International Journal of Database Theory and Application
Vol.10, No.1 (2017), pp.149-162
http://dx.doi.org/10.14257/ijdta.2017.10.1.14

Branch-combined PLSA for Topic Extraction

Jiali Lin, Zhiqiang Wei and Zhen Li*

College of Information Science and Engineering
Ocean University of China
Qingdao, China
**Zhen Li (lizhen0130@gmail.com)*

Abstract

With the developing of the Internet technology, the information on the network is expanding at the speed of geometric progression. Facing such vast network information, quickly extracting the important information becomes the urgent needs. The subject extraction model is a good solution to the problem. In this paper, a new model based on Probabilistic Latent Semantic Analysis (PLSA) is proposed which is called Branch-combined PLSA (BPLSA). BPLSA divides training data into two subsets, and trains subsets separately first, then the global training is implemented. At the same time, Message Passing Interface (MPI) is used for parallel computing to speed up the proposed method. Through the parallelization of the BPLSA, the efficiency is improved greatly.

Keywords*: Branch-combined PLSA; topic extraction; MPI; multi-process*

1. Introduction

Since the 1990s, the Internet as a distributed and open information platform has been developing rapidly. The information on the Internet is growing in an exponential way, which makes people in an era of knowledge explosion. On the one hand, the rapid growth of Internet resources has brought great convenience to people. As long as you have a computer connected to the Internet, all of the resources are available, including the technical literature, news reports, commercial advertising, video media and other information resources. On the other hand, in the face of the vast information resources, people often feel at a loss, and the difficulty of finding the needed knowledge also increases [1]. In order to facilitate the organization, search and understanding of the huge amount of information, text classification is a good solution.

Text classification is an important research. In traditional supervised learning, the topic model is built through a large number of labeled samples. But with the rapid development of data collection and storage technology, the collection of unlabeled samples becomes quite easy, and obtaining a large number of labeled samples is very difficult which takes a lot of time and effort to label the collected samples. As a result, probabilistic topic model is generated. Probabilistic Latent Semantic Analysis (PLSA) is a classical and semi-supervised probabilistic topic model, which was proposed by Hofmann Thomas in 1999 [2]. PLSA is a statistical method, which is used to analyze the words in the text to find hidden themes and the relationships between these themes without marking the document in advance [3].

In this paper, a new model called Branch-combined PLSA (BPLSA) is proposed which is an extension of PLSA. The main idea of BPLSA is to divide the whole data into two parts, then each part extracts their own themes and keywords, finally the overall extraction is implemented by using these themes and keywords. Because the training data

* Corresponding Author

ISSN: 2005-4270 IJDTA

is general very huge, the topic extraction is a very time-consuming process. To improve the computing speed, the parallel implementation of BPLSA is implemented in this paper through Message Passing Interface (MPI) [4] and multi-processing of Python.

The paper is organized as follows. In Section 2, a brief review of related works is given. In Section 3, the sources of the experimental data are introduced. In Section 4, branch-combined PLSA is proposed. The parallelization of BPLSA is given in Section 5. The evaluation results are shown in Section 6. Section 7 concludes with a summary and suggestion for future work.

2. Related Work

Text categorization refers to the process of automatic categorization according to the text content in a given classification system. Since 1990s, many statistical methods and machine learning methods have been used in automatic categorization. For text classification, two schemes are generally considered, including supervised learning and semi-supervised learning. Supervised learning is suitable for the situation with sufficient labeled data, such as the Naive Bayes Classifiers [5] and Support Vector Machines [6]. Semi-supervised learning solves the problem that the labeled data are too few to establish a good classifier. And it makes use of a large number of unlabeled data, together with a small amount of labeled data to improve the classifier [7]. Latent Semantic Analysis (LSA), PLSA and Latent Dirichlet Allocation (LDA) are three classic semi-supervised models. LSA is an index and retrieval method proposed by Scott Deerwester and others in 1990 [8]. The method is the same as the Vector Space Model [9]. The difference is that LSA maps the words and documents to the latent semantic space, which improves the precision of information retrieval [8]. LDA is also a method for text analysis. In LSA, the latent semantic space of the text is obtained by reducing the dimension of the vector space. In LDA, it is obtained by mapping the text to the topic space. PLSA was derived from LSA, and provides solid statistical foundation. In PLSA, each document is considered as the combination of several topics, where these topics or latent semantic variables are obtained using the EM algorithm. EM algorithm is a class of iterative algorithms for maximum likelihood or maximum a posteriori estimation in problems with incomplete data. The unlabeled data are considered incomplete because they come without class labels [10].

The process of training the topic model is very time-consuming. So some parallel computing methods are also used to carry out the training, such as Compute Unified Device Architecture (CUDA), MapReduce [11] and MPI [12]. CUDA is the architecture of software and hardware which is proposed by NVIDIA, and it takes GPU as the parallel computing device. It arouses the interest of many researchers, who try to implement existing algorithms on CUDA. In many cases, it accelerates computing processes by 10 to 100 times [6]. MapReduce and MPI take CPU as the parallel computing device. MapReduce is a distributed and parallel computing model for data processing, which is mainly used to organize clusters to deal with the large scale of data sets. Compared with the traditional distributed and parallel computing model, the model packages the details of task allocation, parallel processing and load balancing. It greatly simplifies the design of distributed programming, so that users can focus their attention on the presentation of distributed computing tasks [13]. MPI is a message-passing programming model, and becomes the representative and standard of this programming model. Although MPI is very large, its ultimate goal is to provide service for interprocess communication. Message passing is a kind of model which is widely used in many kinds of parallel machines, especially in distributed-memory parallel machines. MPI allows the data to be kept in memory, and saves the context for the communication and data exchange between nodes. So it can perform the iterative algorithm, and Hadoop does not have this feature.

