High Performance Latent Dirichlet Allocation for Text Mining

A thesis submitted for Degree of

Doctor of Philosophy

By

Zelong Liu

Department of Electronic and Computer Engineering

School of Engineering and Design

Brunel University

September 2013

Abstract

Latent Dirichlet Allocation (LDA), a total probability generative model, is a three-tier Bayesian model. LDA computes the latent topic structure of the data and obtains the significant information of documents. However, traditional LDA has several limitations in practical applications. LDA cannot be directly used in classification because it is a non-supervised learning model. It needs to be embedded into appropriate classification algorithms. LDA is a generative model as it normally generates the latent topics in the categories where the target documents do not belong to, producing the deviation in computation and reducing the classification accuracy. The number of topics in LDA influences the learning process of model parameters greatly. Noise samples in the training data also affect the final text classification result. And, the quality of LDA based classifiers depends on the quality of the training samples to a great extent. Although parallel LDA algorithms are proposed to deal with huge amounts of data, balancing computing loads in a computer cluster poses another challenge. This thesis presents a text classification method which combines the LDA model and Support Vector Machine (SVM) classification algorithm for an improved accuracy in classification when reducing the dimension of datasets. Based on Density-Based Spatial Clustering of Applications with Noise (DBSCAN), the algorithm automatically optimizes the number of topics to be selected which reduces the number of iterations in computation. Furthermore, this thesis presents a noise data reduction scheme to process noise data. When the noise ratio is large in the training data set, the noise reduction scheme can always produce a high level of accuracy in classification. Finally, the thesis parallelizes LDA using the MapReduce model which is the de facto computing standard in supporting data intensive applications. A genetic algorithm based load balancing algorithm is designed to balance the workloads among computers in a heterogeneous MapReduce cluster where the computers have a variety of computing resources in terms of CPU speed, memory space and hard disk space.

Acknowledgement

I would thank many people for their help. First of all, I would like to thank my PhD supervisor, Prof. Maozhen Li. In the process of the whole research, he always gave me the most help and guidance from end to end. In addition, I have been encouraged greatly with his advice and support so that I could face all the difficulties. Not only did I learn more about my research, but also I learned how to analyze and solve the problem.

I also thank Xiaoyu Chen, Yang Liu, Yu Zhao. They gave me lots of significant opinions. Especially their support and care in the usual life. Moreover, I still would thank the School of Engineering and Design, and Brunel University. During my PhD research years, I achieved all the aspects of the help from the School and the University.

Finally, I thank my parents, girlfriend and housemates. They gave me the greatest support and courage when I was in the most difficult time. Here, I would like to express my heartfelt thanks to all the friends who have helped me. I shall remember the days I spent at Brunel forever.

Author’s Declaration

The work described in this thesis has not been previously submitted for a degree in this or any other university and unless otherwise referenced it is the author’s own work.

Statement of Copyright

List of Abbreviations

AD-LDA	Approximate Distributed LDA
API	Application Program Interface
BP	Belief Propagation
CTM	Correlated Topic Model
DAG	Directed Acyclic Graph
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DLB	Dynamic Load Balancing
DP	Dirichlet Process
EM	Expectation-Maximization
FIFO	First in First out
GA	Genetic Algorithm
GFS	Google File System
GS	Gibbs Sampling
HDFS	Hadoop Distributed File System
HD-LDA	Hierarchical Distributed LDA
HDP	Hierarchical Dirichlet Process
ICE	Internet Communications Engine
Intel TBB	Intel Threading Building Blocks
JDK	Java Development Kit
KLD	Kullback-Leibler Divergence

K-NN	K-Nearest Neighbor
LDA	Latent Dirichlet Allocation
LSI	Latent Semantic Indexing
MCMC	Markov Chain Monte Carlo
MPI	Message Passing Interface
PAM	Pachinko Allocation Model
PLDA	Parallel Latent Dirichlet Allocation
PLSI	Probabilistic Latent Semantic Indexing
SGA	Simple Genetic Algorithm
SSH	Secure Shell
SVD	Singular Value Decomposition
SVM	Support Vector Machines
TF-IDF	Term Frequency-Invert Document Frequency
VB	Variational Bayes
VI	Variational Inference
VSM	Vector Space Module

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Background

So far, the Internet has accumulated a huge number of digital information including news, blogs, web pages, e-books, images, audio, video, social networking and other forms of data, and the number of them has been growing at the speed of the explosion continually [1]. Thus, how people can organize and manage large-scale data effectively and obtain the required useful information quickly has become a huge challenge. For example, the data is too large to use the traditional data analysis tools and techniques to deal with them. Sometimes, even if the data set is relatively small, because of untraditional characteristics of the data, the traditional methods also cannot be used [2] [3]. In addition, the expert system technology can put knowledge into the knowledge base manually by special users or domain experts. Unfortunately, this process often has some deviation and mistake, and it is time-consuming and high-cost [4] [5].

Therefore, it is necessary to develop new technologies and automatic tools which can convert massive data into useful information and knowledge intelligently. Data mining is a technique, and it is able to combine traditional data analysis methods with complex algorithms that can deal with large amounts of data. Data mining is a complex process where the unknown and valuable modes or rules are extracted from mass data. Furthermore, it is an interdiscipline, which is closely related to database system, statistics, machine learning, information science and other disciplines [1] [2] [4]. So, data mining can be seen as the result of the natural evolution of information technology. According to the processing object, it can be divided into object data mining, spatial data mining, multimedia data mining, Web mining and text mining [3] [6].

Text is the most important representation of the information. The statistics research showed that 80 percent of information in an organization was stored in the form of text, which included books, research papers, news articles, Web pages, e-mail and so on [7]. Text is able to express vast and abundant information meanwhile it contains lots of undetected potential knowledge. The whole text set is not structured data and it lacks machine-understandable semantics so that it is quite difficult to deal with a huge number of documents. So in the field of data mining, a new technology which can process the above text data effectively was proposed, which was called text mining [5] [6] [8].

1.1.1 Text Mining Techniques

Text mining was first proposed by Ronen Feldman et al in 1995, which was described like “The Process of extracting interesting Patterns from very large text collections for the purpose of discovering knowledge [6].” Text mining was also known as text data mining or knowledge discovery in texts, which was a process where the unknown, potential, understandable and useful knowledge can be found from mass text data [2] [6] [9] [10]. Meanwhile, it was also a process of analyzing text data, extracting text information and finding out text knowledge [9] [11] [12].

The main techniques of text mining contain text classification, text clustering, text summarization, correlation analysis, information extraction, distribution analysis, trend prediction and so on [13] [14] [15]. Here, text classification and text clustering are the mining methods whose object is the text set. But, the processing object of text summarization and information extraction is a single document. There are many classification methods in text classification, and the frequently-used methods include Native Bayes (NB), K-Nearest Neighbor (K-NN), Support Vector Machines (SVM), Vector Space Module (VSM) and Linear Least Square Fit (LLSF) [6][7][8][9][12][13][14][15].

In the field of text mining, machine learning experts researched and put forward probabilistic topic model, and fast unsupervised machine learning algorithms were used to find out the text hidden information automatically [16] [17] [18] [19]. At present, main mining latent semantic knowledge models are Latent Semantic Indexing (LSI) [20] [21] [22] [23] [24], Probabilistic Latent Semantic Indexing (PLSI) [25] [26] [27] [28] [29] and Latent Dirichlet Allocation (LDA) [30] [31] [32]. Their application almost covers all areas of text mining and information processing, such as text summarization, information retrieval and text classification, etc [33] [34] [35]. Especially, Blei et al put forward the LDA model in 2003, which was widely used to solve text classification, text annotations and other issues. Besides, it created a series of text processing methods which were based on probabilistic topic modeling [17] [36], and it was expanded to images, audio, video and other multimedia data processing fields [37] [38].

LDA is a probabilistic topic modeling which models discrete data sets such as text set [30]. It treats documents as the probability distribution of topics and simplifies the generative process of the text, which helps to handle large-scale text sets efficiently. In addition, LDA is a three-tier Bayesian model, which includes words, topics and documents three-tier structure. It makes each document express as a topic mixture where each topic is a probability distribution of the fixed word list.

In brief, the basic modeling process of LDA is to establish a document-word co-occurrence matrix firstly. And then, a text training set is modeled. Next, inference methods are used to obtain model parameters, such as document-topic matrix θ and topic-word matrix ϕ. Finally, the learned model will be used to predict the topic probability distribution of new documents so as to express text information [17] [30] [32].

Compared with other topic models, LDA has some unique advantages: Firstly, LDA

topic model is a completely probabilistic generative model, which can use mature probability algorithms to train models directly. And, it is easy to use the model [39]. Secondly, the size of the parameter space of LDA is fixed and has nothing to do with the size of a text set so that it is more suitable for large-scale text sets [40]. Thirdly, LDA is a hierarchical model, which is more stable than non-hierarchical models [39].

1.1.2 High Performance Computing for Text Mining

In the aspects of processing speed, storage space, fault tolerance and access speed, the traditional technical architecture and single computer with serial-based approach are more and more unsuitable to handle mass data [41] [42] [43] [44]. Parallel computing is an effective method of improving the computation speed and processing capacity of the computer system. Its basic idea is to decompose a problem into several parts, and each part is computed by an independent processor in parallel [45] [46] [47]. A Parallel computing system can be a supercomputer with multiple processors, and it also can be a cluster which is made up of several independent computers that are interconnected in some way [44] [48]. However, for parallel programming models in the traditional high-performance computing, such as PThread, MPI and OpenMP [49] [50], developers have to understand bottomed configurations and parallel implementation details.

In 2004, Jeffrey Dean and Sanjay Ghemawat who were two engineers of Google proposed the programming idea of MapReduce [51], and applied it to the parallel distributed computing of large-scale data sets. According to functional programming ideas, the MapReduce framework can divide a computational process into two phases Map and Reduce [52] [53] [54]. In the Map phase, the input data is transmitted to map functions in the form of key-value pairs. After operations, an intermediate set of key-value pairs is generated. In the Reduce phase, all the intermediate values sets with the same keys will be merged. The final result will be output in the form of key-value

pairs. For users, they only need to implement two functions map and reduce, reducing the complexity of designing parallel programming significantly [51] [52].

Then, Google CEO Eric Schmidt first proposed the concept of "cloud computing" in 2006 [43]. Each big international Internet information technology company launched a series of products in succession, showing their research results [55] [56]. For example, Google launched Google App Engine which was based on the cloud computing environment development platform. Amazon launched Amazon EC2 (Amazon Elastic Compute Cloud) which was a powerful cloud computing platform and Amazon S3 (Amazon Simple Storage Service) which can provide cloud storage service. Microsoft launched Windows Azure which was a cloud computing platform. Cloud computing is the result of comprehensive development of parallel computing, distributed computing and grid computing. In other words, cloud computing is the commercial implementation of the above computing science concepts [43] [57].

Cloud computing is a new computing mode, is also a new composite mode of computer resources, and represents an innovative business mode [55]. The basic principle of cloud computing is to distribute the required calculation, transmission and storage in local computers or remote servers into a large number of distributed computers, which means that each computer shares huge tasks of calculation, transmission and storage together. Users can gain corresponding services through the network, improving the resource utilization rate effectively and realizing resource acquisition on demand [58] [59].

The above cloud computing platforms are commercial products, which cannot be freely available to the major researchers. Thus, the emergence of open source cloud computing platforms is able to give the major researchers opportunities. The researchers can use these open source projects to constitute a cluster system which are made up of several machines in a laboratory environment, and simulate the cloud computing environment in business systems [57] [58]. Hadoop is one of the most

famous open source distributed computing frameworks, which achieves main technologies of Google cloud computing [60] [61] [62]. Basically, its core content contains Hadoop Distributed File System (HDFS) [53] [60] [61] [63] and MapReduce programming model [64] [65] [66].

Hadoop originated in the projects of Lucene and Nutch, and then it developed into a separate project of Apache foundation. It is mainly used for processing mass data [53] [60] [63]. Besides, it provides a MapReduce framework based on Java with strong portability, which can make the distributed computing apply to large low-cost clusters. Meanwhile, for individuals and companies, it has a high performance so as to lay the foundation for the research and application of their distributed computing [67] [68]. Yahoo was the first company which used, researched and developed Hadoop deeply. In Yahoo, there were more than ten thousand CPUs in a single cluster using Hadoop, and there were hundreds of researchers in the use of Hadoop [52] [54]. Nowadays, it has become the mainstream technology of cloud computing. For example, universities, research institutions and the Internet companies research and use it [67]. Along with the increasing popularity of the cloud computing concept, Hadoop will get more attention and faster development [65].

According to the digitized information report which was published by International Data Corporation (IDC) in 2011, the amount of global information would be doubled and redoubled every two years [1]. So, there is no doubt that it will give the data storage and computing power of servers much pressure, which cannot use traditional topic model algorithms to process large-scale text sets in high-performance servers [44] [69]. In addition, people cannot only hope that the improvement of computer hardware technology is able to enhance the processing efficiency, but they need to use efficient parallel computing technologies to apply to multi-core computers and cluster systems with high-performance. Through a combination of hardware and software, the accelerated processing can be accomplished in the process of complex calculations, solving the bottleneck of computation time and memory capacity. At the

same time, parallel computing can save cost to a certain extent, and complete large-scale computing tasks with a lower investment [51] [68] [70].

When dealing with mass data, the disadvantage of LDA is that the computation complexity will be higher and processing time will be longer [17] [71] [72]. Thus, how LDA can model large-scale text effectively, meet the requirements of the computation time and memory capacity, and finally find out latent topic rules is another big issue. In order to improve the LDA model in processing mass data, the parallel LDA algorithm has become one of the research hotspots.

To research MapReduce parallel algorithm and apply it to topic models has much practical significance [51] [52] [66]. Firstly, parallel algorithms can solve the problem of calculation and storage, and they can process large-scale data more efficiently. Secondly, parallel topic models are able to address many practical application problems, such as text classification, information retrieval, text summarization and so on [17] [73]. Finally, parallel topic models can analyze the massive Internet user behavior data to obtain useful information, which is able to be used in social networking websites, intelligent search and other Internet products [18] [72].

1.2 Motivation of Work

Automatic text classification involves data mining, computational linguistics, informatics, artificial intelligence and other disciplines, which is the basis and core of text mining. It is an important application field of natural language processing, and also a significant application technology of processing large-scale information [2] [4] [74]. In short, text classification technology is to make a large number of documents divide into one or a group of categories where each category represents different concept topics. Automatic text classification systems can help users organize and obtain information well, playing a significant role in improving the speed and

precision of information retrieval [5] [8]. Therefore, it has a very important research value. In text classification, the selection and implementation of classification methods are the core parts of classification systems. How to choose an appropriate classification model is an important issue [75] [76]. In addition, text feature selection is also another key technology in text classification [77] [78] [79].

The main role of making probabilistic topic model apply to text classification is to achieve the dimensionality reduction of the text data representation space by finding out latent topic information from documents. In the early stage, the most typical representative was the LSI model. The dimensionality reduction effect of its feature space is significant, but its classification performance often decreases [80] [81]. Furthermore, its parameter space grows linearly with the training data because of the high computation complexity of SVD operation in LSI. These are limitations of LSI when it is applied to practical problems. PLSI can be seen as a probabilistic improved version of LSI, which can describe the probabilistic structure of the latent semantic space [25] [26]. However, PLSI still has the problem of the parameter space growing linearly with the training set [29] [40].

Compared with LSI and PLSI, the LDA model is a completely probabilistic generative model so that it has a complete internal structure and it is able to use useful probabilistic algorithms to train and make use of the model. In addition, the size of the parameter space of the LDA model has nothing to do with the number of documents. Thus, LDA is more suitable to construct text representation model in a large-scale corpus [30] [32]. And, the LDA model has got successful applications in machine learning, information retrieval and other many fields. In the research of text classification, the LDA model is effective but its performance is not remarkable. The reason is that LDA is a non-supervised learning model so that it cannot be directly used in classification. It needs to be embedded into appropriate classification algorithms [82].

The LDA model itself is a generative model, and it is usually integrated with generative classification algorithms. In this generative integrated mode, the training of classifiers does not use the entire training corpus to gain a single LDA model, but the sub-LDA models of corresponding categories are obtained by each category of the training corpus. In fact, each sub-LDA model describes latent topics of corresponding categories of documents [32] [83]. Because of separate training, each category shares a group of topics, and topics between categories are isolated. But, one of main problems of this classification method is that the target document will take place the forced distribution of latent topics in categories which do not contain the target document, resulting in that the calculation of the generative probability produces the deviation in these categories so as to reduce the classification performance [84]. Therefore, how to choose a suitable classification algorithm combined with the LDA model to construct an effective classifier that has become a challenge.

In the LDA model, topics obey Dirichiet distribution which assumes that the emergence of a topic has nothing to do with the emergence of other topics. But in the real data, many topics have associations between them. For example, when “PC” appears, the probability of “Computer” appearing will be quite high but it is unlikely for “Hospital” to appear. Obviously, this independent assumption is inconsistent with the real data. So, when using the LDA model to model the whole text set, the number of topics in the LDA model will influence the performance of modeling greatly [82]. Therefore, how to determine the optimal number of topics in the LDA model is another research hotspot of LDA.

At present, to determine the optimal number of topics in the LDA model has two main methods: the selection method based on Hierarchical Dirichlet Process (HDP) [86] [87] and the standard method in Bayesian statistics [82]. The former uses the nonparametric feature of Dirichlet Process (DP) to solve the selection problem of the optimal number of topics in LDA. But, it needs to establish a HDP model and a LDA model respectively for the same one data set. Obviously, when processing practical

text classification problems, it will spend a lot of time in computing [86] [88]. The latter needs to specify a series of values of T manually, where T stands for the optimal number of topics in LDA. After carrying out Gibbs sampling algorithm and the related calculation, the value of T in the LDA model can be determined finally [82]. Thus, an algorithm needs to be designed to find out the optimal number of topics automatically with low consuming time and operation.

In addition, in text classification, the quality of classifiers will affect the final classification result greatly. And, the quality of classifiers depends on the quality of the training text set to a great extent. In general, if classes of the training text set are more accurate and its content is more comprehensive, the quality of the obtained classifier will be higher [89] [90]. However, in practical applications, it is quite difficult to obtain this high quality of training text sets, especially large-scale text sets. Usually, the training data contains noise unavoidably, and the noise samples will have a significant impact on the final classification result [91].

Usually, a mainstream approach of noise processing is to identify and remove noise samples from data sets [90] [91]. At present, there are two main noise processing methods based on LDA, the data smoothing method based on the LDA model [92] and the noise identification method based on the category entropy [93] [94]. They can remove a majority of noise samples from text sets effectively to some extent, but they cannot delete all the noise completely [95]. Furthermore, some normal samples will be wrongly removed as noise unavoidably [96]. Therefore, these issues also become a challenge in text classification.

In order to meet the requirements of storage capacity and computation speed, the single-core serial programming is turning to the multi-core parallel programming technology gradually [41] [42]. Parallel programming applying to clusters or servers can solve some limitations when topic model learning algorithms process mass data [69]. So, parallel topic models are able to deal with large-scale data effectively. But,

each node may have different computation speed, communication capability and storage capacity in a heterogeneous cluster so that the load of tasks is unbalanced, which will reduce the computational efficiency of the cluster [70]. Therefore, the load of tasks in a cluster should keep balance as far as possible, decreasing the synchronization waiting time of each node.

The task scheduler mainly solves the problem of load balancing. But in a heterogeneous computing environment, current main parallel computing platforms lack effective scheduling algorithms [70] [97]. For instance, there are three main schedulers in Hadoop, First in First out (FIFO), fair scheduler and capacity scheduler. They are applicable to a homogeneous environment, but they would waste a lot of resources and the overhead would be longer in a heterogeneous environment so as to affect the overall performance of the Hadoop platform. An idea of this thesis is to select and design an appropriate scheduling strategy and corresponding operation parameters. Compute nodes are in working condition all the time as far as possible so as to minimize the overhead of scheduling and synchronization, realizing load balancing and enhancing the performance of parallel programs.

1.3 Major Contributions

In simple terms, major contributions of this thesis are to improve the performance of the typical ordinary LDA algorithm from two aspects, which are increasing accuracy and speeding up. For the problems of text classification, the optimal number of topics in the model, noise processing and dealing with large-scale data sets, the appropriate and effective strategies are designed respectively based on the LDA algorithm. The followings are the details of major contributions in this thesis:

- First of all, probabilistic topic models are reviewed, such as basic concepts and modeling process of LSI, PLSI and LDA. And, their advantages and

disadvantages are analyzed and compared. For the problem that latent topics are assigned by force so as to reduce the classification performance when the traditional LDA model is used in text classification, a method of text classification based on the LDA model is proposed. Its basic idea is to use the LDA algorithm to model the text set first and then obtain the probability distribution of documents in the topic set, which can decrease meaningless features to the classification performance and compress the high-dimensional sparse feature space. In the classifier design, the performance of SVM is better than other text classification algorithms in the aspect of processing high-dimensional data and classification accuracy, so the SVM algorithm is used to construct a text classifier. It can avoid the problems of traditional text representation methods, such as VSM generates high-dimensional sparse feature space. At the same time, it also can overcome the problem that the classification performance is damaged when using LSI. In addition, using LDA to model text sets can reduce the dimensionality of the feature space effectively, shorten the training time of SVM greatly and improve the efficiency of classification.

- When the LDA model is used to model the whole text set, the number of topics will influence the modeling significantly. Thus, according to the idea of calculating sample density to measure the correlation between topics in the density-based clustering algorithm, a selection method of the optimal number of topics based on Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is proposed. Firstly, the relationship between the optimal topic model and the topic similarity is proved in theory. When the average similarity of the topic structure is the smallest, the corresponding model will be optimal. And then, according to the constraint condition, the selection of the optimal value and the parameter estimation of the LDA model are combined in a framework. By calculating the density of each topic, the most unstable topic structure is found. It is iterated again and again until the model is stable, seeking out the optimal number of topics automatically.

- Combining the data smoothing method based on the LDA model and the noise identification method based on the category entropy, a noise data reduction scheme is put forward. Its basic idea is to bring in the concept of information entropy first. After noise processing based on the category entropy, most of noise samples have been removed, improving the quality of the training text set. And then, the generative process of the LDA topic model is used to smooth data so that the quality of the training text set gets further improvement. At this moment, an iterative process of the algorithm is completed. But, the training text set is usually affected by noise samples after one time of iteration. Thus, the whole process will need K times of iteration, where the smoothed training text set can be seen as the initial training data of the next iteration. After finishing iterations, the obtained training text set is the final training text set, which will be applied to classification. The proposed strategy weakens the influence of missing out noise samples in the noise filtering phase. Meanwhile, it maintains the size of the original training set.

- By comparing and analyzing the current main methods of parallel LDA, the PLDA algorithm and its implementation process in the Hadoop platform are studied in detail. For the problem of load balancing in a heterogeneous environment, a static load balancing strategy based on a genetic algorithm for PLDA is proposed. According to the optimal principle of the divisible load theory, each data node should complete local Gibbs sampling at the same time theoretically. In addition, the concept of the mean square error is used to convert it into the fitness function of the genetic algorithm. In fact, to partition data before the start of tasks can minimize the synchronization waiting time between data nodes and improve the performance of the Hadoop cluster in a heterogeneous environment.

1.4 Structure of the Thesis

Chapter 1 introduces the background and motivation of research works of this thesis. Besides, it describes major contributions of this thesis and Chapters arrangement briefly, which made significance, content and plan of the thesis clear.

Chapter 2 first summarizes probabilistic topic models, introduces the basic concepts and the modeling process of LSI, PLSI and LDA in detail and analyzes and compares their advantages and disadvantages. And then, it summarizes three main inference algorithms of LDA model parameter, such as variational inference, belief propagation and Gibbs sampling. Next, it introduces the basic concepts, main steps and parameter settings of genetic algorithms. Finally, the related concepts of Hadoop are overviewed, which contains HDFS, MapReduce programming model, scheduling algorithms of Hadoop and the divisible load theory.

Chapter 3 summarizes key technologies of text classification, including text preprocessing, text representation model, text feature extraction and dimension reduction, text classification model, and classification performance evaluation. After introducing the Gibbs sampling algorithm, a text classification method based on the LDA model is proposed. And then, the LDA topic model and SVM classification algorithm are combined to build a text classification system. At last, the validity and advantages of the proposed classification method are proved by contrast experiments.

