3. Experiments.

3.1. Experiment of algorithm's precision.

We adopt microblog information as the datasets to test the parallel algorithm's precision. In the experiment, cross-validation method is used. The texts are divided into five groups with the same number randomly. In each time, we regard one group as the test data and the other four groups as training data to carry out the algorithm. Five experimental results are shown in Table 1.

Table 1. Precision of sentiment analysis parallel algorithm

Times	1	2	3	4	5	Average
Precision (%)	82.0	84.5	76.0	77.0	78.5	79.6

The table shows that under the condition of complex network information, the average precision of parallel algorithm is 79.6%, which is better than the results of our previous work. The result proves that the parallel algorithm proposed in this paper is more effective for sentiment analysis of network information.

3.2. Comparative experiment between serial algorithm and parallel algorithm based on MapReduce.

In the experiment, we measure the time using serial algorithm and parallel algorithm on 4-node cluster respectively for different sizes of data. The experimental results are shown in Table 2.

Table 2. Performance comparison between serial algorithm and parallel algorithm

Experiment number	Size of file	Running time of serial program	Running time of parallel program	Speedup
1	356MB	544s	284s	1.92
2	667MB	897s	352s	2.55
3	1.3GB	1691s	579s	2.92
4	6.2GB	7568s	2203s	3.44

In comparison of the performance of the algorithms, speedup is used as the evaluation indicator. From Table 2, we can find that with the data scale expanding, the time of the programs is increasing. And the time of parallel algorithm is far less than serial algorithm, especially when dealing with large-scale data. The speedup is higher and higher. The results indicate that the parallel algorithm has good performance when processing large-scale data.

3.3. Comparative experiment of speedup in different clusters.

In the experiment, we process the data of 6.2GB in the clusters with 2, 3 and 4 nodes respectively and record the time of the parallel algorithm. Experimental results are shown in Figure 1.

We find that with the increase in the number of nodes, speedup is nearly linear. When there are 2, 3 and 4 nodes in the cluster, their speedup is very close to 2, 3 and 4, and the results are ideal. Because MapReduce needs to take some time to proceed task scheduling, data segmentation and recombinant and nodes communication. So the parallel algorithm has high efficiency and stability. Meanwhile, the parallel algorithm proposed in this paper solved the speedup exception problem which appeared in our previous research.

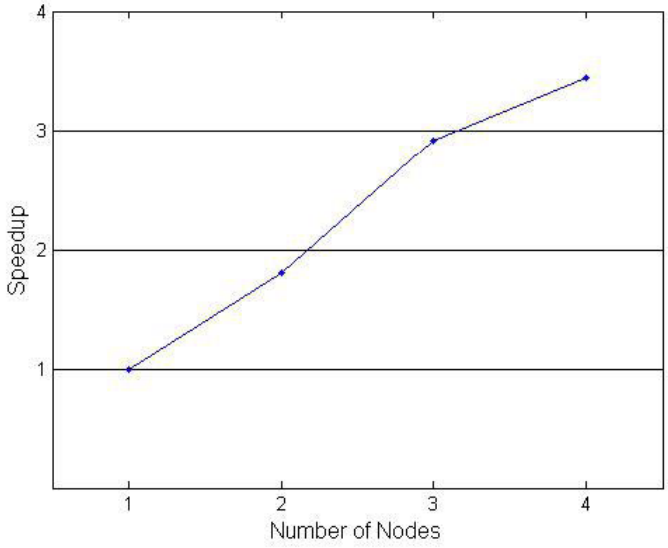

FIGURE 1. The speedup of different clusters

4. **Conclusions.** Considering the differences between network information which is represented by the microblog information and normal text information, we designed an effective sentiment analysis parallel algorithm based on MapReduce for network information. In the condition of large amounts of data, the parallel algorithm has the better speedup, which indicates the parallel algorithm's high efficiency.

On the basis of this study, according to the characteristics of grammar and vocabulary in different network information, we will find the best algorithms for different objects, in that we can not only improve the efficiency of the algorithm but also can improve the precision of sentiment analysis further.

Acknowledgment. This work is supported by the National Natural Science Foundation of China (No. 61170224, 61403329); the Science and Technology Development Plan of Shandong Province of China (No. 2012GGB01017); the Natural Science Foundation of Shandong Province of China (No. ZR2012FL07, ZR2013FQ020).

REFERENCES

[1] F. Sebastiani, Machine learning in automated text categorization, *ACM Computing Surveys*, vol.34, no.1, pp.1-47, 2002.

[2] P. D. Turney and M. L. Littman, Measuring praise and criticism: Inference of semantic orientation from association, *ACM Trans. Information Systems*, 2003.

[3] B. Pang, L. Lee and S. Vaithynanthan, Thumbs up? Sentiment classification using machine learning techniques, *Proc. of the EMNLP*, Philadelphia, pp.79-86, 2002.

[4] Y. Zhu, J. Min, Y. Zhou et al., Semantic orientation computing based on HowNet, *Journal of Chinese Information Processing*, vol.20, no.1, pp.14-20, 2006 (in Chinese).

[5] B. Pang and L. Lee, Opinion mining and sentiment analysis, *Foundations and Trends in Information Retrieval*, vol.2, nos.1-2, pp.1-135, 2008.

[6] T. Xia and Y. Chai, An improvement to TF-IDF: Term distribution based term weight algorithm, *Journal of Software*, vol.6, no.3, pp.413-420, 2011.

[7] L. Lu, Y. Wang and W. Yang, A method of sentiment classification for Chinese comments based on Naive Bayesian, *Journal of Shandong University (Engineering Science)*, no.6, pp.7-11, 2013 (in Chinese).

[8] H. Tang, S. Tan and X. Cheng, Research on sentiment classification of Chinese reviews based on supervised machine learning techniques, *Journal of Chinese Information Processing*, vol.21, no.6, pp.88-94, 2007 (in Chinese).

[9] Z. Zhai, H. Xu, B. Kang et al., Exploiting effective features for Chinese sentiment classification, *Expert Systems with Applications*, vol.38, no.8, pp.9139-9146, 2011.

[10] D. Jeffrey and G. Sanjay, MapReduce: A flexible data processing tool, *Communications of the ACM*, vol.53, no.1, pp.72-77, 2010.
[11] X. Xiang, Y. Gao, L. Shang et al., Parallel text categorization of massive text based on Hadoop, *Computer Science*, vol.38, no.10, pp.184-188, 2011 (in Chinese).
[12] Z. Zhao, Y. Xiang and J. Wang, Text classification based on parallel computing, *Journal of Computer Applications*, vol.33, no.z2, pp.60-62, 2013 (in Chinese).
[13] Y. Yu, X. Xiang and L. Shang, Research on parallelized sentiment classification algorithms, *Computer Science*, vol.40, no.6, pp.206-210, 2013 (in Chinese).
[14] K. Cheng and C. Zhang, Naive Bayesian classifiers using feature weighting, *Computer Simulation*, vol.23, no.10, pp.92-94, 2006 (in Chinese).
[15] H. Zhang and S. L. Sheng, Learning weighted Naive Bayes with accurate ranking, *Proc. of the 4th IEEE International Conference on Data Mining*, pp.567-570, 2004.