This model has made substantial progress in some important computational applications [14].

Python is a powerful language, and it provides some libraries for parallel computing. The threading library of Python can be used to realize multi-thread parallel execution. But due to the global interpreter lock, its efficiency is very low. Then the multi-processing library generated, and it gives a good solution to the problem of global interpreter lock. The transition from a single process to a concurrent execution can be easily realized by using this library [15]. In addition, python also provides a library for the use of MPI. The library is the mpi4py library which provides a good combination of Python and MPI. It is built on MPI, so that the data structure of Python can be transferred between processes. The mpi4py library realizes the parallel architecture of MPI and Python.

3. The Collection of Datasets

In this section, the sources of the datasets and the method for collecting data are introduced.

3.1. Datasets

The data is from eight websites. The data sources and their classification are shown in Table 1. The first column of Table 1 shows the sources of the data. The next column titled "Documents" shows the size of the data used. Here $|D_l|$ means the size of the training data, and $|D_u|$ means the size of the test data.

Table 1. Data Sources and Their Classification

Websites and Their Abbreviations	Documents	
	$\|D_l\|$	$\|D_u\|$
Convention on International Trade in Endangered Species of Wild Fauna and Flora (CITES)	480	100
Fish Information & Services (FIS)	908	100
Great Barrier Reef Marine Park Authority (GBRMPA)	410	100
International Arctic Science Committee (IASC)	144	100
National Oceanic and Atmospheric Administration (NOAA)	926	100
Partnerships in Environmental Management for the Seas of East Asia (PEMSEA)	472	100
United Nations (UN)	870	100
World Customs Organizations (WCO)	1020	100

3.2. Method for Collecting Data

The data used is collected from websites using the data collecting system as shown in Figure 1. The objective of the data collecting system is to collect all papers which are published on these websites. The collecting system uses the web crawler technology, which is the most important part of the system. The main purpose of the crawler is to download the web pages on the Internet to form a mirror backup of the internet contents. A general framework of the web crawler is as shown in Figure 1. The basic workflow of the web crawler is as follows [16].

- First, the seed URLs are selected carefully. These selected seed URLs are put into the waiting queue in which the URLs will be grabbed.
- Second, the crawler takes out the URL from the waiting queue, and parses its DNS by the URL.

- Third, the crawler downloads the web page of the URL, and stores it into the database. In addition, it is necessary to put the URL into the completed queue in which the URLs have been grabbed.
- The final step is to analysis the URLs of the completed queue to find other URLs, and put them into the waiting queue. And then enter the next cycle.

