Design and Implementation of Parallelized LDA Topic Model Based on MapReduce

Duan-wu YAN*, Tie-jun LI, Xiong-fei YANG and Kun CHEN

Department of Information Management, School of Economics & Management, Nanjing University of Science and Technology, Nanjing 210094, China

Corresponding author

Keywords: MapReduce framework, LDA topic model, Gibbs sampling, Parallel computing.

Abstract. In order to solve the efficiency bottleneck of non-parallel LDA topic model while processing large-scale text datasets, a parallel LDA topic model computing framework based on MapReduce is designed and implemented. Performance testing of the parallel LDA topic model is also conducted by using the bibliographic sample data of articles and patents. Experiment shows that, the parallel LDA topic analysis process based on MapReduce framework is feasible. Compared with non-parallel LDA model, the parallel LDA topic model process can obviously improve the analysis efficiency for large-scale text datasets.

Introduction

In recent years, the topic model has gradually become a hot technology in the field of text data analysis. Using the topic model method to analyze the text sets, we can find the hidden knowledge between words in the text information. LDA (Latent Dirichlet Allocation) algorithm is a common topic analysis algorithm. The algorithm aims at modeling text content, which can be used in text clustering, topic mining and domain knowledge analysis. For example, Smith et al.[1] used the LDA model to analyze topic division on the Blekko website's text; Guan Peng et al.[2] combined the LDA topic model with the theory of knowledge life cycle for topic mining and knowledge evolution reveal of scientific literature. Zhang Chenyi et al.[3] proposed a microblog generation model MB-LDA which based on LDA for topic mining of microblog texts. In terms of the optimization and improvement of LDA model application, Yao Quanzhu et al.[4] used the standard method in Bayesian statistical theory to solve the problem of number selection of LDA model topics. However, with the growth of dataset scale, the traditional non-parallel LDA has gradually exposed the shortcomings of long processing time. In order to improve the efficiency of LDA operation, the parallel LDA topic model analysis process is proposed and implemented under the computing framework of MapReduce, and the experimental detection of process efficiency of the parallel LDA algorithm is also conducted through bibliographic text datasets.

Related Work

LDA Topic Model

LDA topic model is a probability generation model, which can be used to identify the latent topic information in the text collection, so it is commonly used for text classification or clustering. The LDA topic model generates a three-level Bayesian network, and uses the implied topic of the text to associate words and texts. The probability distribution of feature words represents the different topics of the text set, and the probability distribution of different topic represents each text. The LDA topic model uses the bag of words to assume that each text is regarded as a word vector, so that the text information that the machine can't handle can be converted into the digital information that the machine can handle and easily model. At the same time, this method only considers the frequency of words appearing in the text without considering the order of words, which simplifies the complexity of the problem.

The advantage of the LDA topic model is that it can effectively mine the text potential topics, but
the training set will be read repeatedly during the operation, which restricts the operation efficiency of the LDA topic model on large-scale datasets. With the increasing amount of data, the problem of LDA topic model computing text topic information longer is becoming more and more prominent, and the relevant researchers have improved the algorithm in different aspects according to their own needs. At present, there are two ways to improve the efficiency of LDA operation, namely, designing parallel LDA algorithm and improved algorithm to reduce workload.

**Parallel Computing Framework of MapReduce**

Compared with traditional parallel programming models, the MapReduce framework shields the underlying implementation details by providing users with interfaces of the Map and the Reduce, so that the program that inherits the interfaces can run on a parallel framework. This method reduces the difficulty of parallel programming, enables users to concentrate on the design of program logic, and improves the efficiency of programming. At the same time, the computing framework of MapReduce also has good features such as fault tolerance, load balancing [5] and scalability.

The MapReduce computing framework encapsulates parallel distributed operations to provide users with two type functional functions of Map and Reduce: the Map function handles the input data, which controls the synchronization of multiple map processes and returns the result set after completion; the Reduce function is responsible for merging and sorting the data output by the Map function, and output the calculation results according to the set format. So the Map and the Reduce are the core ideas of the MapReduce framework.

**LDA Parallelization Process based on MapReduce**

This paper designs and implements a parallel LDA topic model algorithm based on the MapReduce parallel computing framework, which is used to improve the computational efficiency of topic analysis of large-scale text datasets. The algorithm processing mainly includes two processes: text vectorization and topic modeling.

**Text Vector Parallelization based on MapReduce**

The LDA performs a probability distribution operation on the text feature vector. Therefore, the input text set needs to be vectorized. The entire text vectorization includes four main links: text segmentation, weight calculation, feature selection, and vectorization generation. The specific implementation process is as follows:

**Text Segmentation.** Map stage: The main task of this stage is to divide the long text in the topic information and remove the stop words. In this paper, HanLP is used as a Chinese word segmentation tool. This tool allows users to independently expand the word dictionary to improve word segmentation accuracy. 