Chapter 4 introduces two current main methods which can determine the optimal number of topics in the LDA model, the standard method of Bayesian statistics and the method of selecting the optimal number of topics based on HDP. And then, according to the density-based clustering algorithm, the algorithm of selecting the optimal number of topics based on DBSCAN is put forward. The experiments prove its feasibility and validity. In addition, it also introduces two noise processing methods based on the LDA model, including the data smoothing based on the LDA

model and the noise identification based on the category entropy. Next, two methods are combined, and a noise data reduction scheme is proposed. At the end, the experiments verify its validity and the stable performance.

Chapter 5 summarizes and compares the current main methods of parallel LDA, including Mahout's parallelization of LDA, Yahoo's parallel topic model, the algorithm of parallel LDA based on Belief Propagation (BP) and Google's PLDA. After studying the PLDA algorithm and its implementation process in the Hadoop framework, a static load balancing strategy for PLDA based on a genetic algorithm is proposed. And, it describes the design and the implementation of the proposed strategy in detail. Finally, in the real homogeneous and heterogeneous Hadoop cluster environments, the experiments show that the PLDA algorithm is an effective method to deal with mass data. Moreover, in a Hadoop simulation environment, the experiment proves that the proposed strategy is effective and stable, and it can enhance the computational efficiency of the Hadoop cluster.

Chapter 6 summarizes the major research works and contributions of this thesis. In addition, it points out the limitations of the thesis and the direction of future research works.

Chapter 2 Literature Review

2.1 Introduction

This Chapter introduces probability topic models first, and compares and analyzes the advantages and disadvantages of LSI, PLSI and LDA. And then, the main inference algorithms of LDA model parameters are described, such as variational inference, belief propagation algorithm and Gibbs sampling. Finally, the overview of genetic algorithm, the framework of Hadoop which includes HDFS, MapReduce and main scheduling algorithms and the divisible load theory is presented.

2.2 Probability Topic Models

Recently, probability topic models have had very wide range of applications. For example, they have obtained a quite good application effect in the fields of text classification and information retrieval [18]. If a document collection is given, probability topic models will find a set of low-dimensional polynomial distribution by parameter estimation. Here, each polynomial distribution is called a topic, which is used to capture the related information among words. They can extract the comprehensible and stable latent semantic structure without the computer really understanding natural language, finding a relatively short description for the documents in the large-scale data set [98].

The main idea of probability topic models is to assume that the document is a mixed distribution of some topics, and each topic is a probability distribution of words. Thus, probability topic models can be seen as a text generation model. The generation of documents is a simple probability process based on topic models [18] [19]. For example, when a new text is to be generated, a topic distribution will be gained first.

Next, for each word in the text, a topic is obtained randomly by the topic distribution. At the end, a word will be got randomly by the word distribution of the topic.

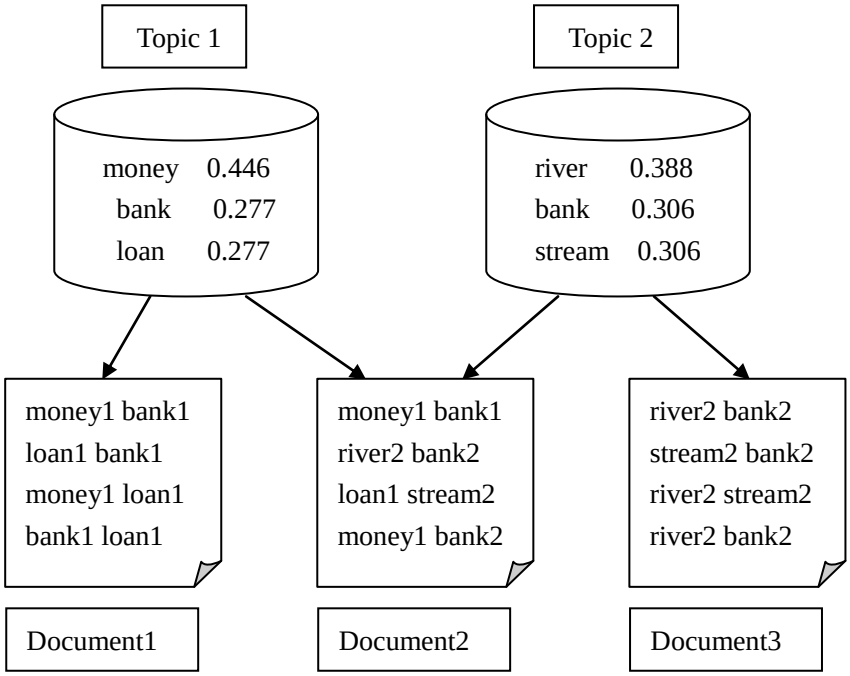

Figure 2.1: The generative process of the topic model.

Figure 2.1 shows a simple text generation process. Topic 1 and topic 2 are associated with money and river, respectively. They have different word distributions so that they can constitute documents by choosing the words which have different importance degree to the topic. Document 1 and document 3 are generated by the respective random sampling of topic 1 and topic 2. But, topic 1 and topic 2 generate document 2 according to the mixture of their different topic distributions. Here, the numbers at the right side of a word are its belonging topic numbers. And, the word is obtained by the random sampling of the numbered topic. In this section, several traditional topic models are introduced briefly. And then, the main inference algorithms of LDA model parameters will be described in detail in the next section.

2.2.1 TF-IDF Model

The Term Frequency-Invert Document Frequency (TF-IDF) model was proposed in 1983 [99]. Its approach is to establish a $V \times D$ matrix, where V represents a vocabulary that contains all the possible words. And, $|V|$ is the size of the vocabulary. D stands for a text set, and $|D|$ is the size of the text set. For each word, the Term Frequency (TF) of them is calculated in all the documents, and the inverse of the number of all the documents which contain the word is also computed. Finally, the product of TF and Invert Document Frequency (IDF) is stored in the corresponding position of the matrix.

According to the core idea of the TF-IDF model, if the frequency of a word appearing in the same one document is higher, which can be measured by TF, or the frequency of the word appearing in all the documents is lower, which can be measured by IDF, the word will be more important to the document. Thus, the importance of the whole document can be obtained by calculating the importance of all the words [103] [104]. Through some simple normalization processing, the probability distribution of these words in the whole text set can be gained [100].

However, TF-IDF is not a real topic model. Its main contribution is to convert the documents with unfixed length to the matrix with fixed length. It does not identify the semantics, and it cannot recognize and deal with synonyms and polysemes [33]. The reason is that all the words are just simple textual characters in the TF-IDF model. Another problem of TF-IDF is that the required memory space and computational time are very huge. In order to contain more word information, it is necessary to expand the size of the vocabulary. In addition, for more model training, the number of the training documents has to increase [101]. Therefore, the needed storage space which saves the TF-IDF matrix will increase by several times.

2.2.2 Mixture of Unigrams

Figure 2.2 (Left) shows that the boxes are "plates" that represent replicates. The outer plate represents documents, and the inner plate represents the repeated choice of words within a document. In addition, the filled circle represents the observed variable w. In the unigrams, the words in each document are generated from a polynomial distribution independently, which can be expressed as Equation (2.1).

$$p(w) = \prod_{n=1}^{N} p(w_n) \tag{2.1}$$

where N is the number of words and w_n represents one of words.

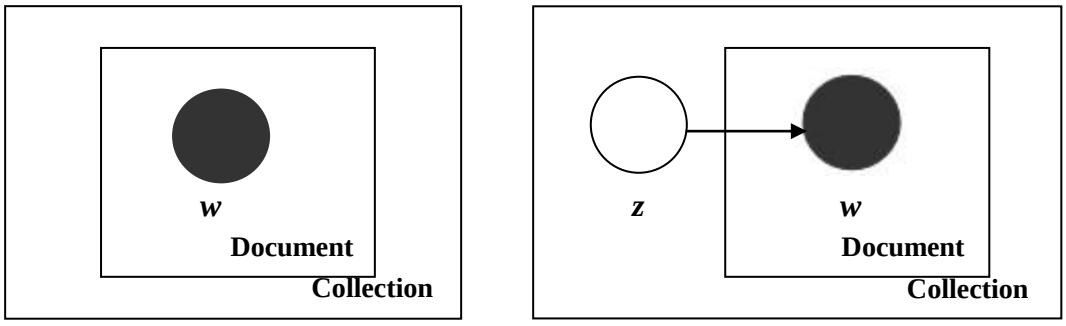

Figure 2.2: (Left) The unigrams.
(Right) The mixture of unigrams.

If a discrete random topic variable z is added into the unigrams, they will become the mixture of unigrams, which is shown in Figure 2.2 (Right). The hollow circle represents the latent variable z, and the black arrow represents the conditional dependency between two variables, namely the conditional probability. In the mixture of unigrams, for each document, a topic z is chosen first. And then, according to the conditional polynomial $p(w|z)$, N words of the document are generated independently [100]. So, the probability of each document can be obtained as

Equation (2.2):

$$P(w) = \sum_{z} P(z) \prod_{n=1}^{N} P(w_n \mid z) = \sum_{z} P(w \mid z) P(z) \tag{2.2}$$

where N is the number of words, w_n represents one of words, w represents the word variable and z represents the topic variable.

When the whole text set is estimated, the mixture of unigrams will assume that each document only express one single topic. And, the distribution of the words can be seen as the topic representation under the assumption of each document corresponding to one topic [102], which can be expressed as Equation (2.3):

$$P(d) = \prod_{w} P(w) = \prod_{w} \sum_{z} P(w \mid z) P(z) = \sum_{z} P(z) \prod_{w} P(w \mid z) \tag{2.3}$$

where d represents the document variable, w represents the word variable and z represents the topic variable. Because of the limitation of the above assumption, the mixture of unigrams is often ineffective when dealing with large-scale text set.

2.2.3 LSI Model

2.2.3.1 Basic Concepts

At the earliest, the idea of probability topic model was originated from Latent Semantic Analysis (LSA). LSA was also known as Latent Semantic Indexing (LSI), which was proposed by Scott Deerwester et al as a semantic space model in 1990 [20]. It is an extension of the Vector Space Model (VSM), addressing the problems of TF-IDF. LSI assumes that the words in the documents have some latent semantic structure. For example, there is basically the same semantic structure between the synonyms. And, the polysemes must have a variety of different semantic structures. Its principle is to use the Singular Value Decomposition (SVD) technique to convert the matrix of TF-IDF to a singular matrix. By removing the small singular value vectors and only retaining the top K maximum values, the original document vector

and the query vector are mapped from the word space to a K-dimensional semantic space. In the low-dimensional space, the latent semantic relation of the TF-IDF matrix will be kept, eliminating the influence of synonyms and polysemes and improving the accuracy of text representation [22] [23].

The LSI model has been widely used in text classification because of its significant dimension reduction effect [20]. In fact, through the SVD of the word-document matrix, the LSI model maps the original matrix to a K-dimensional latent semantic space, where K stands for the number of chosen the largest singular values. After mapping, the singular value vector can farthest reflect the dependence relationship between words and documents. So, the dimensionality of the latent semantic space is much smaller than that of the original space so as to achieve the purpose of dimension reduction [21]. In the low-dimensional space, the documents which originally do not contain or just contain a few same words may have the large similarity because of the co-occurrence relation between the words.

2.2.3.2 The Modeling Process

At present, the classic method of singular value decomposition in matrix is used to achieve LSI. One of its advantages is that its decomposition effect is good and its scalability is strong [20] [105]. First of all, according to a training corpus, a document-word matrix C is constructed. In the matrix, each element stands for the frequency of the corresponding word appearing in a document. According to the SVD algorithm, the matrix C can break up into three matrixes, which is shown in Figure 2.3 and can be expressed as follows:

$$C = U\Sigma V^T \tag{2.4}$$

where C means a $n \times m$ matrix, U stands for a $n \times n$ orthogonal matrix, V represents a $m \times m$ orthogonal matrix, and Σ is defined as follows:

$$\Sigma = diag(\sigma_1, \sigma_2, \cdots, \sigma_i) \tag{2.5}$$

where $i = \min\{n, m\}$ and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_i \geq 0$. Equation (2.4) is referred to as the singular value decomposition form of C. $\sigma_1, \sigma_2, \cdots, \sigma_i$ are the singular values of C. The column vector of U is left singular vector of C, and the row vector of V is right singular vector of C.

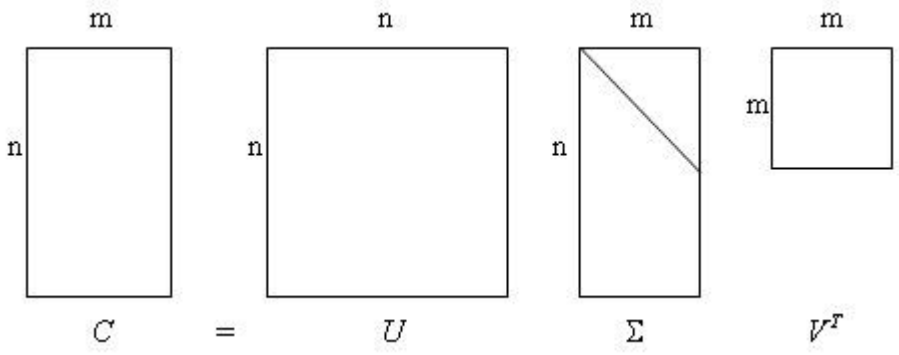

Figure 2.3: The diagram of singular value decomposition (SVD).

As shown in Figure 2.3, the matrix Σ is a diagonal matrix. The diagonal elements which are the singular values of the original matrix C are set to nonzero value in the descending order. The singular values of the matrix represent the characteristics of the matrix, and the larger singular values can stand for the main characteristics of the matrix. Therefore, the smaller singular values can be ignored so as to compress lots of space. When the matrix only keeps a part of values on the diagonal, namely k rows and k columns, the right part of U and the bottom part of V will reduce automatically. At this moment, three new matrixes are generated, such as $U_k(n \times k)$, $\Sigma_k(k \times k)$ and $V_k(k \times m)$. And then, the three matrixes are multiplied to obtain a new similar matrix C_k.

$$C_k = U_k \Sigma_k V_k \tag{2.6}$$

The new matrix C_k has the following two characteristics:

- C_k ignores the smaller singular values so that some noise is removed from the original matrix C to some extent.
- The elements in the matrix C stand for the frequency of the word appearing in the document, so it must be a high order and sparse matrix and difficult to use directly. But due to the deletion of the smaller singular values, each position in the new matrix C_k almost has a value.

Based on the new features of C_k, if it multiplies its own transposition, the similarity matrix between the words will be gained. The similarity matrix is a mathematical representation of the semantic space, which can be applied to keywords expansion and text classification [22]. For example, the cosine value between two row vectors of C_k or two column vectors of $U_k\Sigma_k$ can reflect the similarity of the frequency of two words appearing in the text set. If the value is 1, the two words will have the same occurrence situation. On the contrary, if the value is 0, they will have completely different occurrence situation. If two documents are compared, the cosine value between two column vectors of C_k or two row vectors of $V_k\Sigma_k$ will be able to reflect the similarity between the two documents [22]. When a new document d is compared with the documents in the training set, a vector d' can be obtained by $d' = d^T U_k \Sigma_k^{-1}$. And then, the cosine value between the vector $d'\Sigma_k$ and $V_k\Sigma_k$ reflects the content similarity between the new document d and the documents in the training set.

In text classification, the training text set is usually required to expand. If every expansion needs to carry out the singular value decomposition for the whole training text set again, it will spend too much time. Especially when dealing with large-scale data set, it may not be able to work because of the limited storage space. So, a simple

method, which is called folding-in, can solve this problem [24]. For example, if a new text vector d is added, $d^{'}$ will be added to the column of V_k. If a new word vector w is added, w will be mapped to a new document vector space, namely the column of V_k, getting $w^{'} = w^T V_k \Sigma_k^{-1}$. And then, the word vector $w^{'}$ will be added to the column of U_k. Therefore, the method of folding-in needs less time and storage space than recalculating the whole training text set.

2.2.3.3 The Advantages of LSI

- LSI can be seen as an improvement of VSM, which brings in the concept of latent semantic [23]. It can realize the semantic retrieval to a certain extent, and eliminate the influence which is caused by synonyms and polysemes of some words.
- SVD in LSI is used to achieve the purpose of information filtering and removing noise. Meanwhile, matrix reduced-rank makes high-dimensional representation of the document in VSM map into the low-dimensional representation of the latent semantic space, reducing the scale of the problem greatly [22] [23].
- SVD in LSI only executes mathematical treatment to the matrix, which does not need grammar, semantics and other basic knowledge of natural language processing [20]. Basically, it is an inflexible and easy method.
- In the original matrix, a word only appears in a few documents so that many element values of the matrix will be zero. The data sparseness problem makes matrix manipulation quite difficult. But after SVD, the space dimension is reduced greatly, making the data sparseness problem get some improvement [21].

2.2.3.4 The Disadvantages of LSI

- The physical meaning SVD is not clearly defined. It's hard to control the effect of the classification and clustering of meaning of words [25].

- The process of the SVD algorithm cannot be controlled because its time, space and complexity are too large. Thus, it is difficult to deal with large-scaled data and real applications under the current operational capability [26].
- The updated matrix after the SVD algorithm has both positive and negative values, which means that the value of the similarity between words and matrix may be negative [25]. So, it is hard to present a definite physical meaning of the matrix. In addition, the uncertainty of the value of the similarity would bring some difficulties to further applications.
- The effect of solving synonyms and near-synonyms is not obvious although its dimension reduction effect is significant. The reason is that the process of dimension reduction is inflexible without the prior information [29]. Therefore, the final performance of text classification will be often damaged greatly [100].
- It is difficult to determine the value of K in the SVD algorithm. Generally, the value of K is often determined by empirical equations via comparing possible choices one by one [25] [26].

2.2.4 PLSI Model

2.2.4.1 Basic Concepts

The second major breakthrough of probability models was the Probabilistic Latent Semantic Indexing (PLSI) model, which was presented by Hofmann in 1990 [25] [26]. PLSI is used to simulate the words generation process, extending LSI to the framework of probability statistics. It redesigns the idea of generating models. It abandons the method of matrix transformation in LSI, but makes use of the generative model. Compared with the non-generative model, the generative model describes the reason of generating some probability density functions and the process of the interaction between the factors. But, the non-generative model ignores all the generative process, and presents the final result of various factors superposition directly.

For a complex construction process, the non-generative model often needs a certain idealized processing to obtain the result by mathematical derivation. The generative model tends to use the approximate calculation method to approximate the exact solution. Considered the computing performance, the non-generative model even sometimes can show an exact solution through mathematical equations, but the computing time and complexity are unacceptable because the computational equations often involve some hyper-functions, such as summation, products and continuous integration. Therefore, the models cannot be represented by the classical algebra.

PLSI is a probability model. Its main idea is to construct a semantic space where the dimension is not high. And then, all the words and documents are treated equally, and mapped to the semantic space. In this way, it not only solves the problem of high dimension, but also reflects the relationship between the words. For example, if the semantics of the words is much closer, the points corresponding to the words in the semantic space will be also closer. In the process of constructing the mapping, the PLSI model uses the iterative Expectation Maximization (EM) algorithm, making it more efficient [27] [28].

2.2.4.2 The Modeling Process

First of all, based on the probability principle, the relationship between the document d and the word w is established, which is the probability similarity between them and can be expressed as $P(d,w)$. Figure 2.4 is a schematic diagram of the PLSI model. The boxes are “plates” that represent replicates. The outer plate represents documents, and the inner plate represents the repeated choice of latent topics and words within a document. In addition, the filled circles represent the observed variable d and w, the hollow circle represents the latent variable z, and the black arrow represents the conditional dependency between two variables, namely the

conditional probability.

PLSI not only uses the idea of the generative model, but also reflects the internal relations between d and w. Besides, in order to compress the data, the latent multi-dimensional variable z is brought in, which can be understood as the semantic space or the topic space. The dimension of the space is a constant which is set manually. PLSI assumes that d and w are conditionally independent and the distribution of z on d or w is conditionally independent.

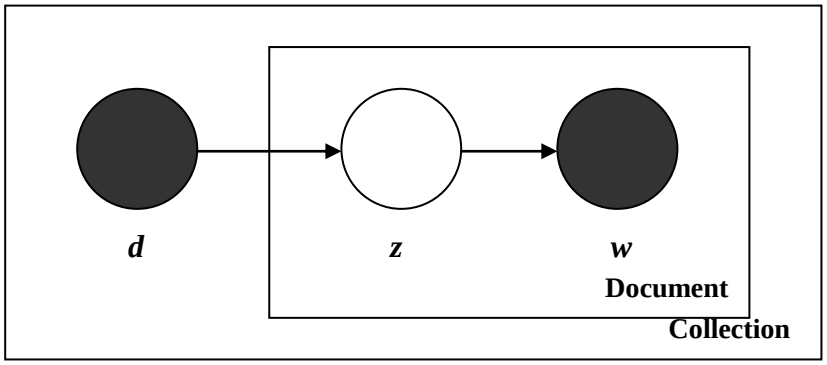

Figure 2.4: The graphical model representation of PLSI.

Secondly, according to the above assumptions and the conditional probability equation, the following equation can be obtained:

$$P(d,w)=\sum_{z\in Z}P(z)P(w\mid z)P(d\mid z) \tag{2.7}$$

where $P(d,w)$ represents the probability similarity between d and w, $P(z)$ means the latent semantic prior probability, $P(w\mid z)$ stands for the conditional probability of z on w, and $P(d\mid z)$ shows the conditional probability of z on d. In order to gain the above parameters in the unsupervised training, it is necessary to use EM algorithm to fit the latent semantic model [25] [28]. Different with the SVD, the decision function in PLSI is a more statistically significant minimum

entropy principle, which is to maximize the following function:

$$L = \sum_{d \in D} \sum_{w \in W} f(d, w) \log P(d, w) \tag{2.8}$$

where $f(d, w)$ means the initial value, which is the frequency of w appearing in d.

Thirdly, the following standard EM algorithm is used to optimize [25] [27] [28].

E-step: $$P(z \mid d, w) = \frac{P(z)P(d \mid z)P(w \mid z)}{\sum_{z' \in Z} P(z')P(d \mid z')P(w \mid z')}, \tag{2.9}$$

M-step: $$P(w \mid z) = \frac{\sum_{d} f(d, w)P(z \mid d, w)}{\sum_{d, w'} f(d, w')P(z \mid d, w')}, \tag{2.10}$$

$$P(d \mid z) = \frac{\sum_{w} f(d, w)P(z \mid d, w)}{\sum_{d', w} f(d', w)P(z \mid d', w)}, \tag{2.11}$$

$$P(z) = \frac{1}{R} \sum_{d, w} f(d, w)P(z \mid d, w), \tag{2.12}$$

$$R \equiv \sum_{d, w} f(d, w) \tag{2.13}$$

After the initialization with the random numbers, E step and M step are used alternately to carry out the iterative calculation. After many iterations, the probability similarity matrix $P(d, w)$ between d and w can be gained. Furthermore, the probability similarity matrix $P(w, w)$ between the words and $P(d, d)$ between the documents also can be calculated, which can be applied to information retrieval and text classification, etc.

For example, the output matrix $P(d, w)$ needs to multiply its own transposition, obtaining the probability similarity matrix $P(w, w)$.

$$P(w_1, w_2) = \sum_{d \in D} P(w_1 \mid d)P(d \mid w_2) \tag{2.14}$$

The core issue of text retrieval is to calculate the similarity between keywords and

documents. The queried keywords are constructed to the word vector w_q, and its dimension is equal to the row vector dimension of the matrix $P(w,w)$. The similarity between the queried keywords and the document d_n can be calculated as follow:

$$Similarity(w_q, d_n) = w_q \cdot P(w,w) \cdot w_n^T \quad (2.15)$$

In addition, the key issue of text classification is to calculate the similarity between the documents. Assume that the word vector w_o is extracted from the document d_o, and its dimension is equal to the row vector dimension of the matrix $P(w,w)$. Here, the elements of w_o are the normalized values of the frequency of words appearing in the document. Similarly, the word vector w_n of the document d_n can be gained. So, the similarity between d_o and d_n can be computed as below:

$$Similarity(d_o, d_n) = w_o \cdot P(w,w) \cdot w_n^T \quad (2.16)$$

2.2.4.3 The Advantages of PLSI

- The latent semantic space z of PLSI has a clear physical meaning, representing the latent topic. In addition, other probability values also have their corresponding physical meanings [25] [26].
- PLSI could solve the problem of synonyms and polysemes effectively, and use Expectation Maximization (EM) algorithm to train latent classes [27]. Compared with LSI, it has strong statistical basis.
- PLSI uses the EM algorithm to get solutions by iteration while computing model, reducing time complexity greatly and raising computing speed [28]. Thus, it's easier to achieve than SVD algorithm.