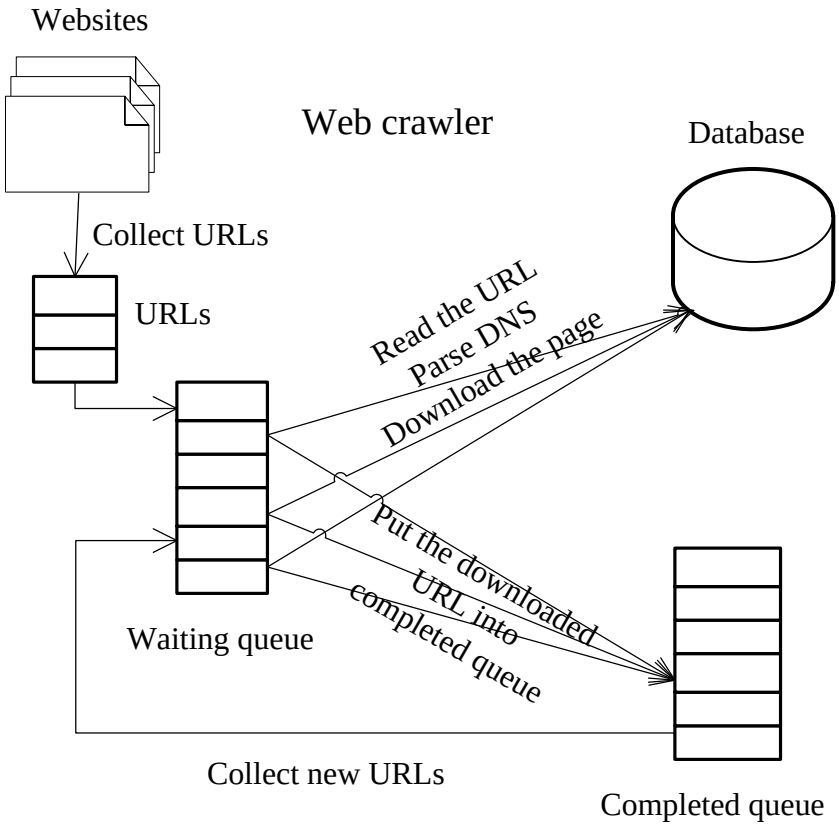

Figure 1. The Data Collecting System

The waiting queue is an important part of the crawler system. It is also a very import question that the URLs in the waiting queue will be grabbed in which order. Because it involves which URL will be grabbed first. And the method of deciding the order of the URL arrangement is called the fetching strategy. The Breadth-first Search strategy is used in the system. The basic idea of the strategy is to find the links in the download page directly, and put them into the end of the waiting queue. It means that the web crawler will first grabbed all link pages in the start page, and then select one of the link pages to continue to grab all link pages in the page. In this paper, the needed information only contains the title, time and the content of each article. So it is necessary to use regular expression to search and match string. Regular expression is a logical formula using to operate on strings. It defined some specific characters in advance. Then it uses some specific characters and the combination of these specific characters to form a regular string. The regular string is used to express a filtering logic for a string.

4. Branch-combined PLSA

At present, probabilistic topic models are all based on the same idea, that text is a random mixture of several topics. Different models will make different statistical assumptions, and get the parameters of the model in different ways [17].

4.1. Problem Definition

Several topics usually need to be discussed for a text, and special vocabulary in the text reflects the special topic which is discussed. The method for building topic model is defining the topic as the probability distribution of words, and the text is a random mixture of topics. Assuming that each website has a topic set $Z = \{z_1,...,z_k\}$, and these topics hide in the articles published on the website. These articles are $D = \{d_1,...,d_m\}$, where each instance $d \in D$ is a text document. The set D is divided into two parts of D_l which is used to train topics and D_u which is used for accuracy evaluation of the algorithm. In the BPLSA, the set D_l is put into two parts of D_1 and D_2 for better performance. All the words appeared in these articles are $W = \{w_1,...,w_n\}$. The final objective is to assign the document $d \in D_u$ to the website as accurately as possible using the training data D_l.

4.2. Learning Based on the BPLSA Model

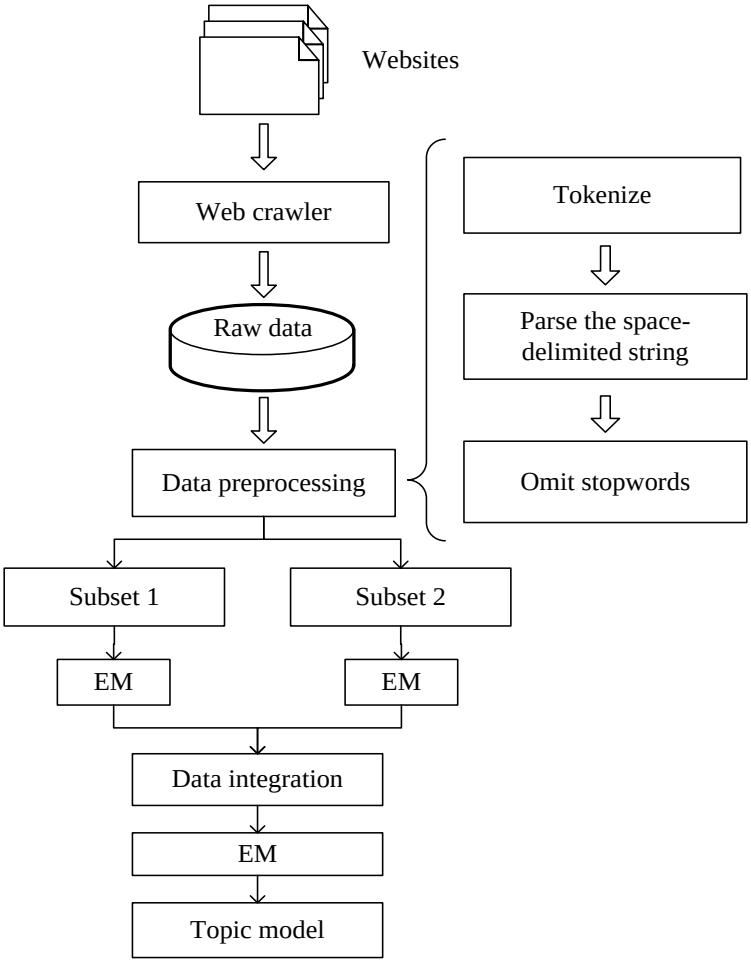

Figure 2. The Training System Based on BPLSA

The training system based on BPLSA is shown in Figure 2. The raw data is obtained by the web crawler. Some preprocessing has been applied to the raw text data before training. The imple/mentation of preprocessing is to split the text file into an ordered list of words, and remove the useless words and punctuations. The first step is to split the text, and obtain the recognizable substring and mark symbols. The second step is to lowercase everything in order to facilitate the processing, and remove punctuations. And then the next step is to parse the space-delimited string from the text and determine whether it is a valid word or not. Finally, all useless words are removed according to the stop words. Stop words refer to these words that do not directly affect the expression of the phrase. In this way, the words of every document are obtained. The training vocabulary is the union of these words.

BPLSA is an extension of PLSA. The difference between them is that PLSA uses the whole dataset for a training, and BPLSA trains subsets separately firstly, then makes the global training. The first step is to train the subset D_1, and get its vocabulary W_1 which appeared in D_1, probability distribution $P_1(w, z)$ of topic set Z on vocabulary W_1, and probability distribution $P_1(z, d)$ of document set D_1 on topic set Z. Then the same training is done on the subset D_2. And the next step is to combine these results of subsets, and implement the global training.