Reduce stage: At this stage, word frequency summarization is performed on the received list to obtain word frequency of different words in the text.

**Feature Selection.** Map stage: The main task of this stage is to filter the text word vector and select the word whose TF weight value exceeds the threshold value as the text vectorization dictionary. 

Reduce stage: The main purpose of this stage is to obtain the text feature dictionary, so the main operation of this part is to count the input feature words and store them in the text, and use the word library as the text information vectorization feature dictionary.

**Weight Calculation.** Map stage: This stage takes thematic text word vector as input, statistics the total word frequency of a single text and the word frequency of the word, and calculates the TF value of the text word. The word in the statistical text, in conjunction with the number of texts in which the word appears, calculates IDF values for different words. The obtained thematic information TF-IDF vector is stored in the Map container. 

Reduce stage: This stage is mainly to get the TF-IDF weight vector of the text. With the text identifier as a key, the calculation result is output in the form of a key value pair “text identifier,
word1: tf-idf1word2: tfi-df2...”, and all the outputs are combined and stored in the text, thereby obtaining the weight vector of the text.

**Text Vectorization.** Map phase: this stage is mainly responsible for reading the whole feature set from the created feature dictionary and matching the document weight vector in the weight calculation results with the feature dictionary. Each text is finally expressed as a feature vector with the same dimension, and the feature vector of the topic is stored in the text, and its output key value pairs are in the form of "text identifier, feature vector".

Reduce stage: the key value pairs generated in the Map stage are sorted out in this stage to generate thematic text vector files, which will be used for the computation of topic clustering.

**Parallel LDA Implementation based on MapReduce**

An important part of the LDA algorithm is the establishment of a document topic model, which usually uses Gibbs sampling to calculate the text topic distribution, but when the method faces a larger text set, the computing performance will reach the bottleneck. When the LDA model is used to model large-scale text, one of the text matrix changes can be ignored, so the text matrix calculation can be distributed to different Mapper with the MapReduce parallel computing framework to improve the operation efficiency. According to the formula (1):

\[
P(z_n^{(m)} = k | Z^{-(m,n)}, \phi^{-(m,n)}, W^{(m)}, \alpha, \beta) \propto \frac{\theta_k^{(m,n)} + \alpha_k}{\sum_{k=1}^{K} (\theta_k^{(m,n)} + \alpha_k)} \frac{\phi_k^{(m,n)} + \beta_k}{\sum_{k=1}^{K} (\phi_k^{(m,n)} + \beta_k)}
\]

In order to simplify the formula, in formula 1, \(V\) is used instead of the original \(w_n^{(m)}\). For a given global variable \(\phi\), the hidden variable \(z_n^{(m)}\) in a text \(W^{(m)}\) is independent of the other text. It can calculate the \(z_n^{(m)}\) separately, distribute the text to different operation nodes and compute the hidden variables of different text in parallel, which can improve the operation efficiency to a certain extent.

The system sends text set, topic set \(Z\) and operation frequency \(z_n^{(m)}\) to different computing nodes. At the same time, the different computing nodes share the same global frequency \(\phi\). In order to ensure that all nodes get the same \(\phi\) at the beginning of each iteration, update the global frequency \(\phi\) after each iteration.

The implementation process of the algorithm is as follows:

**LDA Algorithm Process.** Input: LDA model training parameters: topic number, text topic distribution super parameter \(\alpha\), topic feature distribution super parameter \(\beta\), iteration number.

The segmented text is sent to different data nodes.

**DO**

- Distribution of LDA model parameters;
- Execute the Mapper function;
- Execution of Shuffle operations;
- Execute the Reducer function;

**While(! Converged&&!iteration times < maximum iterations)**

The calculation of word topics based on the output of \(\phi\) and \(\theta\)

**END**

**Map Function Process.** Input: sub corpus set, topic set \(K\) and frequency \(\theta\), global frequency \(\phi\), text topic distribution hyper parameter \(\alpha\) and topic feature distribution super parameter \(\beta\).

The data format of the corpus is the matrix of \(<\text{IntegerWritable}, \text{VectorWritable}>\), in which key is a marker for the text to be clustered to distinguish different text; value is the feature vector of the text.