2.2.4.4 The Disadvantages of PLSI

- EM of PLSI is a completely unsupervised algorithm so its convergence is slow while the algorithm iterates [29].
- After getting the matrix $P(d,w)$ by the EM algorithm of PLSI, if there is a new batch of training data and the new updated matrix $P(d,w)$ is required, the only method is to iterate the algorithm from the beginning. Obviously, the efficiency is poor [39].
- The parameters space of PLSI is proportional to the training data of PLSI so that it is not good for modeling large-scale or dynamic growth corpus [40].
- The PLSI model needs to get a prior probability, which is only based on the existing training set. For the text outside of the training set, there is no suitable prior probability [29].
- Along with the increase of the training samples, the size of the matrix will increase linearly. Thus, PLSI has the problem of overfitting. The overfitting is that there are too many discrete characteristics in the PLSI model. Besides, these discrete characteristics are only suitable for the training text set but cannot describe other text which is outside of the training set [39] [40].

2.2.5 LDA Model

2.2.5.1 Basic Concepts

Aiming at addressing the above problems in PLSI, Blei et al put forward the Latent Dirichlet Allocation (LDA) model in 2003 [17] [30] [32]. Based on the PLSI model, LDA use a K-dimensional latent random variable which obeys the Dirichlet distribution to represent the topic mixture ratio of the document, which simulates the generation process of the document. At present, the LDA model is one of the most popular probability topic models, and it has more comprehensive assumptions of text generation than other models. As shown in Figure 2.5, in the process of text

generation, LDA uses sampling from the Dirichlet distribution to generate a text with the specific topic multinomial distribution, where the text is usually composed of some latent topics. And then, these topics are sampled repeatedly to generate each word for the document. Thus, the latent topics can be seen as the probability distribution of the words in the LDA model. And, each document is expressed as the random mixture of these latent topics according to the specific proportion.

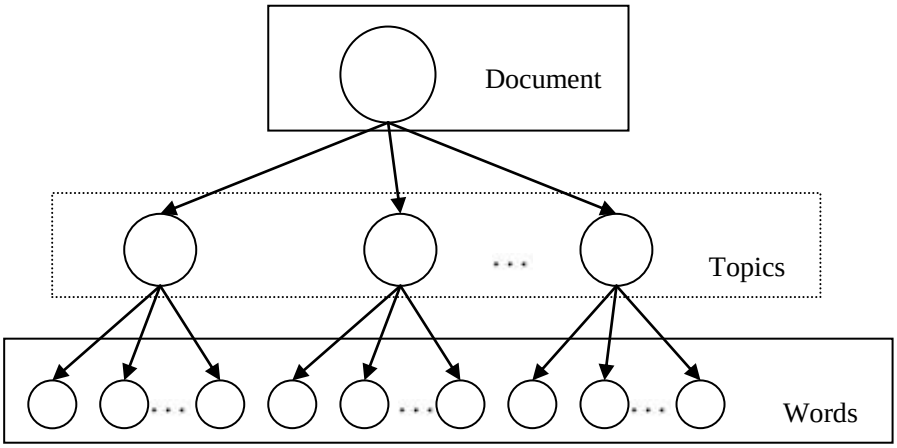

Figure 2.5: The network topology of LDA latent topics.

In addition, Girolami et al pointed out that the PLSI model can be seen as a special case of the LDA model when the Dirichlet distribution degenerates to unitary in 2003 [39]. The LDA model inherits all the advantages of the PLSI model without adding any idealized condition, describing a semantic model more factually. For example, in the selection of topics, PLSI needs to determine a text class label. According to the label, the topic distribution is generated, where the probability between the different topics is independent with each other. But, the Dirichlet distribution is used to generate a topic set in LDA. The topic set is not a series of independent parameters of PLSI, but it is represented by the latent random variables. So, LDA matches the semantic conditions better than other models in reality, and it has a stronger descriptive power. Compared with the PLSI model, the parameter space of LDA is

simple. Besides, the size of the parameter space has nothing to do with the number of training documents in LDA so that there is no overfitting situation. Therefore, the LDA model is a complete probability generative model [39] [40].

2.2.5.2 The Modeling Process

It is necessary to briefly explain that how the LDA model generates a document. For example, suppose that a corpus contains three topics, such as sports, technologies and movies. A document which describes the film production process may contain both the topic technologies and the topic movies. The topic technologies have a series of words that are related with technologies. And, these words have a probability, which means the probability of these words appearing in the document with the topic technologies. Similarly, there are a series of words that are related with movies in the topic movies, and there is a corresponding probability. When generating a document about film production, the topics will be chosen randomly first. The probability of choosing two topics of technologies and movies will be higher. And then, a word will be selected. The probability of choosing the words that are related with the two topics will be higher. So, the selection of a word is completed. After selecting N words continually, a document is created.

The LDA model is a probability generative model which models discrete data sets, is a three-tier Bayesian model, and is a method of modeling the topic information of documents [30] [32]. It describes documents briefly and keeps the essential statistical information, helping to process large-scale text set efficiently. As shown in Figure 2.6 (Left), the text set d like the top great circle can be divided into several latent topics $(t_1, t_2, \mathrm{L}, t_n)$ like the bottom small circles, and the topological structure of these latent topics is linear. Furthermore, to use probability inference algorithms can express a single document as the mixture of the specific proportion of these latent topics.

Figure 2.6 (Right) is the graphical model representation of LDA. The boxes are "plates" that represent replicates. The outer plate represents documents, which means that the topic distribution θ is sampled repeatedly from the Dirichlet distribution for each document in the document collection. And, the inner plate represents the repeated choice of topics and words within a document, which means that words in a document are generated by repeated sampling from the topic distribution. In addition, the hollow circles represent the latent variables α, β, θ and z, the filled circle represents the observed variable w, and the black arrow represents the conditional dependency between two variables, namely the conditional probability.

Furthermore, Figure 2.6 (Right) shows that the LDA model is a typical directed probability graph model with a clear hierarchical structure, such as the document collection-tier, the document-tier and the word-tier. The LDA model is determined by the parameters (α, β) of the document collection-tier, where α reflects the relationship between the latent topics in the document collection and β reflects the probability distribution of all the latent topics themselves. α and β are assumed to be sampled once in the process of generating a document collection. The random variable θ is the document-tier parameter, and its components stand for the proportion of various latent topics in the destination document. θ is sampled once per document. In addition, z and w are the word-tier parameters, where z represents the proportion of the destination document assigning the latent topics to each word and w is the representation of the word vector of the destination document. z and w are sampled once for each word in each document. In brief, for a document, its topic distribution θ is a Dirichlet prior function based on the parameter α. For a word w in a document, its topic z is generated by the topic distribution θ, and the word w is generated by the parameter β.

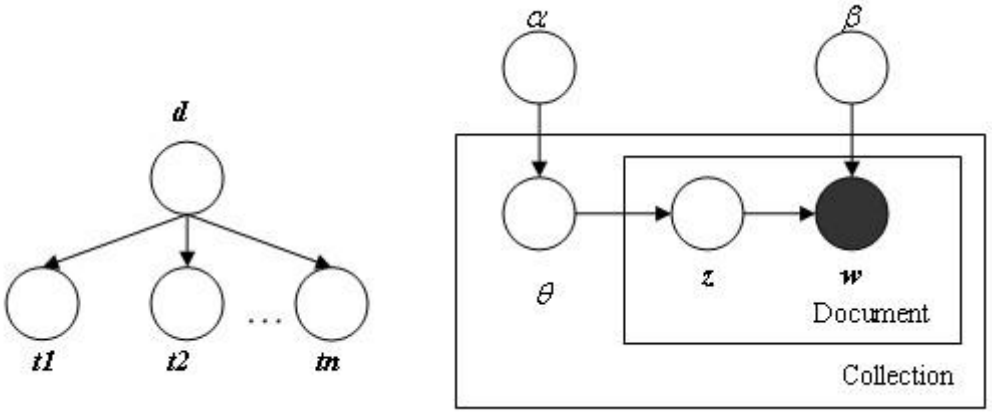

Figure 2.6: (Left) The structure diagram of LDA latent topics.
(Right) The graphical model representation of LDA.

Therefore, the process of LDA probability topic model generating a document is described as follows [30]:

Step 1: For the topic j, according to the Dirichlet distribution $Dir(\beta)$, a word multinomial distribution vector $\phi^{(j)}$ on j is obtained.

Step 2: According to the Poisson distribution $Poisson(\xi)$, the number of words N in the document is gained.

Step 3: According to the Dirichlet distribution $Dir(\alpha)$, a topic probability distribution vector θ of the document is got, where θ is a column vector and α is a parameter of the Dirichlet distribution.

Step 4: For a word w_n of N words in the document:

- A topic k is chosen randomly from a multinomial distribution Multinomial (θ) of θ.
- A word is selected from a multinomial conditional probability distribution

Multinomial ($\phi^{(k)}$) of the topic k, which can be seen as w_n.

If there are T latent topics, the probability of the i-th word w_i appearing in the document d can be expressed as follows:

$$P(w_i) = \sum_{j=1}^{T} P(w_i \mid z_i = j) P(z_i = j) \tag{2.17}$$

where w_i is the i-th feature word in the document d, j is the j-th latent topic and $z_i = j$ means that w_i takes the j-th latent topic. $P(w_i \mid z_i = j)$ is the probability of the word w_i belonging to the topic j. $P(z_i = j)$ is the probability of the topic j belonging to the document d. Let the i-th topic j be expressed as the multinomial distribution of w words, which can be represented as $\phi_w^{(z=j)} = P(w \mid z = j)$. The document d can be expressed as the random mixture of T latent topics like $\theta_{z=j}^{(d)} = P(z = j)$. So, the probability of the word w appearing in the document d can be shown as below:

$$P(w \mid d) = \sum_{j=1}^{T} \phi_w^{(z=j)} \cdot \theta_{z=j}^{(d)} \tag{2.18}$$

In order to make the model deal with new documents outside of the training corpus easily, the prior probability of the Dirichlet (α) in $\theta^{(d)}$ is assumed in the LDA model [30]. Similarly, the prior probability Dirichlet(β) in $\phi^{(z)}$ is supposed so as to infer the model parameters, as follows:

$$w_i \mid z_i, \phi^{(z_i)} \sim Discerte(\phi^{(z_i)}), \phi^{(z_i)} \sim Dirichlet(\beta) \tag{2.19}$$

$$z_i \mid \theta^{(d_i)} \sim Discerte(\theta^{(d_i)}), \theta^{(d_i)} \sim Dirichlet(\alpha) \tag{2.20}$$

where the extent of using topics and words is determined by the specific values of α and β, and different words and topics are used in the same way basically. Thus, it

can be assumed that all the values of α are the same and all the values of β are also the same, which can be called the symmetrical Dirichlet distribution [32].

The LDA model uses the Dirichlet distribution as the conjugate prior of the multinomial distribution ϕ and θ, simplifying the statistical derivation of the model [30]. For T-dimensional multinomial distribution $p = p(p_1, \cdots, p_T)$, the corresponding Dirichlet distribution probability density is as follows:

$$Dir(\alpha_1, \cdots, \alpha_T) = \frac{\Gamma(\sum_j \alpha_j)}{\prod_j \Gamma(\alpha_j)} \prod_{j=1}^{T} p_j^{\alpha_j - 1} \tag{2.21}$$

where the parameters $\alpha_1, \cdots, \alpha_T$ are called the hyper-parameter of the multinomial $p = p(p_1, \cdots, p_T)$. α_j is the priori observation of the frequency of topic j appearing in the document, which means that topic j has appeared α_j times before sampling any real word in the document.

In order to simplify the model, LDA uses the symmetrical Dirichlet distribution, namely let $\alpha_1 = \alpha_2 = \cdots = \alpha_T = \alpha$. Obviously, Dirichlet prior distribution can smooth the distribution of the multinomial $p = p(p_1, \cdots, p_T)$ by the hyper-parameter α, preventing the problem of overfitting to the data set. If the value of α is greater, the multinomial distribution will be more converged to the center. Therefore, via choosing an appropriate α, the problem of overfitting can be avoided [17]. However, if the value of α is too large, the obtained distribution will not reflect the real distribution of the data set.

As shown in Figure 2.6 (Right), the distribution of the hyper-parameters α and β can control the topic distribution θ and the word distribution on topics ϕ by the Dirichlet distribution $Dir(\alpha)$ and $Dir(\beta)$ [30]. And then, θ and ϕ codetermine

each word in the document. For the document d, the joint distribution of all the known and latent variables is as follows:

$$p(w_d, z_d, \theta_d, \phi \mid \alpha, \beta) = p(\phi \mid \beta)\prod_{n=1}^{N_d} p(w_{d,n} \mid \phi^{(z_{d,n})})p(z_{d,n} \mid \theta_d)\, p(\theta_d \mid \alpha) \quad (2.22)$$

After eliminating the variables θ_d, ϕ, z_d, the probability distribution of w_d is obtained as below:

$$p(w_d \mid \alpha, \beta) = \iint p(\theta_d \mid \alpha)\, p(\phi \mid \beta)\prod_{n=1}^{N_d} p(w_{d,n} \mid \theta_d, \phi^{(z_{d,n})})\, d\phi\, d\theta_d \quad (2.23)$$

Therefore, for the whole text set D, its corresponding probability distribution is as follows:

$$p(D \mid \alpha, \beta) = \prod_{d=1}^{|D|} p(w_d \mid \alpha, \beta) = \prod_{d=1}^{|D|} [\iint p(\theta_d \mid \alpha)\, p(\phi \mid \beta)\prod_{n=1}^{N_d} p(w_{d,n} \mid \theta_d, \phi^{(z_{d,n})})\, d\phi\, d\theta_d] \quad (2.24)$$

2.2.5.3 The Advantages of LDA

- The LDA model is the total probability generative model so that it has a clear internal structure, and can use efficient probability inference algorithms to calculate model parameters [30] [39].
- The size of the LDA model parameter space has nothing to do with the number of training documents. Therefore, it is more suitable for handling large-scale corpus [29].
- The LDA model brings in the hyper-parameter to document-topic tier, which is better than PLSI [40]. The priori information is added, which means that the parameters can be seen as the random variables. Besides, it makes LDA become a hierarchical model with more stable structure, avoiding the overfitting [39].

2.3 An Overview of the Main Inference Algorithms of LDA Model Parameters

According to the assumptions which were supposed by Blei et al [30], when LDA models the documents, the topic distribution parameter β and Dirichlet distribution parameter α are fixed in the generative process of text sets. So, for the document d, the posterior probability distribution of its latent variables is as follows:

$$p(\theta, z \mid w, \alpha, \beta) = \frac{p(\theta, z, w \mid \alpha, \beta)}{p(w \mid \alpha, \beta)} \tag{2.25}$$

In the above equation, the normalized item $p(w \mid \alpha, \beta)$ can be calculated by Equation (2.23). After bringing in $Dir(\alpha)$ and $Dir(\beta)$, a new equation can be obtained:

$$p(w \mid \alpha, \beta) = \frac{\Gamma(\sum_i \alpha_i)}{\prod_i \Gamma(\alpha_i)} \int (\prod_{i=1}^{T} \theta_i^{\alpha_i - 1})(\prod_{n=1}^{N_d} \sum_{i=1}^{T} \prod_{j=1}^{N} (\theta_i \beta_{ij})^{w_n^j}) d\theta \tag{2.26}$$

It can be seen that for a document d, the posterior probability distribution α and β of its latent variables θ and ϕ cannot be gained by calculation directly. Usually, there are three main inference methods to estimate the approximate value and infer the model parameters, such as Variational Inference (VI) [31] [32], Belief Propagation (BP）[106] and Gibbs Sampling (GS) [107] [108]. Finally, the latent topic structure of data and the important information of documents are obtained.

2.3.1 Variational Inference (VI)

Blei et al [30] defined a distribution $q(\theta, z \mid \gamma, \phi)$ and brought in the variational parameters γ and ϕ. And then, the variational parameters are used to rewrite the posterior probability distribution of the latent variables θ and z, to approximate the

real posterior probability distribution. The expression of the rewritten posterior distribution is as follows:

$$q(\theta, z \mid \gamma, \phi) = q(\theta \mid \gamma) \prod_{n=1}^{N_d} q(z_n \mid \phi_n) \tag{2.27}$$

where γ and ϕ are the parameters of q. In addition, VI converts an original complex graphical model like Figure 2.6 (Right) to a simple graphical model which is shown in Figure 2.7. Figure 2.7 is the graphical model representation of the variational distribution used to approximate the posterior in LDA. The boxes are "plates" that represent replicates. The outer plate represents documents, and the inner plate represents the repeated choice of topics within a document. In addition, the hollow circles represent the latent variables, where γ and ϕ are the variational parameters and θ and z are the latent variables. And, the black arrow represents the conditional dependency between two variables, namely the conditional probability. Assume that θ and z_n are independent with each other, the word variable w is removed [32].

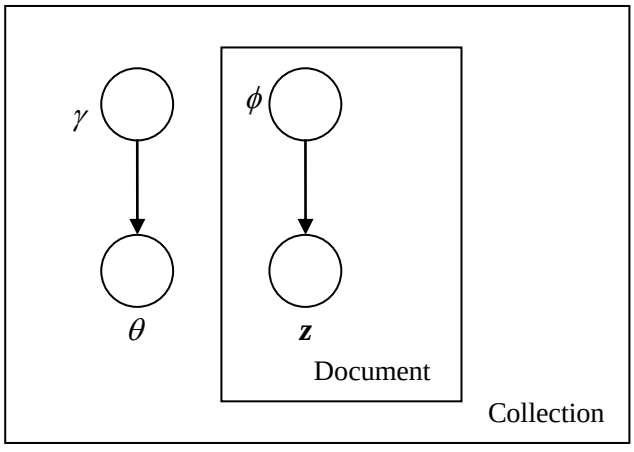

Figure 2.7: The LDA probabilistic graphical model with the variational parameters.

The whole approximation process can be seen as the process in which to find the

suitable variational parameters γ and ϕ makes the Kullback-Leibler Divergence (KLD) [109] between the estimated posteriori distribution and the real posteriori distribution minimum, which can be expressed as the following equation:

$$(\gamma^*, \phi^*) = \arg\min_{(\gamma,\phi)} D(q(\theta, z \mid \gamma, \phi) \parallel p(\theta, z \mid w, \alpha, \beta)) \tag{2.28}$$

where the definition of KLD is that for the discrete distribution P and Q, the KLD between them can be shown as below [110]:

$$D(P \parallel Q) = \sum_i P(i) \ln \frac{P(i)}{Q(i)} \tag{2.29}$$

The parameter estimation process uses Variational EM to make the lower bound of the logarithmic similarity maximized. The specific process can be divided into the following two steps:

E-step：

To minimize the variational parameters of KLD can be obtained by the fixed-point iteration method [105]. So, its updated equations are as follows:

$$\phi_{ni} \propto \beta_{iw_n} \exp\{E_q[\log(\theta_i) \mid \gamma]\}, \tag{2.30}$$

$$\gamma_i = \alpha_i + \sum_{n=1}^{N} \phi_{ni} \tag{2.31}$$

In Equation (2.30), $E_q[\log(\theta_i) \mid \gamma] = \Psi(\gamma_i) - \psi(\sum_{j=1}^{T} \gamma_j)$, where ψ stands for the first partial derivative of $\log\Gamma$ function. It can be gained by Taylor approximation [111].

For a document d, the optimal variational parameters $\{\gamma_d^*, \phi_d^* : d \in D\}$ can be found according to Equations (2.30) and (2.31).

M-step：

In order to estimate the parameters α and β in the LDA model, the parameters which can make the log-likelihood function of the document set maximized need to

be found [28]. For a document set D, its log-likelihood function is defined as follows:

$$l(\alpha,\beta)=\sum_{d=1}^{D}\log p(w_d\mid\alpha,\beta) \tag{2.32}$$

It is difficult to get $p(w_d\mid\alpha,\beta)$ by calculation directly, so the EM algorithm is used to find the parameters which can make the lower bound of the function in Equation (2.32) maximized as replace [111]. The parameter β can be obtained by lagrange multiplier method [112], which can be expressed as below:

$$\beta_{ij}\propto\sum_{d=1}^{D}\sum_{n=1}^{N_d}\phi_{dni}^{*}w_{dn}^{j} \tag{2.33}$$

where β_{ij} expresses the probability of the topic z^i on the word w^j, z^i means that the value of the topic variable z is i and w^j means that the value of the word variable w is j. D represents the number of documents, N_d expresses the number of words, ϕ_{dni}^{*} means the i-th optimizing variational parameter for each word in each document, w_{dn} is a word-level variable and is sampled once for each word in each document, and w_{dn}^{j} represents that the value of w_{dn} is j. In addition, the parameter α can be gained by the Newton-Raphson method [113].

In brief, according to the obtained approximate value in E-Step, the parameters α and β can be gained by maximizing the lower bound of the function in the VI algorithm.

2.3.2 Belief Propagation (BP)

Belief Propagation (BP) is an algorithm which uses the information transfer mode to infer graphical model parameters. It is an effective method of solving the conditional marginal probability. BP is applied to the graphical model containing variables and factors, localizing and distributing variables. In other words, it distributes the process of calculating global integral to the information transfer of each node, and each node is interacted with their adjacent nodes. According to BP algorithm, the calculated amount can be changed from exponential growth to approximate linear growth, making parameter inference methods apply to complex models.

In 2011, Jia Zeng first applied the BP algorithm to the parameter inference of the LDA model [106]. Figure 2.8 is a factor graph based on the LDA model. The model parameters θ_d and ϕ_w are called factors, which are represented by squares. Other variables which connect two factors are called topic labels, which are expressed by circles [114]. Thus, the factor θ_d connects the topic labels of all the documents in the text set, and the factor ϕ_w connects the topic labels of all the words in the word list. θ_d connects two latent variables $Z_{w,d}$ and $Z_{-w,d}$, where $Z_{-w,d}$ stands for the collection of all the words except for w in the document d being assigned the topic labels. So, θ_d connects the topic labels of different words in the same one document. In addition, ϕ_w connects two topic labels $Z_{w,d}$ and $Z_{w,-d}$, where $Z_{w,-d}$ means the collection of the topic labels of the same one word in all the documents except for the document d. So, ϕ_w connects the collection of the topic labels of the same one word in all the documents. The observational variables w and d in LDA classic graphical model are hidden in the factors θ_d and ϕ_w.

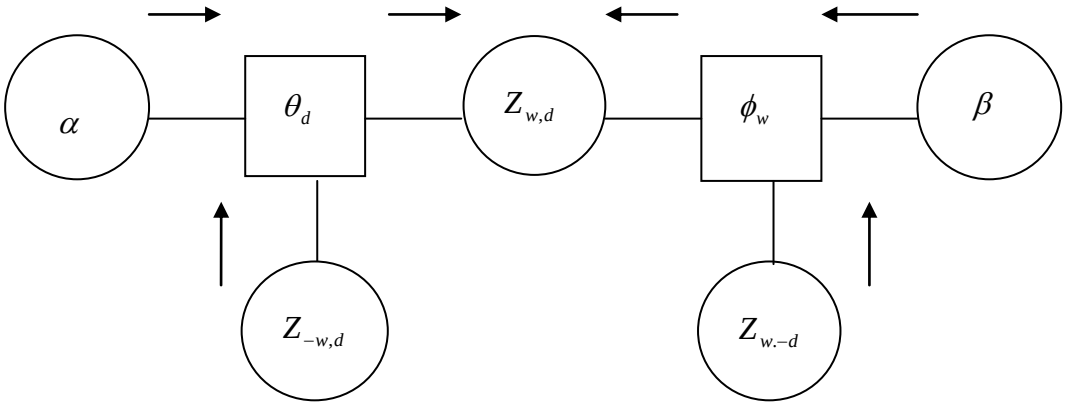

Figure 2.8: Belief propagation in the LDA model based on the factor graph.

As shown in Figure 2.8, the hollow circles represent the latent variables, but the squares represent two factors θ_d and ϕ_w. In addition, the black arrows stand for the direction of information transfer. When BP algorithm is used to estimate the LDA parameters, the topic labels information of words in documents will be determined by the information of the connected factors θ_d and ϕ_w. In other words, it will be determined by the information of different words in the same one document and the information of the same one word in different documents in the previous iteration process [106]. The main steps of the BP algorithm are described as follows:

Step 1: The parameters α, β, T, K and $\mu_{w,d}(k)$ are initialized, where α and β are the latent parameters in LDA, T represents the iterations of the algorithm convergence, K represents the number of topics, and $\mu_{w,d}(k)$ represents the information content.