(1) The implementation of BPLSA

The training process of subset D_1 and D_2 is the same, D_1 is taken as the example. For the given data set D_1, w and d are known variables, but the topic z which w and d correspond to is unknown, that is a latent variable. The parameters of the probabilistic model $P(z \mid d,w)$ are needed to be estimated. But it contains the latent variable z, it is difficult to use a maximum likelihood solution. But if z is known, the problem can be easily solved. The maximum likelihood estimation is just one application of probability theory in statistics, and it is one of the methods of parameter estimation. It is known that a random sample meet a certain probability distribution, but its specific parameters are not clear. The parameter estimation is to observe the result by several experiments, and then use the result to calculate the estimated value of the parameter. The maximum likelihood estimate can be taken as a process of reverse derivation. In most cases, the result can be estimated based on the known conditions. However the maximum likelihood estimation has already known the result, and then seeks the condition which can make the likelihood most of appearing this result, and takes this condition as the estimate value. Because the result cannot be got by using the maximum likelihood estimation directly, the EM algorithm is utilized in our method. EM algorithm is to find the maximum likelihood estimation of the parameters in the probabilistic model, which is dependent on the latent variable.

For the given latent semantic topic, PLSA model assumes that each word and its document are independent of each other. And the probability between the words and documents, the probability between the latent semantic topics and the documents or the words are all subject to conditional independence [18]. On the basis of this assumption, the joint probabilistic model over $D \times W$ can be expressed as

$$P(d,w) = P(d)P(w|d),\ P(w|d) = \sum_{z \in Z} P(w|z)P(z|d) \quad (1)$$

Because the cardinality of z is smaller than the number of documents or words in the collection, the latent variable becomes the bottleneck in predicting words. It is worth noticing that the model can be equivalently parameterized by

$$P(d,w) = \sum_{z \in Z} P(z)P(d|z)P(w|z) \quad (2)$$

which is perfectly symmetric in both entities, documents and words. So in the condition of known d and w, the probabilistic distribution of z can be expressed as

$$P(z|d,w) = \frac{P(z)P(d|z)P(w|z)}{\sum_{z' \in Z} P(z')P(d|z')P(w|z')} \quad (3)$$

In turn, the values of $P(w \mid z)$ and $P(d \mid z)$ in the case of known the distribution of $P(z \mid d,w)$ can be estimated. Their equations are in which $count(d,w)$ means the number of specific word in all documents [2].

$$P(w|z) \propto \sum_{d \in D} count(d,w)P(z|d,w) \quad (4)$$

$$P(d|z) \propto \sum_{w \in W} count(d,w)P(z|d,w) \quad (5)$$

In our problem, the values of $P(z , w)$ and $P(d , z)$ are randomly assigned firstly, and the values of $\sum_{w \in W} P(z_j, w)$ and $\sum_{z \in Z} P(d_i , z)$ are 1. Then according to the equation (3), the posterior probability of the latent variable can be calculated. This is the first step of EM algorithm. According to the maximum likelihood algorithm, the distribution of $P(z , w)$ and $P(d , z)$ can be re-estimated using equation (4) and (5). This is the second step of EM algorithm. The distribution of $P(z , w)$ and $P(d , z)$ has changed, so the first step needs to be done again. Do it in this way until the parameters do not change so far.

(2) The data integration of BPLSA

The results of subsets are needed to be combined for the global training. In the process of integration, it is assumed that the topic set Z in D_1 and D_2 is the same one. Let w_i be denoted as the i*th* word in the vocabulary, z_j be denoted as the j*th* topic in the topic set. In the subset D_1, the distribution $P_1(w , z)$ is got, where $w \in W_1$ and $z \in Z$. In the subset D_2, the distribution $P_2(w , z)$ is got, where $w \in W_2$ and $z \in Z$. In the BPLSA, only a part of W_1 and W_2 are used. For the subset D_1, $P_1(w \mid z_j)$ is sorted in descending order firstly, and then the front part of every topic is extracted. Only these words W'_1 in the extracted parts are used. In the same way, W'_2 for the subset D_2 can be got.

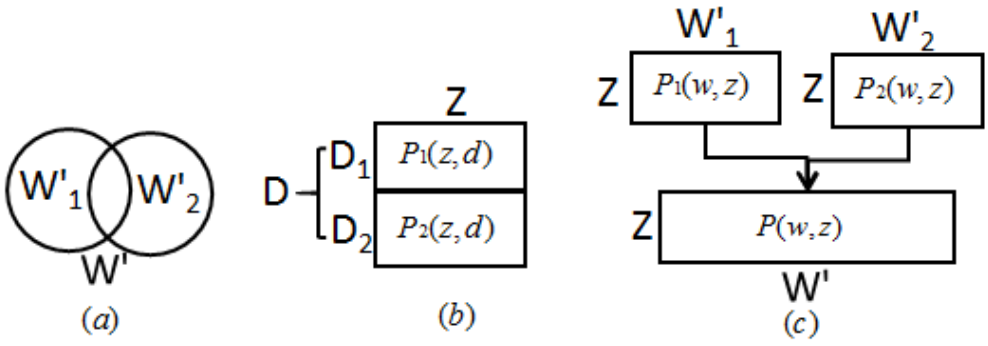

Figure 3. The Data Integration Contains the Integration of *W'* (a), the Integration of *P*(*z* , *d*) (b) and the Integration of *P*(*w* , *z*) (c)

As shown in Figure 3 (a), there may be some same words in W'_1 and W'_2, so the vocabulary W' is the union of W'_1 and W'_2. The W'_1 is put into the W' firstly, and then the W'_2 is added into W'. When adding W'_2 into W', it is needed to judge if $w_i \in W'_2$ is already in W' for the uniqueness of W'. For the $P(z , d)$ of document set D on topic set Z, it is only needed to put $P_1(z , d)$ and $P_2(z , d)$ together as shown in Figure 3 (b).