For (text: text set) {
  For (word: dictionary) {
    Reduction calculate frequency \(\theta\) and local frequency \(\phi\): \(\theta_k^{(m,n)} = 1\), \(\phi_k^{(m,n)} = 1\)
    Calculate the new topic \(k^*\) according to formula 1 and assign it to \(K\).
Incremental calculate frequency $\theta$ and local frequency $\varphi$: 
$$\theta_{k^*}^{(m,n)} += 1, \quad \varphi_{k^*v}^{(m,n)} += 1$$

Output: changes in $\varphi$, key value pairs $<(k^*, v; +1)>$, new topic sets, and frequency $\theta$.

**Shuffle Operation Process.** In the MapReduce operation process, the Shuffle operation is implemented within the system. The main task is to sort and merge the key value pairs $<(k^*, v; +1)>$ which the Map output, and distribute the data to the different Reduce functions according to the similarities and differences of the keys, reduce the information interaction between the Reduce functions, and raise the efficiency of the operation.

(4) Reduce function process
Input: changes in $\varphi$, key value pairs
Update the global frequency according to the change of $\varphi$.
Output: the new global frequency $\varphi$.

**Comparative Analysis of Experiments**

**Experimental Data**
The experimental data of this paper is based on the 50,147 Chinese text data about “Graphene”, including 17,354 article cataloguing data and 32,793 patent description data.

**Comparison of Experimental Results**
In order to compare the running efficiency of the parallel LDA model and the traditional non-parallel LDA model, this experiment tests the operating efficiency of the two algorithms by running text datasets of different sizes. Experiments were performed in multiple rounds of calculations and time-consuming recordings based on the number of topics, text dataset size, and number of iterations. The experimental results are shown in Table 1. It can be seen from the experimental data that, when given the number of text records and the number of iterations, the running time of the parallel LDA is shorter, indicating that the parallel LDA can significantly improve the efficiency of text analysis.

<table>
<thead>
<tr>
<th>text number</th>
<th>100 iterations</th>
<th>200 iterations</th>
<th>300 iterations</th>
<th>400 iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>parallel</td>
<td>non-parallel</td>
<td>parallel</td>
<td>non-parallel</td>
</tr>
<tr>
<td>5000</td>
<td>18.47</td>
<td>24.88</td>
<td>19.78</td>
<td>30.01</td>
</tr>
<tr>
<td>10000</td>
<td>23.08</td>
<td>45.25</td>
<td>25.51</td>
<td>55.27</td>
</tr>
<tr>
<td>15000</td>
<td>29.60</td>
<td>59.58</td>
<td>33.71</td>
<td>73.81</td>
</tr>
<tr>
<td>20000</td>
<td>33.43</td>
<td>79.80</td>
<td>37.41</td>
<td>98.55</td>
</tr>
<tr>
<td>25000</td>
<td>39.99</td>
<td>99.93</td>
<td>44.66</td>
<td>120.95</td>
</tr>
<tr>
<td>30000</td>
<td>44.64</td>
<td>117.58</td>
<td>49.67</td>
<td>144.89</td>
</tr>
<tr>
<td>35000</td>
<td>51.97</td>
<td>134.86</td>
<td>58.95</td>
<td>167.93</td>
</tr>
<tr>
<td>40000</td>
<td>55.99</td>
<td>170.33</td>
<td>64.68</td>
<td>197.99</td>
</tr>
<tr>
<td>45000</td>
<td>62.90</td>
<td>197.70</td>
<td>72.27</td>
<td>229.41</td>
</tr>
<tr>
<td>50000</td>
<td>77.07</td>
<td>230.03</td>
<td>84.16</td>
<td>260.91</td>
</tr>
</tbody>
</table>

Figure 1 as the following shows the trend of changes in the efficiency of parallel LDA and non-parallel LDA in terms of text data scale and number of iterations given the number of topics. From this figure, we can find that when the text dataset size is small, the parallelization of LDA does not show obvious efficiency advantages, and as the scale of text increases, its efficiency advantage becomes more and more obvious.
Summary

The purpose of this paper is to improve the bottleneck problem of LDA topic model in dealing with the efficiency of large-scale text set, so that it can better analyze and process text data. In this paper, a parallel LDA algorithm based on MapReduce computing framework is designed and implemented. This algorithm can control multiple map processes synchronously and improve the processing efficiency of text data. Experiments show that the processing efficiency of the proposed algorithm is significantly higher than that of the traditional non-parallelized LDA thematic model algorithm when dealing with large text sets.

The limitation of this research is that the algorithm running experiment is carried out in a single machine environment without considering the relationship between the scale of multi-host clusters and the efficiency of operation. In the follow-up work, the Hadoop cluster of different scales can be designed to detect the scalability of the parallel LDA algorithm and the relationship between the host cluster size and the operating efficiency.

Acknowledgement

This research was financially supported by the Social Science Foundation of Jiangsu Province of China (Project no: 17TQB009).

References