Step 2: For all the documents in a text set, K information content is calculated from each word. And, the K information content is saved to update $\mu_{w,d}(k)$.

Step 3: The updated $\mu_{w,d}(k)$ is used to update the model parameters θ_d and ϕ_w. At the moment, $t = t+1$, where t represents the current number of iterations, and its initial value is zero.

Step 4: If $t \le T$, repeat step 2 and step 3. When $t > T$, the BP algorithm is converged and end.

Step 5: The final results of θ_d and ϕ_w are output.

The main equation of the information updating is as follows:

$$\mu_{-w,d}^{t+1}(k) \propto \frac{\mu_{w,-d}^{t}(k)+\beta}{\sum_{k}[\mu_{w,-d}^{t}(k)+\beta]} \frac{\mu_{-w,d}^{t}(k)+\alpha}{\sum_{k}[\mu_{-w,d}^{t}(k)+\alpha]} \tag{2.34}$$

In contrast, for the generated information at every time, the information in the BP algorithm is also transmitted from variables to factors [106]. According to Equations (2.35) and (2.36), the information can be transmitted to factors θ_d and ϕ_w retroactively. Therefore, it is a mutual process. After much iteration, the algorithm can achieve convergence finally, obtaining the parameters of the LDA model θ and ϕ as below:

$$\theta_d(k) \leftarrow \frac{[\mu_{\bullet,d}(k)+\alpha]}{\sum_{k}[\mu_{\bullet,d}(k)+\alpha]} \tag{2.35}$$

$$\phi_w(k) \leftarrow \frac{[\mu_{w,\bullet}(k)+\beta]}{\sum_{k}[\mu_{w,\bullet}(k)+\beta]} \tag{2.36}$$

2.3.3 Gibbs Sampling (GS)

Griffiths et al [82] proposed Gibbs sampling which was a kind of Markov Chain Monte Carlo (MCMC) methods, which is easy to implement and can extract topics from large-scale corpus very effectively. Besides, its perplexity and operating speed

are better than other algorithms. Thus, the Gibbs sampling algorithm has become the most popular extraction algorithm of the LDA model [111]. Gibbs sampling is a simple realization form of MCMC [115] [116]. Its basic idea is to structure the Markov chain which converges to a target probability distribution and extract the samples which are considered close to the values of the probability distribution from the chain by sampling from the posterior probability distribution. The key of using Gibbs sampling to infer LDA model parameters is that the model parameters are not obtained directly, but they are gained by sampling the topics of each word indirectly [107]. In other words, the values of model parameters θ and ϕ can be obtained indirectly by calculating the posterior probability. The details of Gibbs sampling algorithm will be introduced in Section 3.3.1 in Chapter 3.

2.3.4 Analysis and Discussion

The obtained model by variational inference has some difference with the real situation. And, EM algorithm often cannot find the optimal solution because of the local maximization problem of the likelihood function. Thus, its accuracy is worse than GS and BP. Besides, VI algorithm brings in the complex digamma function, making iteration and updating parameters need more time and the computational efficiency reduce greatly. BP algorithm uses the idea of information transfer to solve LDA modeling problems and keep all the information in the process of information transfer. Its accuracy is very high, but it has some problems such as taking too much memory and low efficiency. In addition, the studies showed that when processing large-scale real data, the model convergence of GS was faster than variational EM and BP [72]. The scalability of MCMC was better than variational EM and BP. Furthermore, many parallel LDA algorithms made use of GS [117]. Therefore, Gibbs sampling algorithm is used in this thesis, solving the parameters inference problem of the LDA model.

2.4 An Overview of Genetic Algorithm

2.4.1 The Basic Idea of Genetic Algorithm

Genetic Algorithm (GA) is a product which combines life sciences and engineering science, is a calculation model of the biological evolution process of simulating genetic selection and natural elimination. It was proposed by professor Holland at the University of Michigan in 1969, and he published an influential book which was called "Adaptation in Natural and Artificial Systems" [118]. After the generalization and summarization by Dejong and Goldberg et al, a new global optimization search algorithm was put forward [119]. In addition, GA was derived from Darwin's evolution theory, Weizmann's species selection theory and Mendel's population genetics theory. Holland not only designed the simulation and the operating principle of GA, using statistical decision theory to analyze the search mechanism of GA, but also established the famous schema theorem and implicit parallelism principle, laying the foundation for the development of GA [120] [121].

The basic idea of GA is to simulate breeding, crossing and mutation in the process of natural selection and natural genetic [118]. When dealing with the practical problem, every possible solution of the problem will be coded into a chromosome. A number of chromosomes constitute the population. Each chromosome can get a numerical valuation by the fitness function, eliminating the individuals with low fitness and choosing the individuals with high fitness to participate in the genetic operation. After crossover and mutation operators, these chromosomes will generate the next generation of new population. The performance of the new population of chromosomes is better than the previous generation because the new population inherits some excellent features from the previous generation. The optimal solution of the problem is found by generation after generation until the condition of convergence is met or the pre-set number of iterations is achieved [119] [121]. Therefore, GA can be seen as a kind of iterative algorithms, and a gradual evolution process of the

population which are composed by feasible solutions, which has a solid biological foundation [122].

2.4.2 The Main Steps of Genetic Algorithm

Since Holland systematically put forward the complete structure and theory of GA, many scholars proposed various kinds of improved genetic algorithms. The genetic algorithm which was proposed by Holland is usually called Simple Genetic Algorithm (SGA) [118] [122]. SGA mainly includes coding mechanism, fitness function, selection, crossover and mutation. In the practical application, the design of GA is considered from the above five basic factors. The specific details are as follows:

2.4.2.1 Coding Mechanism

The structure of many application problems is very complex, but they can be reduced to a simple coding representation of bit string form. The process of making the problem structure convert into the coding representation of bit string form is called coding. It is the basis of GA, which influences the performance of the algorithm greatly. The coding representation of bit string form is called chromosomes or individuals. The collection of strings constitutes the population, and individuals are strings. For example, in the optimization problem, a string corresponds to a possible solution. In the classification problem, a string can be interpreted as a rule. In SGA, the character set is composed of 0 and 1, and the coding method adopts binary coding. But for the general GA, they can be not limited. In addition, many scholars have done various improvements about the coding of GA, such as Gray coding, dynamic parameter coding, float-point coding, decimal coding, symbolic coding, multi-value coding, Delta coding, hybrid coding, DNA coding and so on [122].

2.4.2.2 Fitness Function

In order to reflect the adaptive capacity of chromosomes, a function which can measure every chromosome in the problem is brought in, which is called fitness function. When searching the possible solutions, GA basically does not use the external information. According to the fitness function, the values of the fitness of each individual in the population are used to search. The fitness function is the criterion which evaluates each individual in the population, reflecting the principle of survival of the fittest in natural evolution. For optimization problems, the fitness function is usually the objective function [121].

The fitness function should reflect the difference between each chromosome and the chromosome with the optimal solution of the problem effectively. If the difference is smaller, the difference between their respective values of the fitness function will be smaller, or it will be bigger. When using GA to solve specific problems, the choice of the fitness function will influence the convergence of the algorithm and the convergence rate greatly. Thus, for different problems, the related parameters are determined as a matter of experience [121].

2.4.2.3 Selection

Selection operation is also called copy operation, its role is to decide whether individuals are eliminated or inherited in the next generation according to the values of the fitness function of individuals in the population. In general, selection makes the individual with bigger value of the fitness have bigger chances of survival, but the individual with smaller value of the fitness have smaller chances of survival. Selection operator can ensure that excellent individuals transmit continuously in genetic operations, making the whole population evolve to an excellent population. SGA makes use of the roulette wheel selection mechanism. In addition, Potts et al summarized about 20 other selection methods, such as random traversal sampling

method, local selection method, championships selection method, steady state copy, optimal string copy, optimal string retention, fitness linear scale variation and so on [123].

2.4.2.4 Crossover

Crossover operation is to replace and recombine part of the structure of two parent individuals to generate new individuals. Its purpose is to compose new individuals, and search in the string space effectively meanwhile reduce the probability of failure to the effective model. Crossover operation is an important feature which can make GA differ from other evolutionary algorithms [122]. Various crossover operators consist of two basic contents [124]: firstly, in the formed population by selection operation, individuals will random pair. And, according to a preset crossover probability P_c, each pair of individuals will be determined that whether they need crossover operation or not. If the value of P_c is bigger, the probability of gene exchange will be higher, or it will be lower. Secondly, if the cross point is set, the part of the structure of the match pair of individuals before and after the point will interchange. In SGA, single point crossover is used. In addition, Potts et al generalized almost 17 crossover methods, such as two point crossover, uniform crossover, multipoint crossover, heuristic crossover, sequence crossover, mixed crossover and so on [123].

2.4.2.5 Mutation

Mutation operation is to change the values of some genes of individuals in the population randomly by a very small mutation probability P_m. Its aim is also to generate new individuals, overcoming the problem of premature convergence caused by losing effective genes. It includes determining the location of the change point and replacing genic values. In general, the importance of mutation operation ranks only

second to the crossover operation, so its role cannot be ignored [118]. In the SGA, for the binary coding, the strings of individuals in the population are selected by P_m, and these genes will be changed. For example, if the value is 1, it will be changed to 0. Or if the value is 0, it will be changed to 1. Furthermore, Potts et al summarized three mutation technologies including management mutation, changed mutation probability and single valued operation. Besides, they also summed up a variety of mutation operations, such as basic bit mutation, uniform mutation, Gaussian mutation, non-uniform mutation, adaptive mutation, multi-level mutation, and so on [123].

2.4.3 The Parameter Settings of Genetic Algorithms

Generally, the parameter settings of GAs contain population size, crossover probability and mutation probability, etc, which can influence accuracy, reliability, computing time and system performance of GAs [118]. At present, the reasonable selection of the parameter settings of GAs lacks relevant theory as the guidance so that the parameters are usually set according to the experience or experiments [121].

The total number of chromosomes in a population is called the population size. If the population size is too small, the search space of GAs will be limited and it is difficult to find out the optimal solution. On the contrary, if it is too big, the convergence time will increase and the program running time will rise [118]. Thus, different problems may have their own appropriate population sizes. Usually, the population size is set from 20 to 200 [125].

In SGA, crossover probability P_c and mutation probability P_m are fixed, which are determined by the experience [122]. When a population size is set from 20 to 200, Schaffer suggested that the optimal parameter value range of P_c would be set between 0.5 and 0.95 [122]. If the value of P_c is bigger, the speed of generating new

individuals will be faster. However, if it is too large, the probability of damaging the genetic model will be greater. If it is too small, the search process will be slower even stagnant. In addition, the value of P_m is usually set between 0.01 and 0.05 [122]. If the value of P_m is too small, it will be difficult to generate new genetic structures. But if it is too large, GAs will become simple algorithms of random search.

2.5 An Overview of Hadoop

Hadoop [196] was a distributed computing open source platform, which was developed by Doug Cutting. By using a simple programming model in a computer cluster, distributed application programs can be written and carried out to process large-scale data [60] [61] [62]. It originated in a subproject Nutch in Apache Lucene, which accomplished MapReduce [66] algorithm and designed its own Hadoop Distributed File System (HDFS) according to Google GFS [54]. Hadoop is mainly composed of HDFS and the distributed computing framework MapReduce. The former is the open source implementation of Google GFS. The latter is the open source implementation of Google MapReduce. Furthermore, Hadoop also includes other subprojects, such as Avro, Core, Zookeeper, HBase, Hive, Pig, ChuKwa [60].

Both of HDFS and MapReduce of Hadoop use master/slave architecture [64]. Masternode is composed of Namenode and Jobtracker, and provides some tools and opens the browser view function of the Hadoop status. Besides, Slave node is made up of Datanode and Tasktracker. Usually, the master/slave architecture has one masternode and several slavenodes. The masternode is a manager or controller, and the slavenode accepts the control of the masternode. In brief, the master/slave architecture of Hadoop is mainly reflected in two key technologies: HDFS and MapReduce [68]. Namenode and Datanode are responsible for completing jobs of HDFS, and Jobtracker and Tasktracker are responsible for finishing jobs of MapReduce.

2.5.1 HDFS

HDFS is mainly used to store all the data in the cluster nodes, and it can achieve the growth of storage capacity and computing power by only adding the number of computers. It is developed by Java language so that it can run in many low-cost computers and has strong fault tolerance. Thus, it provides basement support for distributed computing and storage [60] [62].

A HDFS cluster is often made up of one Namenode and a certain number of Datanodes. Usually, each node is an ordinary computer [65]. The Namenode is a central server, which manages the Namespace of the file system and the client accessing to files. The Datanode manages the storage in local node. The data of users are divided into lots of data blocks which are stored in different Datanodes dispersedly. Datanodes would report the listing of blocks to the Namenode periodically. Both of Namenode and Datanode are designed to run in normal low-cost computers with Linux [67].

The Namenode is the core of the whole HDFS, which carries out the Namespace operations in the file system, such as open, close, rename a file or directory. And, it is also responsible for determining that data blocks map to specific Datanodes [63]. The Datanode is responsible for dealing with the read-write requests of the client in the file system, and executing create, delete and copy to data blocks under the unified scheduling of the Namenode. In a given period, the Datanode gives the Namenode heartbeat reports. And then, the Namenode controls Datanodes by the heartbeat reports, and tests health status of Datanodes continuously.

For users, they only need to use the Namespace which is provided by HDFS to access and modify files, but they do not need to know that which Datanode does read-write the data and which Datanode stores the backup data [65]. When users carry out file operations in HDFS, the effect will be the same with file operations in a stand-alone

system. For example, in a typical cluster, a computer runs only one Namenode, and other computers run one Datanode respectively [65].

2.5.2 MapReduce Programming Model in Hadoop

The framework of MapReduce in Hadoop includes one Jobtracker and a certain number of Tasktrackers [62] [68], which is shown in Figure 2.9. The Jobtracker often runs on the computer which has the Namenode. In a cluster, each of other nodes runs one Tasktracker. The Jobclient is responsible for submitting the user-defined Mapper class and Reducer class to the system, and setting various system parameters. It sends MapReduce jobs to Jobtrackers. After accepting the task, Jobtrackers are responsible for the task scheduling of work nodes, monitoring them periodically and handling failed tasks. The Tasktracker is responsible for reporting the process information to Jobtrackers periodically and executing specific Map/Reduce operations [64].

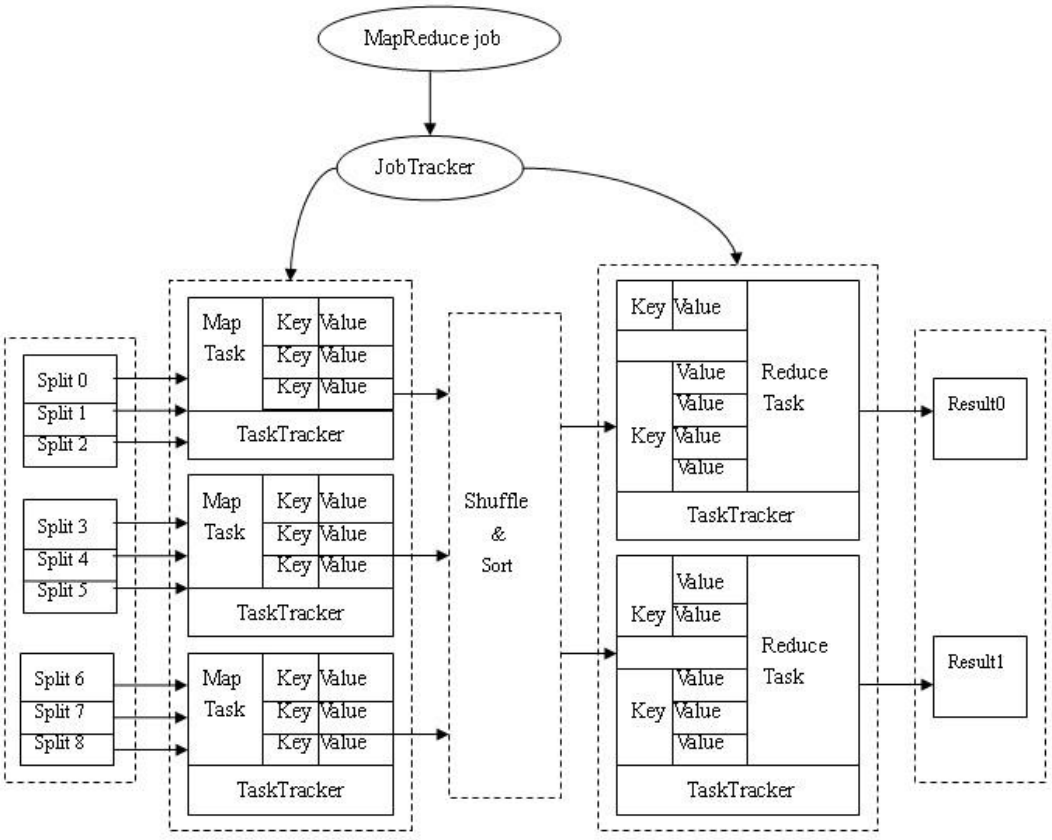

Figure 2.9: The typical MapReduce framework in Hadoop.

The programming framework of MapReduce divides a task into two phases, Map phase and Reduce phase. And, the processed data structures in two phases are (key, value) pairs [52] [68]. Users only need to define their own Map function and Reduce function respectively, and then give them to Hadoop. But, they do not need to care about complex basement details, such as task distributing, concurrency control, resource management and fault tolerance.

Map phase [65] [66]: firstly, the framework of MapReduce divides the input data into many data blocks which are independent with each other so that Map operations can achieve concurrent execution completely. And then, each block will be assigned to different Map tasks. Each Map task will read-in the assigned blocks in the form of (k1, v1). After processing by Map function, intermediate results are generated. Finally, intermediate results will be stored in local intermediate files. Here, intermediate results are output in the form of (k2, v2).

Reduce phase [65] [66]: the number of Reducer can be designed by users, and it can be one or more than one. In the Reduce process, because its input is from the intermediate results which are generated by Map tasks that distribute in multiple nodes, the intermediate results need to be copied in the local file system of the nodes with Reduce tasks by the network transmission way. In the output result from Mappers, Hadoop makes values with all the same keys store in an iterator and transmit to Reducers. After Reducers receiving the input, values with all the same keys will be merged. At the end, the merged result will be output into the specified output file.

2.5.3 Hadoop Scheduling Algorithms

The job scheduling in Hadoop is a process of the Jobtracker assigning tasks to the corresponding Tasktrackers to run [66]. When multiple tasks running at the same time,

the sequence of job execution and the allocation of computing resource will affect the overall performance of the Hadoop platform and the utilization rate of the system resources. The scheduler is a pluggable module. Users can design the scheduler according to their practical application requirements, and then specify the corresponding scheduler in configuration files. When a Hadoop cluster starts, the scheduler will be loaded. At present, Hadoop has several schedulers, such as First In First Out (FIFO), fair scheduler and capacity scheduler [60] [70].

The FIFO scheduler is a Hadoop default scheduler. Its basic idea is to put all the MapReduce jobs into the only one queue. And then, the Jobtracker will scan the whole job queue by the priority sequence or the submission time sequence. Finally, a job which meets the requirement is chosen to run [60]. Fair scheduler was first proposed by the Facebook. Its purpose is to make the MapReduce programming model in a Hadoop cluster deal with the parallel processing of different types of jobs, and guarantee that each job can get equal resources as far as possible. Capacity scheduler was put forward by Yahoo, and its functions are similar to fair scheduling. But, there are many differences in their design and implementation [52].

FIFO is simple and clear, but it has large limitation. For example, it ignores the differences of operational requirements. The last task in the scheduling queue has to wait for that all previous tasks complete job executions, influencing the overall performance of the Hadoop platform and the utilization ratio of the system resources seriously. Fair scheduler has some improvement, but it cannot support large memory operations. Compared with FIFO, capacity scheduler is able to optimize the resource allocation effectively, enhancing the operating efficiency of jobs. However, it must configure lots of parameter values manually, which is quite difficult to normal users who do not understand the overall situation of the cluster and the system [68] [70].

In addition, [97] pointed out the disadvantages of existing schedulers in Hadoop. Hadoop scheduling mechanism is based on the following assumptions [97]:

- All the nodes run by the same speed.
- In the whole operational process, tasks run by the constant speed.
- There is no overhead when starting backup tasks.
- The task progress can represent the work which has been done. In the Reduce phase, copying, sorting and merging of tasks take up one third of the whole completion time respectively.

Assumed conditions 1 and 2 are based on a homogeneous environment, but they are not effective in a heterogeneous environment. So, existing schedulers in Hadoop have not considered the differences of the processing speed between different nodes in a heterogeneous environment. Besides, assumed conditions 3 and 4 are even invalid in a homogeneous environment. [97] pointed out reasons that these assumptions are invalid one by one. To sum up, existing schedulers in Hadoop are suitable for a homogeneous environment. But in a heterogeneous environment, they will waste a lot of resources and spend much overhead, affecting the overall performance of the Hadoop platform.

2.5.4 Divisible Load Theory

Load scheduling including the division and the transmission of the load is one of key factors which influence the performance of parallel and distributed computing [126]. Load scheduling problem can be divided into two categories: non-divisible load scheduling and divisible load scheduling. Some parts of non-divisible load jobs cannot be divided, but they only can be processed by a processor as a whole [130]. For applications, basic assumptions in the divisible load theory are that applications are able to be divided into any size of subtasks. They are independent with each other, and they can run the parallel execution completely. Thus, these applications are called divisible load [127] [128] [129].

An important feature of divisible load scheduling is that data communication only occurs in two phases which are before the start of the calculation and the end of the calculation. Its aim is to assign a given application to each processor, making the completion time of the whole application the shortest [131] [132]. In addition, the divisible load theory provides a useful framework for the divisible load scheduling in a heterogeneous environment. For a given task, if processing nodes and the network heterogeneity are considered fully, the divisible load theory will provide an efficient task division mode and scheduling mode [133] [134] [135].

In Hadoop, the features of the PLDA algorithm include that firstly, the input data can be divided into any size of sub-data. Secondly, the data communication only occurs in two phases which are data distribution phase and returning results from processors phase. It can be seen that the PLDA algorithm in Hadoop is completely in conformity with assumptions of the divisible load theory.

In conclusion, considered that various nodes in a heterogeneous cluster have different computing speed, different communication capability and different storage capacity, a static load balancing strategy based on genetic algorithm for PLDA will be proposed in Chapter 5 in this thesis. In this allocation strategy, because it combines with the optimal principle of the divisible load theory, all the data nodes should complete local Gibbs sampling at the same time theoretically, making the completion time of parallel LDA minimum and improving the performance of Hadoop clusters in a heterogeneous environment.

2.6 Summary

At the beginning, this Chapter introduced the basic concepts and the modeling process of LSI, PLSI and LDA, and analyzed and compared their advantages and disadvantages. LDA topic model is a completely probability generative model, which

can use mature probability algorithms to train models directly. And, it is easy to make use of the model. The size of its parameter space is fixed, which has nothing to do with the scale of text sets. So, it is more suitable for large-scale text sets. Besides, LDA is a hierarchical model, which is more stable than the non-hierarchical model. By describing and comparing main inference algorithms of LDA model parameters like variational inference, belief propagation and Gibbs sampling, the research work of this thesis will be focused on the LDA model with Gibbs sampling algorithm.

And then, basic concepts, main steps and parameter settings of genetic algorithm were introduced. Next, the related concepts of Hadoop were presented, which containing HDFS, MapReduce programming model and Hadoop scheduling algorithms. At present, Hadoop existing schedulers are suitable for a homogeneous environment. But in a heterogeneous environment, they will waste lots of resources and spend much overhead, which affects the overall performance of the Hadoop platform. Finally, the divisible load theory was recommended as a theoretical basis of the proposed static load balancing strategy in Chapter 5.

Chapter 3 Text Classification with Latent Dirichlet Allocation

3.1 Introduction

Automatic text classification is a research hotspot and core technology in the field of information retrieval and text mining. Recently, it has got widespread concern and made remarkable progress. It is one of the hotspots and key technologies of information retrieval, machine learning and natural language processing [75] [76]. Its aim is to find classification rules from the known training text set, obtain a learner, and make the learner classify the unknown new text with good prediction accuracy [136]. This chapter begins with an overview of several key technologies of text classification. Then, a text classification method based on the LDA model is proposed, which uses LDA topic model as a text representation method. Each text is represented as a probability distribution on a fixed latent topic set. SVM is chosen as a classification algorithm. Finally, the proposed method shows a good classification performance, and its accuracy is higher than other two methods.