The integration of $P_1(w , z)$ and $P_2(w , z)$ is very complex as shown in Figure 3 (c). Because the W' is the union of W'_1 and W'_2, the positions of some words in W'_2 will change. It is needed to look for the position in $P(w , z)$ for $P_2(w_i , z_j)$. On the other hand, the position of the word in W'_1 will not change. The $P(w , z)$ is initialized to zeros firstly, and then the $P_1(w , z)$ is added into $P(w , z)$. For the same word, the value of $P(w_i , z_j)$ is the sum of $P_1(w_i , z_j)$ and $P_2(w_k , z_j)$.

5. Realization of Parallelization

In this section, a cluster computing system based on MPI and Linux is used to realize parallelization. The training data used in this paper comes from many websites. It is needed to train the topic model for every website. The process of training the topic model is very time-consuming, so MPI is used to realize the parallel execution of the training of all websites. The multiprocessing package of Python is used to realize the parallel execution of each training task.

5.1. The Use of MPI Cluster

In this paper, there are many websites which need to train topic models. It will take a long time in the way of training them one by one. Assume that the number of websites is *M*, and the number of parallel processes is *N*. It is needed to distribute these websites to these processes. When a process finished the training of one website, it can continue to implement other training. The variable *K* represents the maximum number of websites per process training. Its equation is

$$L = M\%N \quad K = \begin{cases} M/N, & L = 0 \\ \frac{M}{N} + 1, & L > 0 \end{cases} \tag{6}$$

All addresses of these websites are put into an array *PathArray*, and the equation (7) is used to distribute data to processes. Every process has its own unique number, which is represented by variable *rank*. In the equation (7), the *index* value of *PathArray* can be got for every process. When the value of *index* is greater than the value of *N*, it means that the task of this process has completed.

$$index = rank + i \times N, \ i \in [0,K) \tag{7}$$

The first step of parallelization is to start several processes using MPI on the cluster to realize the parallel computing of websites. And then each process as the parent process starts several child processes on its local node to realize the parallel computing of the topic training. This can make full use of every processor, and further improve the running speed. The numbers of parent processes and child processes are decided by the experimental environment and the size of training data.

5.2. Parallel Execution of EM Algorithm

In the process of training the topics, the most time-consuming part is the implementation of the EM algorithm. Because the EM algorithm is an iterative process, it is needed to continue training until the data convergence. Each training task of EM algorithm uses the last training results as the input, so it is difficult to realize the parallel execution of multiple training tasks. And because the training data is usually a large two-dimensional or three-dimensional array, each training task of EM algorithm is more time-consuming. The large amount of data can be divided into several small data, and using the concurrent execution of multiple processes can improve the speed.

The multi threads in Python are not really multi threads. Most of the situations in Python need to use the multi processes to realize the full use of CPU. In the parallel operation of multi processes, when the number of operating objects is small, directly using the multiprocessing package to dynamic start multiple processes is OK. However, if the target is too large, manually limiting the number of processes is too cumbersome. In this point, the process pool can play its effectiveness, which could provide a specified number of processes for users. When a new request is submitted to the pool, it will create a new process for the execution of the request if the pool is not full; but if the number of process in the pool has reached the maximum value, the request will wait until there is process completed in the pool, then the pool will create a new process for it [15].

On the whole, EM algorithm can be divided into two steps of E and M. In this paper , it is divided into three steps. These three steps are calculating the probability distribution of documents, topics and vocabulary, updating the probability distribution of documents and topics, and updating the probability distribution of topics and vocabulary, which are deserialized separately. Their parallel methods are the same, so there will only introduce the parallel method of the first step. The first step is using the probability distribution of document and topic, the probability distribution of topics and words to calculate the probability distribution of the documents, topics and words under the condition of known the document. To realize the parallel implementation, it is needed to create a process pool with a capacity of n firstly, which is defined by the user. And then every document is passed to the pool in turn using the imap function provided by the process pool. Final, the process pool will open a number of processes according to the number of tasks to implement the calculation in parallel.

6. Experiment

In this section, the experimental environment, the evaluation metrics, the overall performance of BPLSA compared with PLSA and the parallel results are introduced.

6.1. The Experimental Environment

The cluster system is designed according to the distributed architecture of cluster system, mainly includes the following two parts [19].

The hardware environment: the cluster uses three general-purpose computers, and they are connected by network equipment and lines. The computer used is the model with a memory of 1.9G. And the processor of it is Pentium Dual (R) CPU E2140 Intel @ 1.60GHz.