3.2 Overview of Text Classification

Text classification is a process in which a category label is added automatically or manually into the given text according to its content. By classifying text, users can not only find needed information more accurately, but also can browse information quite conveniently [13] [14]. Automatic text classification is a supervised learning process. According to the given training document collection, a relational model between document feature and document categories is built by some method. And then, the relational model is used to classify the given unknown documents [76].

From the view of mathematics, text classification is a mapping process, which maps the unclassified text to an existing categorization [75]. This mapping can be a one-to-one mapping, or a one-to-many mapping, because a document can be associated with multiple categories. The mapping rule of text classification is that the system summarizes the classified rule and establishes discrimination equations and discrimination rules according to information of several samples in each existing category. The system determines their classes when dealing with new documents based on the discrimination rules.

Overall, automatic text classification consists of five parts as shown in Figure 3.1: text preprocessing, text representation model, text feature extraction and dimension reduction, text classification model, and classification performance evaluation [75] [137]. In this chapter, the Vector Space Model (VSM) [23] [138] [139] is chosen as text representation, selecting words as the feature, making text collection structure a high-dimensional and sparse word-document matrix.

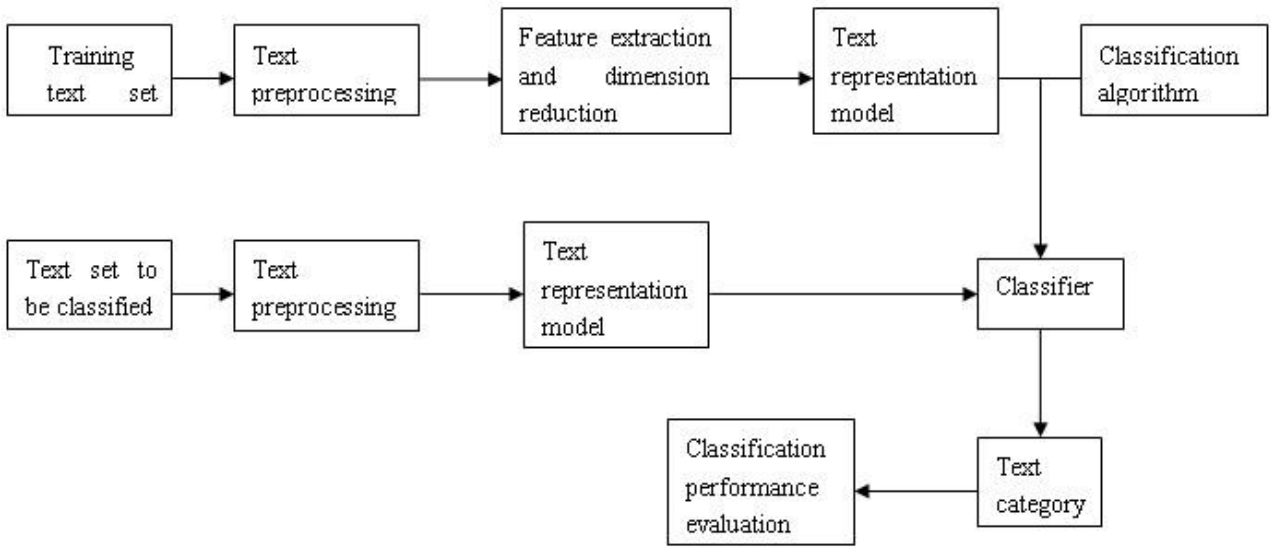

Figure 3.1: The typical process of automatic text classification.

Dimensionality reduction of word-document matrix is beneficial to improve the efficiency and performance of the classifier. Therefore, one of the most typical probabilistic topic models, Latent Semantic Indexing (LSI), is used to map data from the original high-dimensional space to a new low-dimensional space by space transformation [20] [21] [22]. It is a method which can explore latent semantic relation among words according to the word co-occurrence information. Its basic principle is to use the SVD technique in matrix theory to transform a word frequency matrix into a singular matrix. It only keeps K maximum values from the singular value vector, mapping text vector and query vector from the word space to a K-dimensional semantic space. The dimension reduction effect of LSI in text classification is remarkable, but it may filter out very important features for rare category in the whole document collection, leading to impact on the final classification performance [24] [98]. Moreover, the complexity of the algorithm implementation is another problem of the LSI model, which cannot be ignored [140].

Therefore, this chapter presents a text classification method based on LDA. LDA is a probabilistic growth model, which models discrete data sets such as text collection. It is a three-tier Bayesian model, containing words, topics and documents three-tier structure [30] [31]. Each document is expressed as a mixture of topics, and each topic is a multinomial distribution on a fixed vocabulary. These topics are shared by all documents in the collection. Each document is generated by sampling a specific topic proportion from the Dirichiet distribution. SVM is chosen as classification model, because its separated surface model can overcome sample distribution, redundant features, over-fitting and other factors effectively [141] [142] [143]. In addition, the study showed that SVM has the advantages of efficiency and stability compared with Naive Bayes, linear classification, decision trees, KNN and other classification methods [144] [145] [146].

3.2.1 The Content of Text Classification

Text classification is a process which classifies a large amount of documents into one or more categories on the basis of document's content or properties. The content can be media news, technology reports, emails, technology patents, web pages, books or part of it [7] [13]. Text classification concerns text categories including topics of text (such as sports, politics, economy, arts, etc.), the style of text, or the relationship between text and other things. If documents are classified manually, people need to read through all the articles, then classify and keep them. It requires many experts with rich experience and specialized knowledge to do lots of work. Obviously, this process has some disadvantages such as long cycle, high cost and low efficiency. It is difficult to meet the real needs [75] [147]. Thus, this chapter focuses on how text can be classified automatically.

3.2.2 Text Preprocessing

The purpose of text preprocessing is to structure the unstructured text, and keep the key words which are useful to distinguish the categories of text topics [148]. Its process includes text representation, removing function words and content words that are not useful to represent category. Generally, natural language text contains a large number of function words which have no specific meaning, such as prepositions, articles, conjunctions, pronouns, etc. Besides, there are some common content words which have very high frequency in almost all of the text, can not distinguish text, and even interfere with keywords, reducing the processing efficiency and accuracy of the classification system. So they should be filtered out.

As a result, removing the non-feature words can be done through the following two steps [149]: Firstly, according to a word segmentation system, part of speech is tagged, and then all the function words are deleted, such as articles, prepositions, adverbs,

modal verbs, conjunctions and pronouns, etc. Secondly, a stop words list is established, and then those content words which have high frequency in all categories are put in that list. The words and phrases in the stop words list are filtered out while analyzing the text. After the above two steps, the text becomes a sequence of feature words. For each document, the frequency of occurrence of each word is counted.

3.2.3 Text Representation

Text representation is to use a certain feature words to represent the document [150]. In text classification, only the feature words need to be processed in order to accomplish the processing of the unstructured text, which is a converted step from unstructured text to structured text. Therefore, establishing a text representation is an important technology, relating to how documents are applied to the classification system. As we know, the computer does not have the human intelligence. After reading the article, people can have fuzzy understanding for its content based on their own comprehensive ability. But the computer can not really understand the article. Hence, the text itself can not be directly used for classification. It must be represented as a mathematical model to facilitate machine processing [11].

In practical application, the related text vector or word frequency matrix is very large, and the processing of text information can not be completed because of the dependence between words. So generally, it assumes that words are mutually independent in the process in order to reduce the complexity of text information processing [151]. Although this assumption is inconsistent with practical situation, the complexity of computing is greatly reduced for text classification and the classification performance has also been improved significantly. Therefore, many classification algorithms are based on this assumption as a prerequisite. This method is called the “bags of words” model. Counting the frequency of occurrence of each word in each document is the basis for modeling algorithms. Counting the frequency

of occurrence of all words in all documents can produce the word frequency matrix. Text information based on natural language that is difficult to use numbers to describe is expressed as a mathematical matrix form, which can handle text according to the matrix operations [152].

This thesis uses the vector space model to represent text as a vector that is made up of feature words. VSM has good computability and operability, is a text representation method with good performance in recent years, and is one of the most classic models at present [138] [139]. It assumes that the categories of documents are often only related to the frequency of occurrence of certain specific words or phrases in the document, but have nothing to do with their location or sequence in the document. It ignores the complex relationship among paragraphs, sentences and words in the text structure. A document can be expressed as a multi-dimensional vector form that is composed of feature words [23].

For example, documents and the whole feature set compose a matrix $A_{mn} = (a_{ij})$ together, in which the value of a_{ij} stands for the weights of the i-th feature word $(0 \le i \le m-1)$ in the j-th document $(0 \le j \le n-1)$. Each row represents a feature word and each column represents a document after feature selection. Generally, the TP-IDF is used to do weight calculation [103] [104]. After calculating the weights, the normalization processing is needed. Equation (3.1) or (3.2) represents the final weight calculation.

$$W(t,\vec{d}) = \frac{tf(t,\vec{d}) \times \log(N/n_t + 0.01)}{\sqrt{\sum_{i \in \vec{d}} [tf(i,\vec{d}) \times \log(N/n_i + 1)]^2}} \tag{3.1}$$

$$W(t,\vec{d}) = \frac{(1 + \log_2 tf(t,\vec{d})) \times \log_2(N/n_t + 0.01)}{\sqrt{\sum_{i \in \vec{d}} [(1 + \log_2 tf(i,\vec{d})) \times \log_2(N/n_i)]^2}} \tag{3.2}$$

where $W(t,\vec{d})$ is the weight of word t in document $\vec{d}$. $tf(t,\vec{d})$ is the word frequency

of word t in document $\vec{d}$. N is the total number of training samples. n_t is the number of times of t appearing in the training text set. The denominator is the normalization factor.

3.2.4 Text Feature Extraction and Dimension Reduction

For an ordinary text, the word frequency information is extracted as the classification feature [81] [154] [155]. However, the number of text words is quite large, hence dimensionality reduction is needed [80] [81] [153]. There are two main reasons: Firstly, to increase the speed and improve the efficiency of classification. Secondly, the values of different words for classification are different. For instance, the contribution of some common words in each of the categories to text classification is small, but the contribution of some words which have a major proportion in certain specific class and a minor proportion in other classes to text classification is large. In order to improve the classification accuracy, those weak expressive words should be removed or weakened for each class, screening out the feature set for the class, so as to highlight the effective features of classification.

For example, the number of feature words is often able to reach 10^4 or more, but in fact real key feature words are just a small part of them. So, the really needed feature subspace is required to be found from the huge feature space. This thesis uses a wrapper approach to reduce the space [154]. It maps data from the original high-dimensional space to a new low-dimensional space by space transformation. The value of each dimension in the low-dimensional space is obtained by a linear or nonlinear transformation from the original space. It uses the linear or nonlinear transformation to make redundant or invalid information map to relatively weak dimensions, so as to ensure the effect of feature extraction [155].

In brief, the basic idea of feature extraction and dimensionality reduction is as follows [80] [81] [155]:

- Initially, the feature set includes all the words appeared in the document.
- Vectorizing all training documents constitutes a vector matrix.
- Using a probabilistic topic model makes the vector matrix transform into a small matrix. At the moment, the space where the new matrix is new feature space. After that, it is trained by the classifier.
- To be classified documents are transformed into the vector of original feature space. Then, the vector is transformed into the vector of new feature space by the corresponding conversion method. At this time, this vector of the new feature space is the final vector.

Taking the LSI model as an example, the detailed description is as follows: it assumes that D is the initial training set of documents. $t \times d$ represents the document-word matrix, which is the matrix in the original feature space. Each row of the matrix stands for a document vector. And it assumes that r is the rank of D. D can be expressed as $D = U\Sigma V^T$ through SVD, where Σ is a $r \times r$ matrix. Each value in the matrix corresponds to the singular value of D. U is a $t \times r$ matrix composed of the left singular column vector, V is a $d \times r$ matrix composed of the right singular column vector. The k largest singular values and their corresponding singular value vector are only retained, obtaining an approximate D_k which can be represented as $D_k = U_k \Sigma_k V_k^T$. This value of D_k is the optimal k approximation of D. At this moment, U_k that is a $t \times k$ matrix can map the column vector d of $t \times l$ matrix to k-dimensional vectors: $d_k = U_k^T d$. The vectors obtained by the above conversion can be used to training and classification [20] [22].

3.2.5 Text Classification Algorithms

The core issue of text classification is that how a classification function or classification model can be constructed from a large number of labeled training samples according to certain strategy, and how this model can be used to map the unknown class text to the specified class space. At present, common classification methods based on VSM include the Naïve Bayesian algorithm, K-Nearest Neighbor (KNN), SVM, and so on [75] [76]. There are other methods based on machine learning such as decision trees and neural network methods. They have been widely applied to information retrieval, text classification, knowledge management, expert systems and other areas. In terms of the accuracy of algorithms, SVM is the best, but it needs long time and large computation overhead [145]. KNN is second, but its computational time increases linearly when the training set grows. The Neural network classification algorithm is better than the Naïve Bayesian algorithm in recall and precision. The Naïve Bayesian algorithm has a strong theoretical background, and its computing speed is the fastest [146].

Vapnik et al. proposed SVM, a method of machine learning according to the statistical theory in 1995 [156]. Joachims introduced SVM into text classification, which can make a text classification problem transform into a series of second-class classification problems [141]. Its basic principle is that the vector space is divided into two disjoint spaces first. The features in the feature space become flat on two sides of the plane by constructing a hyperplane. The feature points of two sides of the plane belong to different classes so that the points of the space can be assigned to two different classes [144].

When SVM is used for text classification, a vector space for SVM is constructed first, which means that all training texts are mapped to a vector space. Then, a hyperplane which can classify the training text correctly or approximately correctly is built in this space. This hyperplane is needed to satisfy classification features of the original

training text as far as possible. Next, the text to be classified is mapped to this vector space by the same procedure. Finally, to estimate the relationship between the mapped vector and the hyperplane can determine which class the text to be classified belongs to [142] [145].

Figure 3.2 shows the separating hyperplane of SVM algorithm, where circle and triangle respectively stand for two classes of text. O1 is the classification line. H1 and H2 are respectively the samples which are the closest to the classification line in two classes, and are straight lines which paralleling to the classification line. The distance between them is called classification interval. The optimal classification line not only can separate two classes correctly, but also can make the classification interval the largest. In Figure 3.2, the sample points in H1 and H2 are called the support vector.

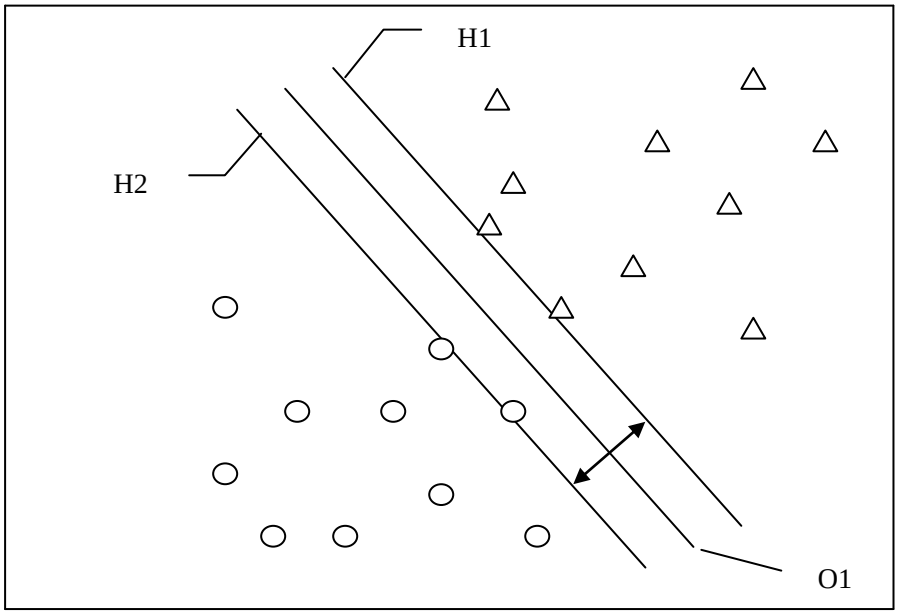

Figure 3.2: The separating hyperplane of SVM algorithm.

Suppose a given set of training data $T = (x_i, y_i)$, where $i = 1, \cdots, n$, $x_i \in R^d$ is the

feature vector of the i-th sample, $y_i \in \{+1,-1\}$ is the category tag of the i-th sample. A real-valued function is found on $x_i \in X = R^d$ so that $f(x) = \text{sgn}(g(x))$. When g(x) is a linear function, $g(x) = (wx) + b$, and function $f(x)$ determines classification rules, which is called a linear classification learning machine. When g(x) is a nonlinear function, it is called a nonlinear classification learning machine [141].

For the linear classification problem, the general form of linear discriminant function in a d-dimensional space is $g(x) = wx + b$, the hyperplane equation is $wx + b = 0$. After normalizing the discriminant function, all samples satisfy $|g(x) \geq 1|$. If the hyperplane can classify all samples correctly, it should satisfy the following equation:

$$y_i[wx + b] - 1 \geq 0, \quad i = 1, \cdots, n \tag{3.3}$$

At this point, the classification interval is $2/\|w\|$. The largest interval is obtained when the value of $\|w\|^2$ is minimized so that $\Phi(w) = \frac{1}{2}\|w\|^2 = \frac{1}{2} w \cdot w$. If the hyperplane can satisfy Equation (3.3) meanwhile minimizing $\Phi(w)$, it is called the optimal hyperplane [145]. It is a quadrtic optimization problem, so the above optimal hyperplane problem can be transformed into the dual form of optimization problem by a Lagrange multiplier method. The equation is $\sum_{i=1}^{n} \alpha_i y_i = 0,$ where α_i is the Lagrange coefficient, $\alpha_i \geq 0, i = 1, \cdots, n$. To solve the following maximum function there is a unique solution.

$$\Theta(\alpha) = \sum_{i=1}^{n} \partial_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \partial_i \partial_j (x_i, x_j) \tag{3.4}$$

If ∂_i^* is the optimal solution, $w^* = \sum_{i=1}^{n} y_i \partial_i^* x_i$. According to Kuhn-Tucker conditions

[141], this optimization problem must meet $\partial_i(y_i(wx_i+b)-1)=0$, where $i=1,\cdots,n$. From the above equation, the samples which are away from the hyperplane corresponding to ∂_i must be zero. But non-zero ∂_i corresponding to samples must be located on the hyperplane, which are called support vectors. The value of b^* can be brought in the optimal hyperplane equation by any support vector, obtaining $b^*=-\frac{1}{2}(w^*x_r+x_s)$, where x_r and x_s are any support vectors from two classes respectively and they meet $\alpha_r>0$, $y_r=-1$; $\alpha_s>0$, $y_s=1$. Finally, the optimal classification function can be got as follows:

$$f(x)=\mathrm{sgn}[\sum_{i=1}^{n}w^*x^*+b^*]=\mathrm{sgn}[\sum_{i=1}^{n}y_i\alpha_i(x_i,x)+b'] \tag{3.5}$$

For nonlinear problems [145] [156], the original feature space can be mapped to a high-dimensional space by the kernel function, making the original samples linearly separable in a high-dimensional space. Constructing the optimal classification hyperplane in a high-dimensional space can transform nonlinear problems into linear problems. Assuming that nonlinear mapping $\Theta: R^d \to H$, and the input space samples R^d are mapped to a high-dimensional feature space H. when the optimal classification hyperplane is constructed in the feature space, the algorithm only uses dot product, namely $\Theta(x_i,x_j)$, in the space rather than individual $\Theta(x_i)$. Hence, a function K needs to be found, which can satisfy Equation $K(x_i,x_j)=\Theta(x_i)\cdot\Theta(x_j)$. In fact, the inner product operation is required only in the high-dimensional space, which can be achieved by the functions of the original space. Therefore, using appropriate inner product $K(x_i,x_j)$ in the optimal classification hyperplane can accomplish linear classification after a nonlinear transformation. At this moment, Equation (3.4) can be changed to Equation (3.6).

$$\Theta(\alpha)=\sum_{i=1}^{n}\partial_i-\frac{1}{2}\sum_{i-1}^{n}\sum_{j=1}^{n}y_iy_j\partial_i\partial_jK(x_i,x_j) \tag{3.6}$$

And the corresponding classication function can be changed to Equation (3.7).

$$f(x) = \text{sgn}[\sum_{i=1}^{n} y_i \alpha_i K(x_i, x) + b''] \quad (3.7)$$

In Equation (3.7), b'' is the classification threshold. If $f(x) > 0$, x belongs to one class; Otherwise, it does not belong to another class.

SVM has many unique advantages when it deals with small samples, nonlinear and high-dimensional pattern recognition problems [141] [143]. For example, the SVM algorithm is not limited in samples tending to infinity theory. It is also suitable for text classification of a large sample set, and for function fitting and other machine learning issues. Therefore, this chapter mainly introduces and uses SVM as the text classification algorithm. But the research showed that when SVM is used alone, its training speed is influenced greatly by the size of the training set because of the high computational overhead [142].

3.2.6 Classification Performance Evaluation System

In general, the text classification results are evaluated from three aspects: validity, computational complexity and simplicity of description [157]. Validity measures the ability of a classifier classifying correctly. Computational complexity includes time complexity and space complexity, which means the classification speed and the size of the employed hardware resources. Simplicity of description means that algorithms are described as simply as possible. In these three aspects, validity is the most important factor which can determine whether the text classification system is qualified, and also is the basis for other evaluation indicators [158]. The specifics are shown in Section 3.4.4 Evaluation methods.

3.3 Text Classification based on LDA

3.3.1 Gibbs Sampling Approximate Inference Parameters of LDA Model

In order to obtain the probability topic distribution of text, this thesis does not directly calculate the word distribution on topic ϕ and the topic distribution on document θ. Instead, according to the visible word sequence in document, the posterior probability $p(w|z)$ namely probability of word w giving topic z can be obtained. At last, using the Gibbs sampling algorithm can indirectly gain the value of ϕ and θ [30] [108] [111].

Markov Chain Monte Carlo (MCMC) [107] [108] [159] is an approximate iterative method, which can extract sample values from a complex probability distribution. It allows the Markov chains converge to target distribution, and then sample from the Markov chain. Each state of the Markov chain is assignment for sampling variables. Transitions between states follow simple rules. Gibbs sampling [108] [111] is a simple realization form of the MCMC, its purpose is to construct the Markov chain which converges to a probability distribution of the target, and extract the samples which are considered close to the value of the probability distribution from the Markov chain. So to determine the probability distribution function of the target is the key of Gibbs sampling.

For each word w_i, the corresponding topic variable z_i is assigned an integer t in$[1,2,\cdots,T]$, which means that this word corresponds to the t-th topic. For each word token i in the text set, w_i and d_i respectively express its word index and document index. Gibbs sampling processes each word token one by one. In the conditions of the

known topic distribution of other word token, the possibility of current word token belonging to each topic is estimated. Based on the conditional distribution namely the posterior probability, a topic is reselected as current topic of word token. This conditional distribution can be represented as $p(z_i = j \mid z_{-i}, w_i)$, where $z_i = j$ means that w_i is assigned to topic j. w_i not only represents a word, but also relates with its position in the text. z_{-i} shows the topic of other all word token except for current word token, which means the assignments of all $z_k (k \neq i)$.

For the LDA model of texts, the words of topics only need to be assigned, which means that variable z_i will be sampled. Its equation is the following [38]:

$$p(z_i = j \mid z_{-i}, w_i) \propto \frac{n_{-i,j}^{(w_i)} + \beta}{n_{-i,j}^{*} + W\beta} \cdot \frac{n_{-i,j}^{(d_i)} + \alpha}{n_{-i,*}^{(d_i)} + T\alpha} \tag{3.8}$$

where T is the number of topics and W is the number of words in the uniqueness vocabulary. $n_{-i,j}^{(w_i)}$ is the number of times of w_i that is assigned to topic j. $n_{-i,j}^{(d_i)}$ is the number of words of d_i that are assigned to topic j. $n_{-i,j}^{(*)}$ is the number of all the words that are assigned to topic j. $n_{-i,*}^{(d_i)}$ is the number of all the words that are assigned to topics in d_i. This is a non-standard probability distribution and it is not normalized. The complete Equation (3.9) is that Equation (3.8) divides by the sum of the probability of the corresponding all topics [82].

$$p(z_i = j \mid z_{-i}, w_i) = \frac{\frac{n_{-i,j}^{(w_i)} + \beta}{n_{-i,j}^{(*)} + W\beta} \cdot \frac{n_{-i,j}^{(d_i)} + \alpha}{n_{-i,*}^{(d_i)} + T\alpha}}{\sum_{j=1}^{T} \frac{n_{-i,j}^{(w_i)} + \beta}{n_{-i,j}^{(*)} + W\beta} \cdot \frac{n_{-i,j}^{(d_i)} + \alpha}{n_{-i,*}^{(d_i)} + T\alpha}} \tag{3.9}$$

where T is the number of topics and W is the number of words in the uniqueness vocabulary. $n_{-i,j}^{(w_i)}$ is the number of times of w_i that is assigned to topic j. $n_{-i,j}^{(d_i)}$ is the number of words of d_i that are assigned to topic j. $n_{-i,j}^{(*)}$ is the number of all the

words that are assigned to topic j. $n_{-i,*}^{(d_i)}$ is the number of all the words that are assigned to topics in d_i. The number of all words does not include the assignment of $z_i = j$.

To sum up, the basic idea of Gibbs sampling algorithm is to use the obtained distribution of all topics on words ϕ and the obtained distribution of document d on topics θ_d to infer current sampling word w_{di} belonging to topic z_{di}. It is an iterative process to get all the latent variables [108]. According to the above process, it can be seen that to assign a word to a topic is influenced by two probability distributions: one is the distribution of topics on words ϕ, another is the distribution of documents on topics θ.

Equation (3.9) also shows that for a given word token i, the factors that affect topic label include two parts. The left part corresponds to the probability of word w_i on topic j. The right part corresponds to the probability of topic j appearing on the topic distribution of document d. Once a word's many word tokens are labeled as topic j, the probability of any word token of this word being labeled as topic j will increase.

Therefore, Gibbs sampling algorithm in the LDA model applying to topic modeling of corpus is detailed below [82] [107] [111]:

Firstly, z_i is initialized to a random integer between 1 and T namely $z_i \in [1,T], i \in [1,N]$, where T is the number of topics and N is the number of all the feature words appearing in the text in corpus. It is the initial state of the Markov chain.

Secondly, all of the N word tokens in the text set are in turn re-assigned a new topic in each round of Gibbs sampling, which means that i loops from 1 to N

namely $for(i=1, i \le N, i++)$. According to Equation (3.9), words are assigned to topics so that the next state of the Markov chain can be calculated and obtained.

Thirdly, in the early stage of the sampling process, the result of Gibbs sampling is not accurate enough because the simulation of posterior probability is inadequate. After enough iterations to the second step and after the early stage, the result of Gibbs sampling approaches the target distribution gradually. At last, it is a steady state that is very close to the target distribution. At this moment, the Markov chain approaches the target distribution. The current value of z_i is taken and recorded as a sample.

Finally, in order to ensure that the autocorrelation is small, other samples are recorded after a number of iterations. Word token is abandoned and w represents unique word. For every single sample, ϕ and θ can be estimated by the following equations:

$$\tilde{\phi}_w^{(z=j)} = \frac{n_j^{(w)} + \beta}{n_j^{(*)} + W\beta} \tag{3.10}$$

$$\tilde{\theta}_{z=j}^{(d)} = \frac{n_j^{(d)} + \alpha}{n_*^{(d)} + T\alpha} \tag{3.11}$$

where $n_j^{(w)}$ is the frequency of word w being assigned to topic j. $n_j^{(*)}$ is the number of all words being assigned to topic j. $n_j^{(d)}$ is the number of words in document d being assigned to topic j. $n_*^{(d)}$ is the number of all the words in document d being assigned to topics.

3.3.2 The Specific Steps of Text Classification

The research showed that the training convergence rate of SVM on a large data set is slow. And it needs a lot of storage resource and very high computing power. However, its maximum interval classification model can effectively conquer the sample

distribution, redundant features, over-fitting and other factors. Besides it has good generalization ability. Compared with other classification algorithms, SVM has the advantages of effect and stability. [142] [145] has proved that its performance in text classification is more excellent than other algorithms. This chapter uses feature words' probability distribution of the LDA model, and then trains SVM classifier on the latent topic-document matrix. In order to improve the performance of text classification, the good ability of features dimension reduction and text representation of the LDA model and the strong classification capacity of SVM are combined.

The text classification method based on the LDA model uses LDA to model the text set. Using Gibbs sampling infers and indirectly calculates model parameters, obtaining the probability distribution of documents on topics [108]. After that, SVM is chosen and trained on the latent topic-document matrix. Finally, a text classification system is constructed. The specific steps are as follows:

Step 1: For the original text set, extracting stem and removing stop words form a collection of feature words. Or choose a suitable and processed text set.

Step 2: The number of optimal topics selection method is used to determine the number of optimal topics T of the LDA model. It can make the model fit the effective information in corpus data best. Details of the method are given in the next chapter.

Step 3: LDA is used to model the training text set. Model parameters are inferred by the Gibbs sampling algorithm. After enough times of iterations, each text stands for the probability distribution of fixed latent topic set. The latent topic-document matrix of the training text set $A_{t\times d}$ is gained, where t is the number of topics of the latent topic set and d is the number of documents.

Step 4: SVM is trained on the above matrix $A_{t\times d}$. Then, a text classifier is constructed, which is a SVM classification model based on LDA.

Step 5: The documents to be classified after preprocessing taking the place of document d in Equation (3.8) are processed by Gibbs sampling algorithm. After few times of iterations, the value of ϕ and θ can be calculated according to Equation (3.10) and (3.11). In addition, the probability distribution vector of latent topic set of document d to be classified is obtained.

Step 6: To bring in the constructed SVM classification model forecasts the categories of documents to be classified.

The documents to be classified are those new documents that have not been processed when the corpus is trained. If every unknown text is added to corpus, the whole model will be retrained. Obviously, it wastes too much time, and also it is not necessary. In this thesis, the documents to be classified after preprocessing are only processed by Gibbs sampling algorithm, reducing the number of iterations.

3.4 Experiment and Analysis

3.4.1 Experimental Environment

This experiment was done under a personal laptop. Its CPU is Intel(R) Core(TM) 2 Duo CPU T8100 2.10GHz, 2.09GHz. Its memory is 3.00 GB. The capacity of hard ware is 160G. The operating system is Microsoft Windows XP Professional Version 2002 SP3. The development environment is Matlab 7.0.

3.4.2 Training Environment for SVM

LIBSVM [160] [161] is a simple, fast and effective general SVM software package, which was developed by Dr Chih-Jen Lin et al. It can solve classification problems

(including *C-SVC* and *n-SVC*), regression problems (including *e-SVR* and *n-SVR*), distribution estimation (such as on*e-class-SVM*) and other issues, providing four common kernel functions which are linear function, polynomial function, radial basis function (RBF) and S-shaped function. It can effectively solve multi-class problem, cross-validation choosing the parameters, weighting for the unbalanced samples, probability estimation of multi-class problem and so on.

LIBSVM is an open source software package [161]. It not only provides algorithm source code of C++ language of LIBSVM, but also supplies Python, Java, R, MATLAB, Perl, Ruby, LabVIEW, C#.net and other languages' interface. It can be used in Windows or UNIX platform expediently [160]. Therefore, this experiment makes use of Libsvm 3.0 as the training environment for SVM, and also python2.71 and graphics software gnuplot 4.4.3.

In brief, the general steps of using LIBSVM are [160] [161]:

Step 1: According to the required format of LIBSVM software package, the data set is prepared.

Step 2: The simple scaling operation is performed on the above data.

Step 3: RBF kernel function is considered to use like $K(x, y) = e^{-\gamma\|x-y\|^2}$.

Step 4: Use cross-validation to select optimal parameters C and γ.

Step 5: Use the above optimal parameters C and γ to train the whole training set, obtaining the SVM model.

Step 6: The above gained model is used to test and forecast.

3.4.3 The Data Set

The quality of the training text set can influence the performance of machine learning classifier greatly [24]. Thus, the training text set must be able to represent various

documents which need to be processed by the classification system accurately, while training documents are able to reflect the complete text statistical information of the class accurately. In general, the text set should be recognized and classified corpus in the study of text classification so that the performance of different classification methods and systems can be compared accurately [14] [162].

Currently, the commonly used text set includes WebKB，Reuters21578，Reuters Corpus Volumnl(RCVl) and 20Newsgroup etc [163]. WebKB data set was done by the World Wide Knowledge Base Project of Carnegie Mellon University text learning group, which is consisted of web data from web sites of several universities' departments of computing. The Reuters21578 text set contained text which Reuters has been collecting since 1987. Furthermore, text indexing and classification were done manually by Reuters. The RCV1 data set is also from Reuters, and mainly for multi-label text classification.

The 20Newsgroup data set contains about 20000 documents from 20 news groups [163] [164]. This is a balanced corpus, and its each class has 1000 documents approximately. As shown in Table 3.1, this experiment chooses comp subset where there are five classes. Likewise, each class has almost 1000 documents. In addition, the training set and test set were divided according to the ratio of 3:2.

Table 3.1: The distribution of five classes of text in the 20newsgroup corpus.

Class number	Class	The number of texts	Distribution
1	comp.graphics	975	19.94%
2	comp.os.ms-windows.misc	983	20.10%
3	comp.sys.ibm.pc.hardware	985	20.14%
4	comp.sys.mac.hardware	961	19.65%
5	comp.windows.x	986	20.16%

3.4.4 Evaluation Methods

In order to evaluate the validity of the automatic text classification system, the traditional evaluation criteria are used as evaluation methods firstly, which contains Precision P, Recall R and F1 measure F1 [157] [158]. Their equations are as follows:

$$P_i = \frac{t_i}{m_i} \tag{3.12}$$

where t_i is the number of correctly classified texts in the i-th class. m_i is the number of texts which are classified by a classifier as belonging to the i-th class. Equation (3.12) shows the precision of the i-th class.

$$R_i = \frac{t_i}{n_i} \tag{3.13}$$

where t_i is the number of correctly classified texts in the i-th class. n_i is the number of texts belonging to the i-th class in the original text set. Equation (3.13) means the recall of the i-th class.

$$F1_i = \frac{2 \times P_i \times R_i}{P_i + R_i} \tag{3.14}$$

where P_i is the precision of the i-th class. R_i is the recall of the i-th class. Equation (3.14) expresses the comprehensive classification rate of the i-th class, and it is a usual method of performance evaluation which combines precision and recall together.

However, precision, recall and F1 measure are all for the performance of a single class. When the whole performance of a classification method is evaluated, the results of all classes need to be considered and combined. In general, there are two kinds of comprehensive methods, macro-average and micro-average [157]. Compared with micro-average index, macro-average index is affected greatly by small classes. They respectively include macro-average precision $Macro\_P$, macro-average

recall $Macro\_R$, macro-average F1 measure $Macro\_F1$, micro-average precision $Micro\_P$, micro-average recall $Micro\_R$ and micro-average F1 measure $Micro\_F1$.

Macro-average is the arithmetic mean of performance index of each class, and micro-average is the arithmetic mean of performance index of each document [157]. For a single document, its precision and recall are the same. In addition, a document only can give a predicted class, which is either 1 or 0. Therefore, its micro-average precision and recall are the same. According to Equation (3.14), for the same data set, its micro-average precision, micro-average recall and micro-average F1 measure are the same. Their equations are as follows:

$$Macro\_P = \frac{1}{k}\sum_{i=1}^{k} P_i \tag{3.15}$$

where P_i is the precision of i-th class and k is the total number of classes of the original text.

$$Macro\_R = \frac{1}{k}\sum_{i=1}^{k} R_i \tag{3.16}$$

where R_i is the recall of i-th class and k is the total number of classes of the original text.

$$Macro\_F1 = \frac{2 \times Macro\_P \times Macro\_R}{Macro\_P + Marco\_R} \tag{3.17}$$

where $Macro\_P$ is macro-average precision and $Macro\_R$ is macro-average recall.

$$Micro\_F1 = Micro\_R = Micro\_P = \frac{\sum_{i=1}^{k} t_i}{\sum_{i=1}^{k} m_i} \tag{3.18}$$

where t_i is the number of correctly classified texts in the i-th class, m_i is the

number of texts which are classified by a classifier as belonging to the i-th class, and k is the total number of classes of the original text.

3.4.5 Experimental Results

First of all, the original text of experiment data set is preprocessed, which includes extracting word stem, Stemming and removing stop words [148]. For example, Stemming function of Snowball (porter2) can be used to extract word stem, forming feature words, such as "success" instead of "successful". The Stemming technology can make the inflections revert to the original word. Some stop words' contribution to the classification accuracy is little, which should be abandoned. Such as "a，an，the，of". So that, it can effectively reduce the dimension of the feature space, improve the processing speed and decrease overheads of the algorithm [149].

Then, the whole training set is modeled by LDA, and setting $\alpha = 50/T$, $\beta = 0.01$ according to the experience [165]. The number of topics is given as 120 namely $T = 120$, the reason and details will be showed in Section 4.2 in the next chapter. After 1000 times of iterations of Gibbs sampling algorithm, the latent topic- document matrix $\mathrm{A}_{t \times d}$ is obtained. At this moment, each document represents the probability mixture distribution $p(z \mid d)$ in the topic set which has 120 topics. Finally, a classifier based on SVM is built on the matrix $\mathrm{A}_{t \times d}$. SVM classification algorithm is achieved by Libsvm and Matlab platform.

VSM is used for text representation, and SVM is used for classifying after extracting features as contrast experiment 1. The word-document matrix is decomposed by SVD of LSI, building a latent semantic matrix, realizing the dimension reduction of features, and combining with SVM algorithm as contrast experiment 2. This process is finished by Matlab. For example, function svds is invoked to decompose the

word-document matrix, generating a latent semantic space. Commonly, the value of k is between 100 and 300 [166]. If the value of k is too small, the useful information may be lost. But if it is too large, the storage space will limit it. According to the training text set in this experiment, the different values of k were tested, finally it was set to 150. The results of the proposed text classification method based on the LDA model in this chapter, contrast experiments 1 and 2 are shown in Figure 3.3 and Table 2.

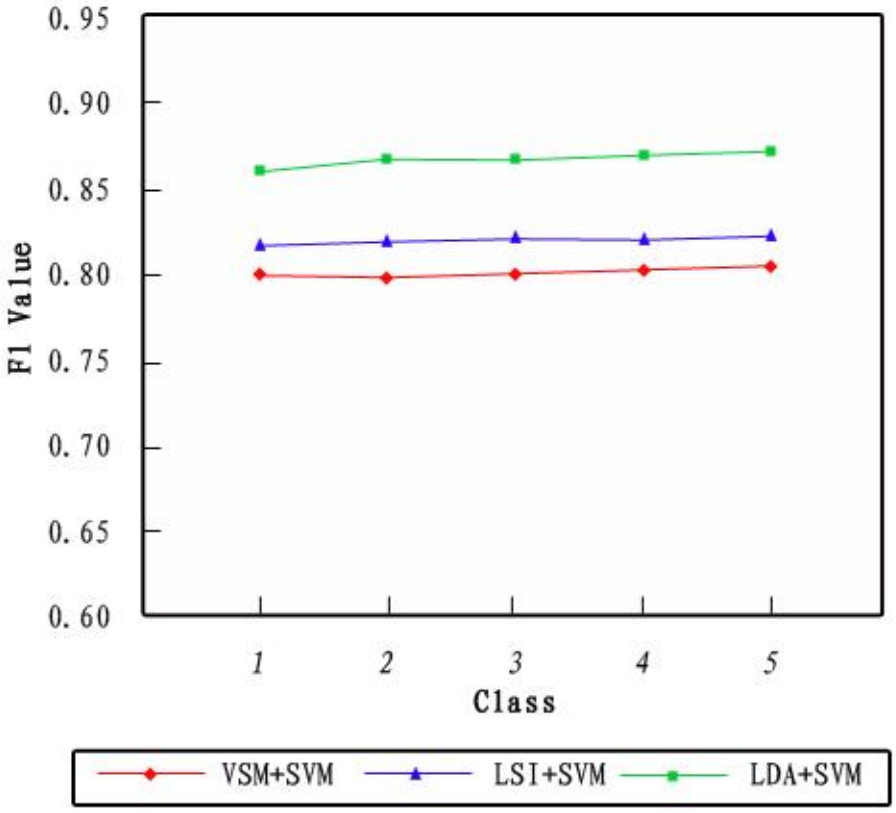

Figure 3.3: Comparison of the performance of three methods on each class.

If the difference among the classes in the data set is large, the difference between macro-average and micro-average will be relatively large. In Table 3.2, their difference is little, which indicates the difference among the classes in the training set is small. It is consistent with the actual situation. Since the five classes in this

experiment belong to the same comp subset, the difference between each other is little.

Table 3.2: Comparison of three methods' macro-average and micro-average.

Methods	**Macro_P**	**Macro_R**	**Macro_Fl**	**Micro_Fl**
VSM+SVM	0.804153	0.805347	0.805023	0.80574
LSI+SVM	0.825966	0.822337	0.82398	0.828164
LDA+SVM	0.865325	0.86968	0.864919	0.868738

After preprocessing, the number of features in the feature set of the original corpus is 36487. In contrast experiment 2, the dimension of semantic space can be reduced greatly by SVD of LSI. And compared with contrast experiment 1, contrast experiment 2 uses less number of features and also guarantees the effect of classification. The experimental results show that a text classification method based on LDA which uses Gibbs sampling to infer modeling parameters and chooses SVM as classification algorithm obtains a better effect of classification. In Table 3.2, all of its evaluation parameters reach 86%, which are higher than the other two methods.

Table 3.3: Comparison of three methods' dimensionality reduction degree to corpus.

	VSM	LSI	LDA
The number of features after dimensionality reduction	1600	150	120
Dimensionality reduction degree	95.615%	99.589%	99.671%

Furthermore, it can be showed that a LDA model with 120 topics modeling text set can decrease feature space up to 99.671% and reduce the training time of SVM in Table 3.3. And the results show that its performance of classification is not affected. It

not only can overcome the problem of the damaged classification performance which is caused by feature dimension reduction, but also avoid the issue which does not consider the semantic relations among words while extracting features. Compared with two contrast tests, the method of constructing a text classifier based on LDA and SVM can improve the performance and efficiency of text classification effectively.

3.5 Summary

At the beginning, this chapter summarized the key technologies of text classification, including text preprocessing, text representation model, text feature extraction and dimension reduction, text classification model, and classification performance evaluation. And then, a method of text classification was proposed. It made use of LDA probabilistic topic model as text representation, which is combined with SVM classification algorithm to construct a text classifier. Each document is represented as a probability distribution of the fixed latent topic set. A detailed description about how to use Gibbs sampling algorithm to estimate the modeling parameters was introduced. Besides, the steps of the proposed classification method were described. Finally, the validity and advantages of the text classifier based on LDA and SVM were validated by contrast experiments of three methods, which are VSM+SVM, LSI+SVM and LDA+SVM.

Chapter 4 Accuracy Enhancement with Optimized Topics and Noise Processing

4.1 Introduction

In the LDA model, topics obey Dirichlet distribution. This distribution assumes that a topic has nothing to do with other topics. But in fact, there is a relationship among topics. The contradiction between this independent assumption and the real data makes the changes of the number of topics affect the LDA model greatly. Therefore, when the corpus is modeled by LDA, the number of topics T has a great effect on the LDA model fitting the text set. In addition, some research showed that noisy data in the training text set can influence the result of text classification badly. So in addition to an efficient and feasible classification algorithm, it is necessary to process noisy data in the training text set before constructing the classifier and performing classification. This way, the quality of classifier and the accuracy of the result of classification can be ensured.

4.2 The Method of Selecting the Optimal Number of Topics

In the last chapter, the corpus was modeled by LDA and the SVM algorithm was combined so as to build a classifier. The number of topics was 120 in the experiment. Thus, each document was expressed as 120-dimensional polynomial vector of topic set, and the feature space was reduced by 99.671%. But, how was the optimal number of topics 120 determined? The method of selecting the optimal number of topics will be introduced below.

4.2.1 Current Main Selection Methods of the Optimal Number of Topics Based on LDA Model.

Topics in the LDA model can capture the relationship among words, but LDA cannot express the relationship among topics because the sampling based on the Dirichlet distribution assumes that topics are independent with each other. It will limit the capabilities of LDA representing the large-scale data set and predict the new data. Therefore, many scholars have begun to research more abundant structure to describe the relationship among topics. Blei, etc. presented the Correlated Topic Model (CTM) [85]. The key of CTM is that it uses the Logistic-Normal distribution instead of the Dirichlet distribution to describe latent topics of document collections. The Logistic-Normal distribution has two sets of parameters, mean vector and covariance matrix. The role of mean vector is similar to the Dirichlet parameters in LDA, which is applied to express the relative strength of latent topics. Covariance matrix represents the correlation between each pair of latent topics. The above structure information is not involved in the LDA model.

But, CTM also has limitation which can only describe the correlation between two topics. Thus, the Pachinko Allocation Model (PAM) was put forward further [167]. Its core idea is to use the Directed Acyclic Graph (DAG) to represent the structure among latent topics. In DAG of PAM, each leaf node stands for a word, every intermediate node represents a topic, and each topic is a multinomial distribution based on its child node. PAM can extend the meaning of topics. It not only can be a multinomial distribution based on the word space, but also can be a multinomial distribution based on other topics which are called super topics. Therefore, PAM can not only describe the relationship among words, but also describe the relationship among topics flexibly [167].

However, it is the same as other topic models where the number of topics T has to be

determined manually. To set the number of topics T will affect the extracted topic structure directly.

4.2.1.1 The Method of Selecting the Optimal Number of Topics Based on HDP

Hierarchical Dirichlet Process (HDP) was proposed by Y.Teh [88] as a hierarchical modeling approach in 2005, which is a kind of generative model. In addition, Y.Teh [86] put forward using HDP to find the optimal number of topics T in LDA. HDP is a kind of distribution on random probability measure set, which can model the grouped data which has a predefined multilayer structure. HDP model is shown in Figure 4.1, the rectangular boxes are "plates" that represent replicates. The hollow circles represent the latent variables G_0, G_j and θ_{ji}, and the filled circle represents the observed variable x_{ji}. In addition, the squares represent the parameters or the base distribution H, γ and α_0, and the black arrow represents the conditional dependency between two variables, namely the conditional probability. HDP contains a global probability measure G_0. Each predefined group is expressed by a Dirichlet Process（DP）. This DP corresponding probability measure G_j can be obtained from the upper level of DP sampling, such as $G_j \mid \alpha_0, G_0 \sim DP(\alpha_0, G_0)$. The members of G_0 are shared by all the DP, but the different DP has a different mixing ratio G_j. The hyper-parameters of HDP include the base distribution H, aggregation degree parameter γ and α_0. H provide prior distribution for θ_{ji}. G_0 obeys the DP with H and γ, such as $G_0 \mid \gamma, H \sim DP(\gamma, H)$. If an HDP model can be used as a grouped data about the prior distribution of θ_{ji}, for any group j, it assumes that $\theta_{j1}, \theta_{j2}, \cdots, \theta_{ji}$ are independent identically distributed random variables of G_j, such as $\theta_{ji} \mid G_j \sim G_j$. Each θ_{ji} distribution can be used to generate the corresponding

observed variable x_{ji}, such as $x_{ji} \mid \theta_{ji} \sim F(\theta_{ji})$.

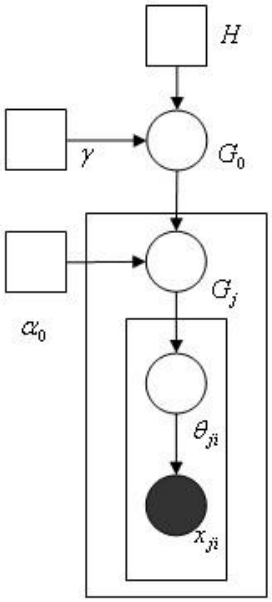

Figure 4.1: HDP model.