The software environment: the operating system is Ubuntu 14.04 LTS, and all nodes are in the same LAN. The used protocol for communication between nodes without passwords is SSH, and the used tools contain MPICH2 and Python 2.7.6.

6.2. Evaluation Metric

There exist many evaluation metrics for measuring the performance of the new model. In this paper, the accuracy and time are used as the metrics. There must be some kind of connection between the data of each website, and they are related to the topics. For the accuracy of the extracted web topic models, the following approach is accepted. First, it is necessary to ensure that the test data is randomly selected from web sites, and they did not participate in the training. Then, the test data is classified according to the web topic models. Finally, the accuracy of BPLSA and PLSA can be evaluated according to the accuracy of the classification results.

The dataset of each website is divided into D_l and D_u, which are all belong to one website. It is assumed that T is a function which maps from document d to its true website $o = T(d)$, and L is the function which maps from document d to its prediction website $o = L(d)$ by the new model. The *accuracy* can be defined as [20]

$$accuracy = |\{d|d \in D_u \cap T(d) = L(d)\}|/|D_u| \quad (8)$$

The metric time means the running time of the program to complete the training.

For parallel results, the accuracy and speedup are used to evaluate the performance. Speedup is defined as the ratio of the execution time of the sequential program to the execution time of the parallel program for implementing the same result, in which T_s means the sequential execution time, and T_p means the parallel execution time running on multiple processors, p is the number of used processors.

$$S_p = T_s / T_p \quad (9)$$

6.3. Performance Comparison of PLSA and BPLSA

(1) Performance on accuracy

First, the labeled data is trained separately using PLSA and BPLSA, and the topic models are obtained. Then the accuracy of the training results is obtained using the unlabeled data. Here one hundred documents of each website are randomly selected for evaluation. The accuracy of the evaluation results is shown in Figure 4. The X-axis represents different website. The Y-axis represents the accuracy of PLSA and BPLSA. The average accuracy of PLSA is 0.92875, and the average accuracy of BPLSA is 0.945. It can be observed that the accuracy of BPLSA is better than PLSA. In the PLSA, the initial values of the $P(z , w)$ and $P(d , z)$ are assigned randomly. However, in the BPLSA the subsets are trained firstly, and the results of the subsets are used as the initial values of $P(z , w)$ and $P(d , z)$. So the accuracy of BPLSA is better than PLSA.

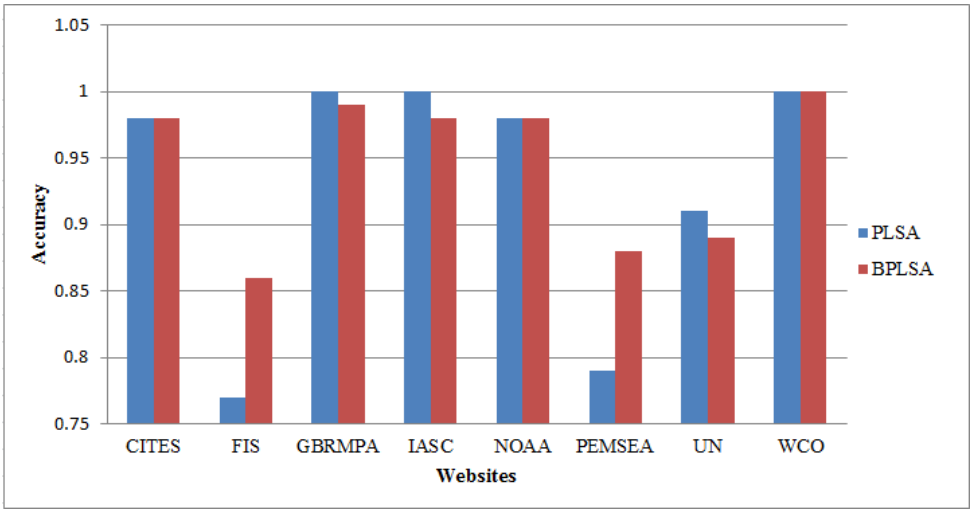

Figure 4.The Accuracy of PLSA and BPLSA for very Website

(2) Performance on time

The training time of each website is shown in Table 2. The first column shows the sources of websites. The second column shows the time of training using PLSA and BPLSA. From the second column, it can be observed that the training time of BPLSA is shorter than the training time of PLSA. The final column shows the speedup of using BPLSA rather than PLSA. When dealing with large data sets, the virtual memory will be used if the physical memory is exhausted, and the internal paging system will be used if the virtual memory is exhausted. And the speed of the subsequent memory management mode will decrease in an exponential way comparing with the former model. The size of data becomes small after partitioning, and the physical memory is enough, so the speed is faster.