HDP model can be seen as a nonparametric model of LDA. In the LDA model, the weights of T topics in the text should be determined according to the Dirichlet distribution firstly. Next, one of T topics is chosen, and the specific words are selected according to the probability of all words in the topic. Furthermore, the number of topics in the HDP model can be extended. Y.Teh considered that the structure of HDP is similar to the structure of LDA so that the nonparametric feature of HDP was used to solve the problem of selecting the optimal number of topics in LDA [86]. Its basic idea is to use a near-infinite probability measure G_0 to replace the finite mixture of topics in LDA. A new DP and G_j are built for each document depending on the different mixing ratio. All documents share the mixing element of G_0. The experiment found that the optimal number of mixing element was consistent with the optimal number of topics in LDA by analyzing the histogram of sampling the number

of mixing element in the HDP model [86]. Therefore, it can solve the problem of selecting the optimal number of topics T in LDA.

However, the above method has to establish respectively an HDP model and a LDA model for the same one data set. Obviously, when a large set of documents or the real text set is processed, the method based on HDP will spend lots of time in computing [87] [168].

4.2.1.2 The Standard Method of Bayesian Statistics

In the LDA topic model, the selection of the number of topics needs to be given manually. And, the number of topics will affect the quality of LDA modeling document collection directly. The classic method is to use the standard method of Bayesian statistics [82] to select the optimal number of topics T. In the LDA model, α and β respectively are the Dirichlet prior probability hypothesis on the multinomial distribution θ and ϕ. The feature of its natural conjugate means that the value of the joint probability $p(w,z)$ can be obtained by integrating θ and ϕ. Because $p(w,z)=p(w|z)p(z)$ where ϕ and θ appear in the first and second item of the right side of the equation respectively, the value of the first item $p(w|z)$ can be gained by integrating ϕ. The corresponding equation is as follows:

$$p(w|z)=\left(\frac{\Gamma(W\beta)}{\Gamma(\beta)^{W}}\right)^{T}\sum_{j=1}^{T}\frac{\prod_{w}\Gamma(n_{j}^{(w)}+\beta)}{\Gamma(n_{j}^{(*)}+W\beta)} \tag{4.1}$$

where, $\Gamma(*)$ is the standard gamma function. $n_j^{(w)}$ stands for the frequency of word w being assigned to topic j. $n_j^{(*)}$ represents the number of all the words being assigned to topic j. $p(w|T)$ can be approximate to the average value of $p(w|z)$, and M is defined as the frequency of sampling in the Gibbs sampling algorithm. The corresponding Equation is as follows:

$$\frac{1}{p(w|T)} = \frac{1}{M}\sum_{m=1}^{M}\frac{1}{p(w|z^{(m)})} \tag{4.2}$$

The probability distribution of the different number of latent topics for modeling text set can be obtained by Equation (4.2), which is the value of $p(w|T)$. The value of $p(w|T)$ is proportional to the fitting degree of the LDA model to effective information in the text set, to determine the optimal number of latent topics T [82].

4.2.2 A Density-based Clustering Method of Selecting the Optimal Number of Topics in LDA

The clustering is to gather a large number of data samples into many classes, which can make the similarity among samples in the same class maximum but in the different class minimum. From this perspective, a density function can be designed to calculate the density around each sample. According to the value of the density around each sample, the areas where the samples are relatively concentrated can be found. These areas are the targeted classes. This clustering method is called density-based clustering [169] [170]. Its basic idea is to use the high-density areas among the low-density areas in the data space to be defined as classes. The adjacent text areas with a major distribution density of the data space are connected to recognize the irregular shape of classes, remove some noisy data and process abnormal data effectively. In other words, as long as the density of points in an area is greater than a threshold value, the points will be added into the cluster which is close to them. In addition, it has been proved that the density-based clustering is a very effective clustering method. The representative algorithm is the Density-Based Spatial Clustering of Application with Noise (DBSCAN) [171] [172] [173] [174].

4.2.2.1 DBSCAN Algorithm

First of all, DBSCAN defines two parameters: the cluster radius ε and the value of MinPts. The former is the cluster density which is represented by the mean distance among clusters. The latter is the number of text within the scope of a single cluster. Then, several basic concepts of DBSCAN algorithm will be introduced as follows [171] [172] [175]:

- Density: the density of any point in space is the number of points in the circular area where the centre of the circle is the point and the radius is ε.
- ε-Neighborhood: the neighborhood of any point in space is the points in the circular area where the centre of the circle is the point and the radius is ε.
- Core Object: if the density of any point in space is greater than a given threshold MinPts, it is called a core object.
- Border Object: if the density of any point in space is less than a given threshold MinPts, it is called border object.
- Directly Density Reachable: a data set D, ε and MinPts are given, and $p \in D, q \in D$. If q is a core object and p is the ε-Neighborhood of q, p is directly density reachable from q.
- Density Reachable: a data set D, ε and MinPts are given. If there is a chain $p_i \in D(1 \le i \le n,)$ and $p_1 = q, p_n = p$, and p_i can reach p_{i+1} directly, p is density reachable from q.
- Density Connected: a data set D, ε and MinPts are given, and $p \in D, q \in D$. If there is an object r, and p and q are density reachable from r, p and q are density connected.
- Cluster: it is the collection of the maximum density connected points based on directly density reachable.
- Noise: the noise is the objects which are not included in any cluster.

In summary, the basic idea of DBSCAN algorithm is that for each object in a cluster, the number of objects in the neighborhood with the given radius ε must be greater than a given value. In other words, the neighborhood density must be greater than a certain threshold MinPts. According to the above nine definitions, it can be seen that DBSCAN algorithm is based on that any core point object can determine a unique class [174].

In brief, its algorithm process is as follows [171] [173]: firstly, the core object is searched in a data set. Once a core object is found, a cluster with this core object is generated. Next, DBSCAN will search directly density reachable objects from these core objects by loop iteration, in which some density reachable clusters may be merged. Finally, the end condition of this algorithm is that no any new sample points can be added to any cluster.

In addition, the density-based clustering algorithm DBSCAN is suitable for processing large-scale text set without specifying the number of classes. And, it also has a high processing speed, handles noise points effectively, and discovers arbitrary shape of clusters [171] [175].

4.2.2.2 The Relationship between the Optimal Model and the Topic Similarity

The LDA model can extract the latent topic structure in data set by analyzing a large number of statistical data. And, each data set has an optimal structure. Thus, when the average similarity among topics is the smallest, the model will be optimal. The following is the relevant proof in theory.

The distribution of topic in a V-dimensional word space in β matrix $p(w_v \mid z_i)$ is used to represent topic vector, and the relevance among topic vectors is measured by standard cosine distance of vectors:

$$corre(z_i, z_j) = corre(\beta_i, \beta_j) = \frac{\sum_{v=o}^{V} \beta_{iv} \cdot \beta_{jv}}{\sqrt{\sum_{v=0}^{V} (\beta_{iv})^2 \sum_{v=0}^{V} (\beta_{jv})^2}} \tag{4.3}$$

The smaller the value of $corre(z_i, z_j)$, the more independent the relevance between topics. At this moment, the average similarity among all the topics is used to measure the stability of the topic structure:

$$avg\_corre(structure) = \frac{\sum_{i=1}^{T-1} \sum_{j=i+1}^{T} corre(z_i, z_j)}{T * (T-1)/2} \tag{4.4}$$

$$\arg\min(avg\_corre(structure)) = \arg\min\left[\frac{\sum_{i=1}^{T-1} \sum_{j=i+1}^{T} corre(z_i, z_j)}{T * (T-1)/2} \right]$$

$$\arg\min(avg\_corre(structure)) = \arg\min\left[\sum_{i=1}^{T-1} \sum_{j=i+1}^{T} \frac{\sum_{v=o}^{V} \beta_{iv} \cdot \beta_{jv}}{\sqrt{\sum_{v=0}^{V} (\beta_{iv})^2 \sum_{v=0}^{V} (\beta_{jv})^2}} \right] \tag{4.5}$$

According to Bayes' theorem [82], $p(w_n \mid z_n) \propto p(z_n \mid w_n) p(w_n)$. Thus, Equation (4.5) can be converted into the function about $p(z_i \mid w_v)$. Combined with the constraint condition of the LDA model $\sum_{i=1}^{T} p(z_i \mid w_v) = 1$, when all w_v on a certain z get obvious peaks, the above equation can meet the condition. The condition can be further expressed as:

$$\max \sum_{i=1}^{T} p(z_i \mid w_v)^2 \tag{4.6}$$

Meanwhile, the optimization function can be stated as Equation (4.7) in the process of EM algorithm iteratively computing the model parameters α and β.

$$(\alpha^*, \beta^*) = \arg\max\ p(D \mid \alpha, \beta) = \arg\max \prod_{d_i \in D} p(d_i \mid \alpha, \beta) \tag{4.7}$$

where $p(d \mid \alpha, \beta) = \prod_{n=1}^{N_d} \sum_{i=1}^{T} p(\theta_d) p(z_{di} \mid \theta_d) p(w_n \mid z_{di})$ (4.8)

Blei et al used variational method to approximate inference [32], two latent variables γ and ϕ were brought in. Equations (4.9) and (4.10) can be obtained as follows:

$$\gamma_i = \alpha_i + \sum_{n=1}^{N} \phi_{ni} \tag{4.9}$$

$$\beta_{ij} \propto \sum_{d=1}^{M} \sum_{n=1}^{N_d} \phi_{ni} w_{dn}^{j} \tag{4.10}$$

where ϕ_{ni} stands for the probability $p(z_i \mid w_n)$ that the n-th word is generated by topic z_i.

Equations (4.9) and (4.10) are put into Equation (4.8), to deduce that Equation (4.8) will satisfy the following relationship:

$$p(d \mid \alpha, \beta) \propto \prod_{n=1}^{N_d} \sum_{i=1}^{T} (\phi_{ni})^2 \tag{4.11}$$

Obviously, when Equation (4.6) is satisfied, $p(d \mid \alpha, \beta)$ can reach maximum. According to Equation (4.7), it meets the condition of the optimal model at this moment. Therefore, it is proved that when the average similarity among topics is the smallest, the model is optimal in the LDA model.

4.2.2.3 A Method of Selecting the Optimal Number of Topics Based on DBSCAN

As the standard method of Bayesian statistics in Section 4.2.1.2, the optimal number of topics in LDA can be found under the condition of the given number of topics [82]. By the theoretical proof in Section 4.2.2.2, the relationship between the optimal LDA model and the topic relevance can be obtained. Namely, when the average similarity of the topic structure is the smallest, the corresponding model is optimal. Using it as a

constraint condition, selection of the optimal number of topics T and parameters estimation of the LDA model are combined in a framework. According to the idea which calculates the density of samples to measure the correlation among topics based on the density-based clustering algorithm, a method of selecting the optimal number of topics based on DBSCAN is proposed in this chapter.

Its basic idea is to calculate the density of each topic, and find the most unstable topic under the known structure. Then, it is iterated until the model is stable. Next, the optimal number of topics in the LDA model can be found automatically without a given number of topics in advance. At the end, the effectiveness of the proposed method will be verified by experiment in Section 4.2.3.

At the beginning, according to the basic idea and concepts of DBSCAN [171] [172] [173], three definitions are made as follows:

- Topic density: topic ***Z*** and distance ***R*** are given. Draw a circle with the centre of the circle being ***Z*** and the radius ***R***. According to Equation (4.3), the similarity between topic ***Z*** and other topics is calculated respectively. The number of topics with the similarity values falling within the circle is called the density of ***Z*** based on ***R***, written as Density ***(Z, R)***.
- Model parameter: a topic model ***M*** and a positive integer ***n*** are given. The number of topics whose density is less than or equal to ***n*** in the model is called the base of the model, written as Parameter ***(M, n)***.
- Reference sample: for a point ***z*** in the topic distribution, radius ***r*** and a threshold ***m***, if Density$(z, r) \leq m$, ***z*** representing the word space vector is called a reference sample of topic ***z***. But, a reference sample is not a real document vector in the data set. It is a virtual point in the spatial distribution of words.

On the basis of the above definitions, the process steps of the method of selecting the optimal number of topics based on density are as follows:

Step 1: According to any given initial values of K, the complete statistical matrix of the Dirirchlet distribution is initialized by random sampling, to obtain an initial model LDA（α，β）.

Step 2: Taking the topic distribution matrix β of the initial model LDA（α，β） as a result of the initial clustering, the similarity matrix among all the topics and the average similarity are calculated according to Equation (4.3), $r = avg\_corre(\beta)$. Based on ***r***, the density of all topics is obtained, which is represented as Density(***Z***,***r***). According to the second definition, let ***n*** = 0, parameter P of this model ***M*** can be calculated, which is expressed as P = Parameter (M, 0).

Step 3: According to the reference value of K in Step 2, the model parameters of LDA will be re-estimated. The updated function of K is as follows:

$$K_{n+1} = K_n + f(r) \times (K_n - P_n) \tag{4.12}$$

In Equation (4.12), $f(r)$ stands for the direction of change of ***r***. When it is opposite to the previous direction, $f_{n+1}(r) = -1 \times f_n(r)$. When it is the same as the previous direction, $f_{n+1}(r) = f_n(r)$. Let $f_0(r) = -1$, when $f(r) = -1$, topics will be sorted by density from small to large. The first P topics are regarded as reference samples, and then the complete statistical matrix of the next LDA model parameter estimation is initialized. Step 2 and step 3 are repeated until the average similarity ***r*** and parameter K converge at the same time.

Equation (4.12) shows that the condition of K convergence is $\arg\min(K_n - P_n)$. From the definition of model parameters, P increases with the decrease of the average similarity ***r***, and $P_n \le K_n$. When ***r*** is the minimum, ***P*** is the maximum. Hence, ***r*** and K can be guaranteed to converge at the same time. At this moment, the obtained value of K is the optimal number of topics in the LDA model.

4.2.3 Experiment and Result Analysis

In this Chapter, the experiment environment, data set and the related data preprocessing are the same as the previous Chapter. The basic idea of this experiment is to use the standard method of Bayesian statistics in Section 4.2.1.2 to find the optimal number of topics T in the LDA model firstly. Then, the value of T is regarded as a reference value. Finally, the obtained optimal number of topics by the proposed method based on density is compared with the reference value, to verify the effectiveness of the proposed method.

Let $\alpha = 50/T, \beta = 0.01$ [165], and T take different values, such as 20, 40, 60, 80, 100, 120, 140, 160, 180, 200. According to different values of T, Gibbs sampling algorithm is run respectively, and the changes of the values of $\log p(w|T)$ are observed. The experimental result is shown in the Figure 4.2 below.

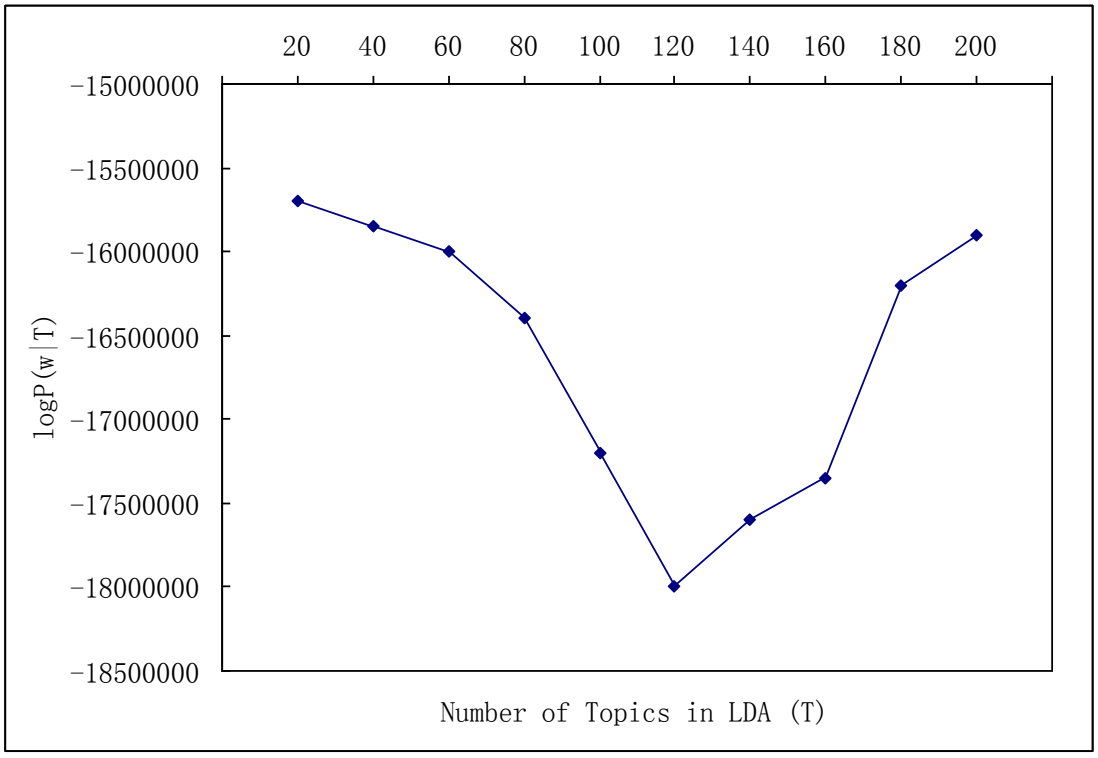

Figure 4.2: The relationship between logP(w|T) and T.

It can be seen from Figure 4.2 that when the number of topics T is 120, the value of $\log p(w|T)$ is the minimum. After that, it begins to increase sharply. This result shows that when the number of topics T is 120, the model can fit the useful information of the corpus best. Therefore, the number of topics was set to 120 in the experiment in Chapter 3. In other words, the optimal number of topics in the LDA model is 120, which will serve as a reference value to prove that the proposed method based on density in this Chapter is effective and feasible.

The method based on density of selecting the optimal number of topics is used. T is initialized with seven different values for seven groups of tests, such as 10, 50, 100, 200, 300, 400 and 500. Their results are shown in Table 4.1. It can be seen from Table 4.1 that the proposed method can mainly find the optimal value of T after several iterations, meanwhile the optimal value of T is 120. In addition, if the initial value is closer to the optimal value of T, the number of iterations will be less.

Table 4.1: Results of the proposed algorithm to find the optimal value of topic.

Initial values of topics	The optimal value of topic	Iterations
10	118	19
50	117	27
100	125	6
200	124	14
300	123	25
400	126	31
500	124	38

In the standard method of Bayesian statistics, the number of topics T has to be

assigned a series of values manually. After Gibbs sampling algorithm for the LDA model, the corresponding values of $\log p(w|T)$ are gained in sequence so that the optimal number of topics can be determined. However, the given values of T are not universal, and the values of $\log p(w|T)$ have to be calculated. Therefore, its processing time is long, and its efficiency is poor.

The method based on DBSCAN of selecting the optimal number of topics can find the optimal number of topics in the LDA model automatically with relatively less iterations, which does not need to be assigned the number of topics manually. It can be seen from the above experiment that the propose method based on density is able to find the optimal number of topics in the LDA model. Therefore, it is effective and feasible.

4.3 Noisy Data Reduction

4.3.1 The Noise Problem in Text Classification

In order to deal with huge amounts of text information and improve the efficiency of classification, the automatic text classification based on statistical theory and machine learning has replaced the manual text classification gradually. Generally, automatic text classification is divided into two stages. In the first stage, a classifier is constructed by the training text set with category labels. In the second stage, new documents are classified by the constructed classifier. As can be seen, the quality of the classifier will influence the final result of text classification directly, and it depends on the quality of the training text set to a great extent. Generally speaking, if classes of the training text set are more accurate and its content is more comprehensive, the quality of the obtained classifier will be higher [89].

But in the actual application of text classification, it is difficult to provide the training

text set of high quality and precise classification in advance [91]. Some coarse classification data is often used as training text directly, such as the collated webpage data according to the predefined classes and papers which are divided into various classes in journals. These data has the class features, which means that data in each class has some common features. But in the text set of various classes, there are a certain number of documents which have wrong class labels. In other words, the content of the documents does not match with the tagged classes. These documents with wrong class labels are called noise data in text classification. If the text data of coarse classification with noise data is used to training classifier, the accuracy of the classification result will be influenced [90].

Generally, the noise data can be divided into two categories [89]: characteristic noise and category noise. In a text classification system, all the categories are usually on the same level, which means they are in the same plane of the class space. When the classification operation is executed, text to be classified will be compared with all of the categories. The text will be assigned to appropriate categories. In the field of text classification, the characteristic noise of the training text set usually can not be considered. The influence of the category noise to the classification result is mainly reflected in the following two aspects:

- In the process of classification, except for the classifier which is really associated with text to be classified, the output of other classifiers is noise information which affects the final classification result. It is difficult to correct classification error which is caused by the classification noise. And in the practical application of text classification, a large number of classes and large ambiguity among classes make this kind of noise influence the classification performance more seriously.

- In the method of plane classification, when the number of categories increases, the processing time of the classifier and the computational cost will increase. For

either the single classification system where the classifier only outputs one of the most relevant classes, or the multi-classification system where the classifier can output multiple related classes, the number of categories which are really related to the text to be classified is limited. In fact, lots of processing time and memory space is spent on dealing with category noise, which leads to the increase in the cost of the classifier and the decrease in the response time and classification effectiveness.

In addition, there are two kinds of sources in category noise [91]: the first one is the inconsistent sample. For example, the same one sample is in different categories. Another one is the wrong label. For instance, the training samples are classified manually with wrong labels. The latter often happens in the classification of large-scale training set, so the research of this chapter only focuses on the second form of category noise.

4.3.2 The Current Main LDA-based Methods of Noise Processing

4.3.2.1 The Data Smoothing Based on the Generation Process of Topic Model

In the classical LDA model, each text has its own independent topic distribution. In text classification, assume that the documents in the same class have the same probability distribution of topics. Thus, the construction process of the training text set can be regarded as a generation process of topic model. Each class in the text set has a corresponding latent probability topic model. For text collection which is corresponding to each class in training set, a probability topic model can be obtained by a topic extraction algorithm. The new text which is generated by the topic model still belongs to the class which is corresponding to this model [92].

To sum up, if word token i is kept, Equation (3.10) and Equation (3.11) in Section 3.3.1 in Chapter 3 can be changed to:

$$\phi_i^{'(z=j)} = \frac{C_{ij}^{WT} + \beta}{\sum_{k=1}^{W} C_{kj}^{WT} + W\beta} \tag{4.13}$$

$$\theta_{z=j}^{'} = \frac{\sum_{d=1}^{D} C_{dj}^{DT} + \alpha}{\sum_{k=1}^{T}\sum_{d=1}^{D} C_{dk}^{DT} + T\alpha} \tag{4.14}$$

where W stands for the number of all the words in the text set. T represents the number of all the topics in the text set. D means the number of all the documents in the text set. C^{WT} and C^{DT} are respectively the integral matrix of $W \times T$ and $D \times T$. C_{ij}^{WT} stands for the frequency of word w_i being assigned to topic j. $\sum_{k=1}^{W} C_{kj}^{WT}$ represents the frequency of all words being assigned to topic j. $\sum_{d=1}^{D} C_{dj}^{DT}$ means the frequency of words in all the text being labeled. $\sum_{k=1}^{T}\sum_{d=1}^{D} C_{dk}^{DT}$ shows the total number of times words in all the text being labeled with topics.

Therefore, as long as the probability topic models which are corresponding to each class can be extracted from the training text set, the whole training text set is able to be regenerated by these models. The regenerated new text obeys the probability distribution of real data, which can be widely used in smoothing noise and other fields [92] [176] [177]. The process of the algorithm which can extract the LDA model from the text collection of a single category is shown as follows:

Step 1: All the words in the text set are statistic to construct the vector $\vec{t} = (t_1, t_2, \cdots, t_N)$, where N is the total number of words.

Step 2: Word index $\overline{WS}$ and document index $\overline{DS}$ of each word are established. WS_i stands for the word index of the i-th word. DS_i represents the document index of the i-th word, which means that the i-th word is from the DS_i-th document.

Step 3: The topic label vector of words $\vec{z}$ is initialized randomly. For example, z_i shows that the i-th word is labeled to the z_i-th topic. C^{WT} and C^{DT} are updated at the same time, and vector $\overline{zt}$ is used to record the number of occurrences of each topic.

Step 4: For each word w_i, let the corresponding values of C^{WT}, C^{DT} and $\overline{zt}$ decrease by 1, where $i \in \{1,2,\mathrm{L},N\}$.

Step 5: According to Equation (3.9) in Section 3.3.1 in Chapter 3, the probability of each word w_i belonging to various topics $p(z_i = j \mid z_{-i}, w_i)$ can be computed, where $j = 1,2,\mathrm{L},T$.

Step 6: According to the value of $p(z_i = j \mid z_{-i}, w_i)$ namely a new probability distribution of topics, a topic j is chosen randomly as a new topic of word w_i, and all the words in the text set are performed in turn. Besides, C^{WT}, C^{DT} and $\overline{zt}$ are updated at the same time, meanwhile their corresponding values will increase by 1.