Table 2. The Running Time of PLSA and BPLSA

Websites	Time		Speedup
	PLSA	BPLSA	
CITES	18644.2	14169.4	1.32
FIS	54876.6	39895.4	1.37
GBRMPA	8945.4	7373.7	1.21
IASC	1550.3	1450.9	1.06
NOAA	51902.9	38835.6	1.33
PEMSEA	14627.9	10967.4	1.33
UN	81736.9	58866.4	1.38
WCO	40801.2	30415.2	1.34

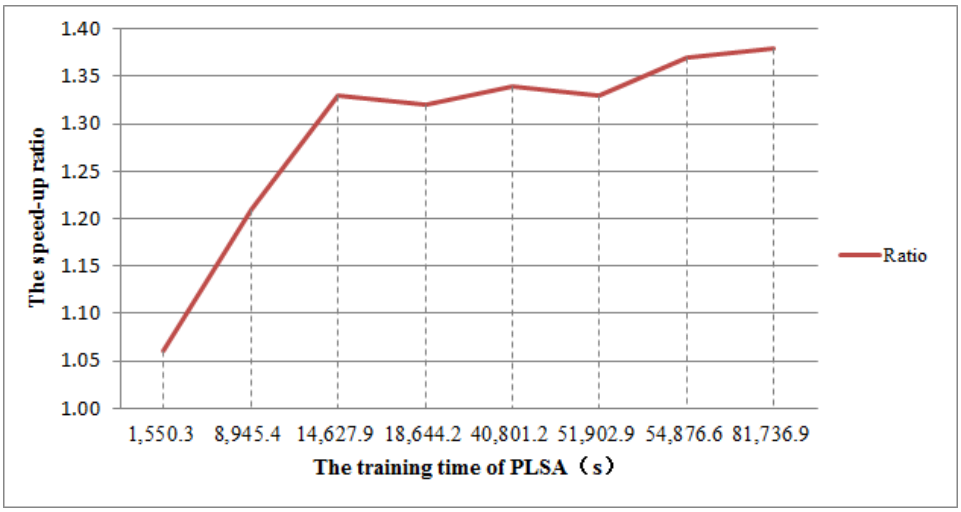

Figure 5. The Relationship between the Speedup and the Training Time

The Figure 5 shows the relationship between the training time and the speedup. With the increase of training time, the speedup is gradually increasing. The growth of the speedup is fast and obvious at the beginning. To a certain extent, the speedup basically maintains at about 0.25. The training time is related to the data size. The greater the data size is, the longer the training time is. So the Figure 5 also represents the relationship between the data size and the speedup. The greater the data size is, the better the effect of acceleration is. With the increasing of the data size, the occupied non-physical memory is more and more, so the run speed is more and more slow, and the effect of partitioning is more and more obvious. Due to the limitation of hardware conditions, the speedup is basically stable when the data size reaches a certain degree.

6.4. Parallel Result

The Figure 6 shows the accuracy of PLSA and parallel BPLSA. The X-axis represents different website. The Y-axis represents the accuracy of PLSA and parallel BPLSA. The average accuracy of PLSA is 0.92875, and the average accuracy of parallel BPLSA is 0.9475. It can be observed that the parallel execution of BPLSA does not affect the accuracy. The initial parameters of subset training are assigned randomly, so the results of BPLSA and Parallel BPLSA are slightly different.

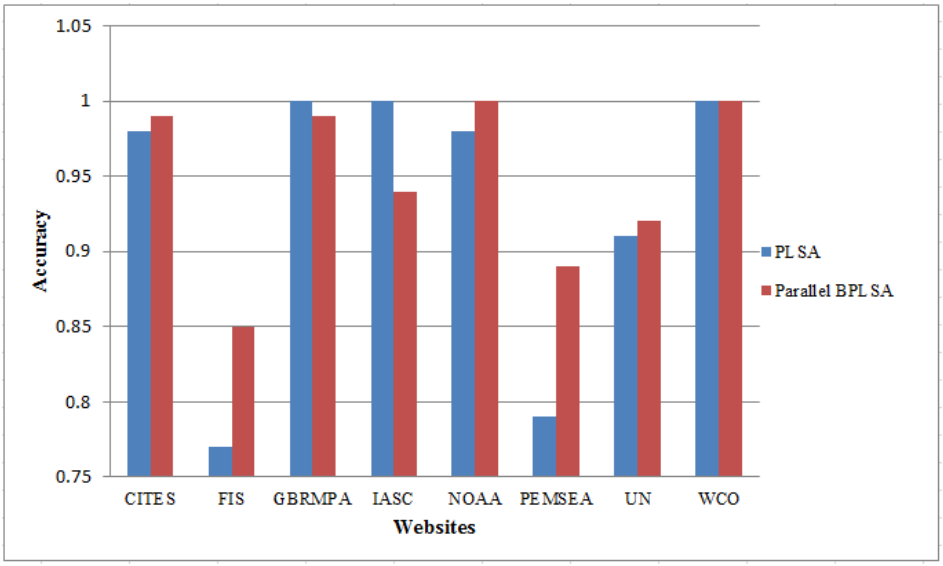

Figure 5. The Accuracy of PLSA and Parallel BPLSA for every Website

Table 3. The Overall Time of PLSA, BPLSA and Parallel BPLSA

	Overall Time	Speedup
PLSA	273085.4	1.00
BPLSA	202074.0	1.35
Parallel BPLSA	31218.2	8.75

The Table 3 shows the overall time of PLSA, BPLSA and parallel BPLSA. The speedup of PLSA and BPLSA is 1.35, the speedup of serial and parallel running of BPLSA is 6.47, and the speedup of PLSA and parallel BPLSA is 8.75. This proves that the parallel implementation is successful.

7. Conclusion and Future Work

In this paper, a new model called Branch-combined PLSA is proposed for topic extraction. The experiment in this paper also proved that the accuracy of BPLSA is higher, and the running speed is faster than PLSA. So BPLSA is a successful extension of PLSA. The parallel execution of BPLSA is also realized using MPI and Python, and the result of it is good.

As a future work, it is considered to use other topic extraction algorithms for classification. And the branch-combined idea will be used on other algorithms to study the universal of this idea. Moreover, it is planned to do further study on topic extraction with other features.

Acknowledgments

This work is supported by the National Nature Science Foundation of China (No.61602430, No.61672475, No.61402428), Self Innovation and Achievements Transformation of Shandong Province (No.2014CGZH0708) and Major Projects of Shandong Province (No.2015ZDZX05002)

References

[1] Xiaobo Jin, "A Survey on Text Classification", Journal of Automation Panorama, **(2006)** May, pp. 24-29.
[2] T. Hofmann, "Probabilistic Latent Semantic Analysis", Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence (SIGIR), San Francisco, CA, USA, **(1999)** July.
[3] D. M. Blei, "Probabilistic Topic Models", Journal of Communication of the ACM, vol. 55, no. 4, **(2012)** Aprial, pp. 77-84.
[4] CORPORATE The MPI Forum, "MPI: a message passing interface", Proceedings of the 1993 ACM/IEEE conference on Supercomputing, **(1993)** November 15-19.
[5] V. Bijalwan, V. Kumar, P. Kumari and J. Pascual, "KNN based Machine Learning Approach for Text and Document Mining", International Journal of Database Theory and Application, vol. 7, no. 1, **(2014)**, pp. 61-70.
[6] K. Sopyła, P. Drozda and P. Górecki, "SVM with CUDA accelerated kernels for big sparse problems", Proceedings of the 11th international conference on Artificial Intelligence and Soft Computing, Zakopane, Poland, **(2012)** April 29-May 03.
[7] G. R. Xue, W. Y. Dai, Q. Yang, and Y. Yu, "Topic-bridged plsa for cross-domain text classification", Proceedings of the 31st ACM Annual International Conference on Research and Development in Information Retrieval (SIGIR), Singapore, **(2008)**.
[8] S. Deerwester, S. T. Dumais*, G. W. Furnas and T. K. Landauer "Indexing by latent semantic analysis", Journal of the American Society for Information Science, vol. 41, no. 6, **(1990)** September, pp. 391-407.
[9] DL. Lee, H. Chuang, and K. Seamons. "Document Ranking and the Vector-Space Model", Journal of IEEE Software, vol. 14, no. 2, **(1997)**, pp. 67-75.
[10] K. Nigam,A. K. McCallum, S. Thrun and T. Mitchell, "Text Classification from Labeled and Unlabeled Documents using EM", Journal of Machine Learning, vol. 39, **(2000)**, pp. 103–134.
[11] Ling Wei, Yang Li and Yongjiang Wei, "Representative Information Retrieval Algorithm Based on PageRank Algorithm and MapReduce Model", International Journal of Database Theory and Application, vol. 9, no. 3, **(2016)**, pp. 25-36.
[12] Yi Wang, Hongjie Bai, M. Stanton, Wen-Yen Chen and E. Y. Chang, "PLDA: Parallel Latent Dirichlet Allocation for Large-Scale Applications", Proceedings of the 5th International Conference on Algorithmic Aspects in Information and Management, San Francisco, CA, USA, **(2009)** June 15-17.
[13] J. Dean, S. Ghemawat, "Mapreduce: Simplified Data Processing on Large Clusters", Journal of Communication of the ACM: 50th Anniversary Issue, vol. 51, no. 1, **(2008)**, pp. 107-113.
[14] Jin Lu, Kaisheng Zhang, Ming Chen and Ke Ma, "Implementation of Parallel Convolution Based on MPI", Proceedings of 3rd International Conference on Computer Science and Network Technology, Dalian, China, **(2013)**.
[15] Mark Lutz, Editor, "Programming Python, 4th edition", China Electric Power Press, **(2006)**.
[16] Demao Zhou, Zhoujun Li, "Survey of High-performance Web Crawler", Journal of Computer Science, vol. 36, no. 8, **(2009)** August.
[17] Jing Shi, Meng Fan and Wanlong Li, "Topic Analysis Based on LDA Model", Journal of Acta Automatica Sinica, vol. 35, no. 12, **(2009)** December, pp. 1586-1592.
[18] Yunying Wang, "Research on web-page semantic annotation algorithm based on PLSA model", Journal of Intelligence, Chinese, vol. 32, no. 1, **(2003)** January, pp. 141-144.
[19] Fang Meng and Qingxia Li, "Construction of MPICH2 Parallel Cluster System Based on Ubuntu", Journal of Science & Technology Information, no. 23, **(2008)**, pp. 54.
[20] Y. Yang, "An Evaluation of Statistical Approaches to Text Categorization", Journal of Information Retrieval, vol. 1, no. 1/2, **(1999)**, pp. 69–90.

Authors

Jiali Lin, She was born in 1990 and from Shandong Province, China. After graduating from China University of Petroleum since 2014, she has been in Ocean University of China for further study. Her research interests include parallel programming and Natural Language Processing

Zhiqiang Wei, He is currently a professor at Ocean University of China. His research interests include software engineering and intelligent computing.

Zhen Li, He is currently a lecture at Ocean University of China. His research interests include wearable computing, pattern recognition and IOT.