Step 7: All the words in the text set are repeated by steps 4, 5 and 6.

Step 8: The steps 4, 5, 6 and 7 are one computation period of Gibbs sampling. Repeat the process until that the sampling result reaches convergence or the maximum number of iterations is reached.

Step 9: According to Equations (4.13) and (4.14), the approximate solution θ' and ϕ' of θ and ϕ can be obtained.

Step 10: The LDA model with θ' and ϕ' is output and stored in the local hard disk.

Usually, the extracted LDA model contains a large amount of data, and takes up major memory space. If there are many classes in the training text set, the LDA model will not be suitable to be stored in memory. So in step 10, the LDA model is stored in the local hard disk. When many classes of the training text set are processed, the model will be loaded into memory so as to save memory space.

After finishing extraction of the LDA model of a class, the model can be used to generate the text which belongs to the class. In brief, the specific process of the LDA model generating a text is as follows [92]:

Step 1: The LDA model is loaded from the above stored file so that the approximate solution θ' and ϕ' can be gained.

Step 2: The average length N of all the documents which belong to the class in the training text set is calculated.

Step 3: Let the length of new text be d, and $d = \text{Ø}$.

Step 4: According to the topic probability distribution of text θ', a topic j is selected randomly as the topic which the current word belongs to.

Step 5: According to word probability distribution on topic j $\phi'^{(z=j)}$, a word w is chose randomly as a new word in the new text. And, let $d = d \cup \{w\}$.

Step 6: If $|d| <$ N, return to step 4, namely, repeat steps 4 and 5 until $|d| = N$.

Step 7: The document d is output.

In general, in the text set with noise samples, the number of noise samples in each class relative to the number of normal samples in the same class is the minority [177]. The obtained LDA model by the Gibbs sampling algorithm can almost reflect the correct semantic information of the class. Although it will be influenced by noise samples, the new generated text is basically close to the class. Based on this point, for each category of the training text set, the corresponding LDA model can be got by the

Gibbs sampling algorithm. Then, the obtained LDA model is used to regenerate all samples of the corresponding classes, which can replace the original samples as the new training samples. Finally, this method can achieve data smoothing and weaken the influence of noise to some extent [92] [95].

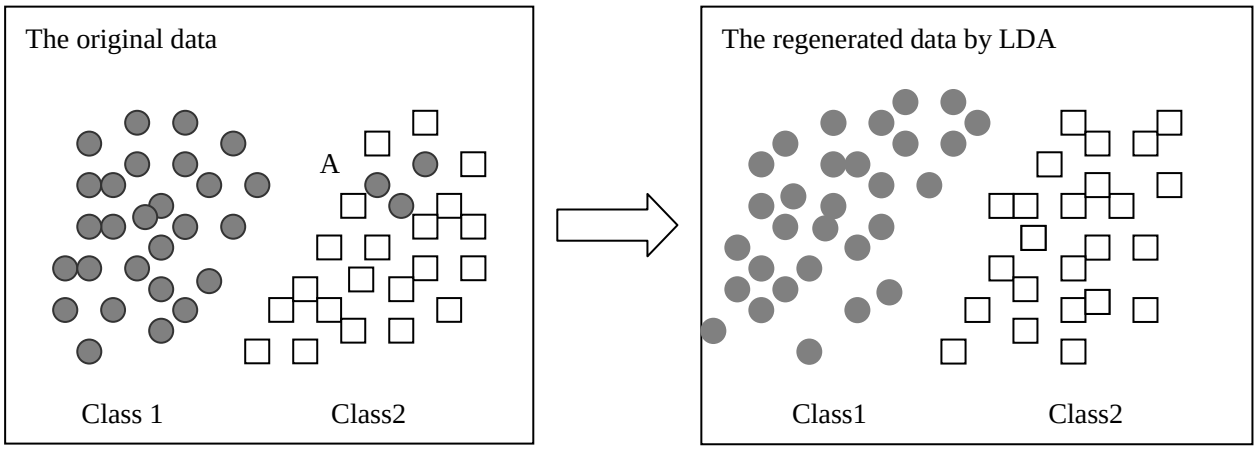

Figure 4.3: The data smoothing based on LDA weakens the influence of noise samples.

For example, there are two classes in location A in the original training data in Figure 4.3. The samples of class 2 are incorrectly labeled to class 1. If they are processed without data smoothing, the noise samples will influence the obtained classifier. Some samples with obvious class features may be sorted into wrong classes. If the training set is processed by data smoothing based on the generative process of the LDA model, the quality of the regenerated training data is significantly higher than the original data with noise.

4.3.2.2 The Category Entropy Based on LDA Model

The concept of entropy is from thermodynamics. In thermodynamics, entropy is defined as the Log Koc of the number of available system states, which is called thermal entropy. It is a physical quantity which can express the degree of molecular

state clutter. Information entropy was first proposed by the American electrical engineer C.E.Shannon in 1948 [178], which brought the thermodynamics entropy into information theory. The studies have shown that thermal entropy and information entropy are connected on the quantity. The symbols of information entropy and thermodynamic entropy should be opposite [94]. Information entropy can measure the uncertainty or amount of information of a random event. For example, the size of the uncertainty of the random event can be described by its probability distribution function. And, the amount of information can be expressed by the size of the eliminated uncertainty. If a system is more ordered, the information entropy will be smaller. Conversely, if a system is more chaotic, the information entropy will be higher. Therefore, information entropy is also a measurement of the degree of the system ordering.

Suppose a discrete random event X has n states, which are $x_1, x_{2,} \cdots, x_n$. Their corresponding priori probabilities are $p_1, p_2, \cdots, p_n$, where $p(X = x_i \mid X = x_j) = 0, i \neq j$ and $\sum_{i=1}^{n} p_i = 1$. The information entropy H(X) satisfies the following three conditions [178]:

- Continuity: $H(p_1, p_2, \cdots, p_n)$ is the continuous function of p_i, where $i = 1,2, \cdots, n$.
- Symmetry: $H(p_1, p_2, \cdots, p_n)$ has nothing to do with the ordering of $p_1, p_2, \cdots, p_n$.
- Additivity: if $p_n = Q_1 + Q_2$ and $Q_1 \geq 0, Q_2 \geq 0$,

$$H(p_1, p_2, \cdots, p_n) = H(p_1, p_2, \cdots, p_{n-1}, Q_1, Q_2) = H(p_1, p_2, \cdots, p_{n-1}) + p_n H\left(\frac{Q_1}{p_n}, \frac{Q_2}{p_n}\right)$$

The only expression of information entropy H is:

$$H(p_1, p_2, \cdots, p_n) = -\lambda \sum_{i=1}^{n} p(x_i) \log_{\psi} p(x_i) \tag{4.15}$$

where λ is a positive integer. In general, $\lambda = 1$, Equation (4.15) can be changed to Equation (4.16):

$$H(p_1, p_2, \cdots, p_n) = -\sum_{i=1}^{n} p(x_i) \log_{\psi} p(x_i) \tag{4.16}$$

Equation (4.16) is the most basic expression of information entropy, where ψ is related to the unit of information entropy. When $\psi = 2$, the unit of entropy is bit. When $\psi = e$, the unit of entropy is nat. When $\psi = 10$, the unit of entropy is dit [178].

There are always some documents in the training text set. When they are processed by a classifier, it is not very definite that which class they should belong to. Because they may be noise, or may be normal samples which lie on the boundary of classification. If they are the latter, they can be used to improve the accuracy of the classifiers.

However, the possibility of these samples belonging to normal samples of the boundary of classification is very small in a noisy environment. If they are kept, there will be a huge risk. Therefore, a method of category entropy based on the LDA model can eliminate noise samples effectively [179] [180].

If we assume that the training text set contains n categories which are $C = \{C_1, C_2, \cdots, C_i, \cdots, C_n\}$, the training set can be divided into n subsets which are $D = \{D_1, D_2, \cdots, D_i, \cdots, D_n\}$ according to the categories. The corresponding LDA models $M = \{M_1, M_2, \cdots, M_i, \cdots, M_n\}$ are extracted by the Gibbs sampling algorithm from these subsets. For each document d in the training set, the similarity

between document d and model M_i can be used to represent the likelihood of it belonging to class C_i:

$$P(d, M_i) = \prod_{w=1}^{W_d} \sum_{t=1}^{T} \theta_{M_i}(t) \times \phi_{M_i}^{(t)}(w) \tag{4.17}$$

$$sim(d, M_i) = \frac{P(d, M_i)}{\sum_{j=1}^{n} P(d, M_j)} \tag{4.18}$$

where W_d stands for the number of all the words in document d. $\theta_{M_i}(t)$ means the probability of topic t appearing in model M_i. $\phi_{M_i}^{(t)}(w)$ shows the probability of word w which belongs to topic t appearing in model M_i.

Obviously, if the similarity between document d and a model M_i is very high but the similarity between document d and other models is quite low, the possibility of document d belonging to the class which is corresponding to the model M_i will be very large [94]. On the other hand, if the similarity between document d and every model is similar, it will be difficult to determine which class document d belongs to. According to the concept of information entropy, the category entropy of each training sample is calculated. Then, the samples with uncertain categories are removed by the comparison of category entropy [178]. The category entropy of document d can be calculated by the following Equation:

$$H(d) = -\sum_{i=1}^{n} sim(d, M_i) \times \lg(sim(d, M_i)) \tag{4.19}$$

As can be seen from Equation (4.19), if the category of document d is more indeterminate, the corresponding category entropy $H(d)$ is higher. Because the uncertainty of category of noise samples is larger, their corresponding category entropy is higher than the normal samples [181]. Thus, the category entropy of all the

samples in the training text set should be calculated firstly. Then, all the samples are sorted by their category entropy. Finally, the samples which have the highest values of the category entropy will be removed according to a certain ratio or a threshold, abandoning most of noise samples effectively [93].

4.3.3 A Noise Data Reduction Scheme

At present, major studies have focused on classification algorithm research, such as feature selection, selecting the training text and the combination of some algorithms, to improve the performance of the classifier [182]. But, the algorithms about improving the quality of training data are neglected. Although the above two methods of noise processing based on the LDA model can remove most of noise samples in the text set effectively, they cannot remove all the noise completely [93] [94]. In addition, some normal samples are wrongly deleted as noise inevitably [96]. Therefore, a noise data reduction scheme is proposed here in the thesis, which combines the method of data smoothing and the method of noise identification based on the category entropy. The specific steps of the algorithm are described as follows:

Input: the original training text set D, the ratio of noise reduction q and the maximum number of iterations K.
Output: the final training text set D^f.

Step 1: The original training text set is duplicated as the current training text set D^1.

Step 2: In the new training text set D^1, the LDA models $M = \{M_1, M_2, \cdots, M_n\}$ are extracted from D^1 for all the classes $C = \{C_1, C_2, \cdots, C_n\}$ by the Gibbs sampling algorithm.

Step 3: For each sample d ($d \in D^1$), their category entropy $H(d)$ is calculated respectively according to the models M.

Step 4: The obtained category entropy is sorted. Then, the samples which have the top values of the category entropy in D^1 will be removed, obtaining the training text set D^2 after noise reduction, where the number of the removed samples is $q \times |D^1|$.

Step 5: In the training text set D^2, the new LDA models $M^1 = \{M^1{}_1, M^1{}_2, \cdots, M^1{}_n\}$ are extracted by the Gibbs sampling algorithm.

Step 6: In the training text set D^2, new documents are generated by models $M^1 = \{M^1{}_1, M^1{}_2, \cdots, M^1{}_n\}$ for all the classes $C = \{C_1, C_2, \cdots, C_n\}$, where the number of new documents is the same as the number of documents in each class in D^1. Then, the training text set D^3 is gained after data smoothing.

Step 7: If the current number of iterations k reaches the maximum number of iterations K, the algorithm will go on to the next step. Otherwise, let $D^1 = D^3$ and return to step 2.

Step 8: At the end, the final training text set D^f is obtained as: $D^f = D^3$.

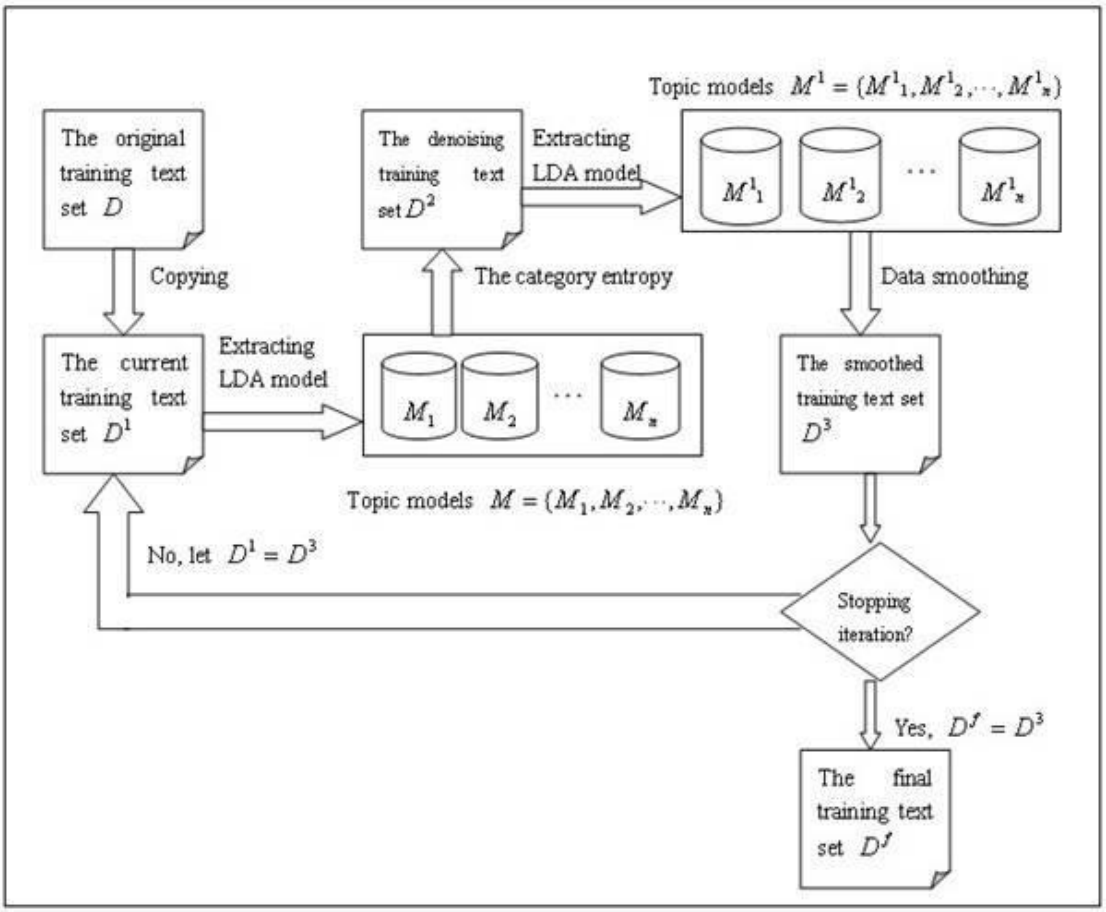

Figure 4.4: The flow diagram of a noise data reduction scheme.

The algorithm flow chart in Figure 4.4 can summarize the above process briefly. In step 5, most of noise samples have been removed by the noise processing based on the category entropy, improving the quality of the training text set. Therefore, M^1 is closer to the real topic model than M. In step 6, the new training text set is generated by M^1 so that the quality of the training text set is improved further, where the number of generated texts is the same as the number of the original training text set. At this time, an iterative process of the algorithm is completed.

However, the training text set is usually still influenced by noise samples after an iterative process. And, there is the space for improving the quality of the training text set. Thus, the smoothed text set will be the initial training data of the next iteration.

After iterations, the obtained new text set is the final training text set. The whole process needs K times of iterations. Then, a new classifier can be constructed as the optimal classifier of text classification according to the appropriate classification algorithm. Usually, 3 times of iterations can make the algorithm achieve the optimal state. But if the number of iterations is too much, the effect will be reduced because of information lose. The reason and details will be given in Section 4.3.4.

4.3.4 The Experiment and Result Analysis

In this chapter, the experiment environment, experiment data set and related data preprocessing are the same as the experiment in Chapter 3. As shown in Table 4.2, the texts which are from class 1 and class 2 in Table 3.1 are mixed as the first group of data. Then, the texts which are from class 3, class 4 and class 5 in Table 3.1 are mixed as the second group of data. Finally, the texts which are from all the five classes in Table 3.1 are mixed as the third group of data. The training set and test set are still divided according to the ratio of 3:2. And, SVM is chosen as the classification algorithm like in the experiment in Chapter 3.

Table 4.2: Experimental data sets.

Group of data	Class	The number of texts
1	comp.graphics, comp.os.ms-windows.misc	1958
2	comp.sys.ibm.pc.hardware, comp.sys.mac.hardware, comp.windows.x	2932
3	comp.graphics, comp.os.ms-windows.misc, comp.sys.ibm.pc.hardware, comp.sys.mac.hardware, comp.windows.x	4890

In addition, the method of adding noise into the training text set in this chapter is that the text is randomly selected from each class according to a certain ratio firstly. Then, the selected text will be added into any class in the training text set as the noise data. Here, the noise ratio is the proportion of noise data in all the data of the text set. In the experiment, the values of noise ratio are respectively 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35 and 0.4. Next, in order to test that different noise ratios influence the classification result, the noise ratio will be increased little by little.

First of all, how the number of iterations K in the proposed algorithm of noise processing influences the effect of noise processing will be tested on the first group of data. Here, the precision is the proportion of correct samples in the text set after removing noise. As shown in Figure 4.5, the precision decreases slowly after more than three times of iterations with various noise ratios. The reason is that too many times of iterations will cause some information which is beneficial to classify after data smoothing to be lost, which affects the effect of classification. Based on this point, in order to save computational overheads at the same time, the maximum number of iterations is generally not more than three. Therefore, the number of iterations K is set to 3 in this experiment, and the ratio of noise reduction q is set to 0.4.

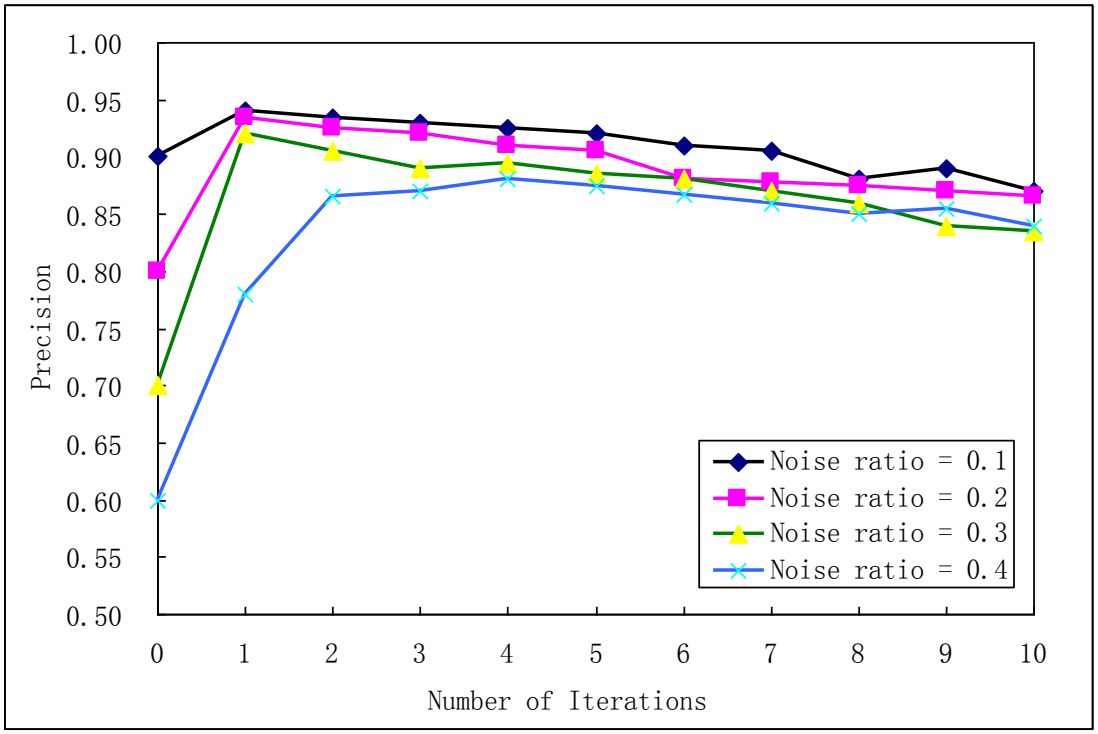

Figure 4.5: The relationship between the number of iterations and the effect of noise processing with different noise ratios.

In order to verify the performance and feasibility of the proposed strategy based on the LDA model, the following four methods are used as four contrast experiments, and SVM is chosen as the classification algorithm for them. First, the training data has no any noise processing, which is test 1. Second, the training data is only processed by the data smoothing method based on the generative process of the topic model, which is test 2. Third, the training data is only processed by the method of category entropy based on the LDA model, which is test 3. Fourth, all the noise samples are removed from the training data directly and accurately and other normal samples are kept fully, which is test 4. The fourth method is a benchmark reference method. Usually, its result is the best that an algorithm of noise reduction can achieve.

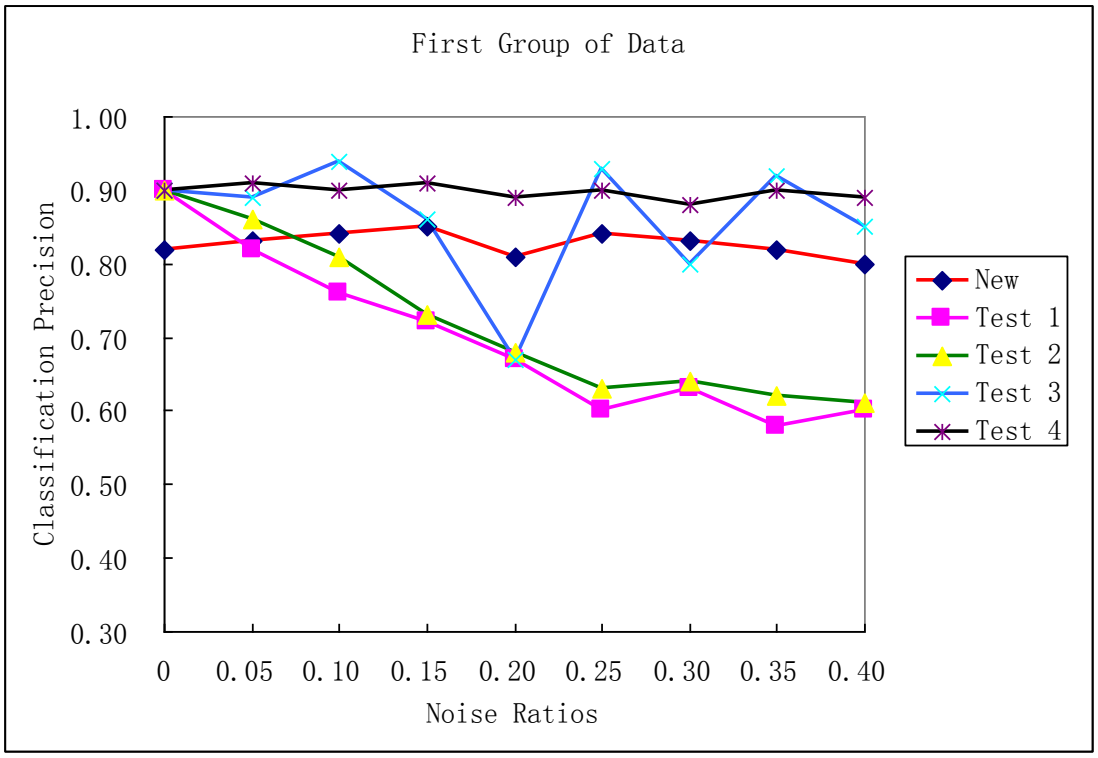

Figure 4.6: Classification results of different methods with various noise ratios in first group of data.

Figure 4.6, Figure 4.7 and Figure 4.8 show the classification results of different methods with various ratios of noise, where NEW is the proposed method in Section 4.3.3. In short, except for test 4, the performance of NEW is significantly better than other methods in the vast majority of cases because its classification precision is affected by the noise ratio weakly, and its performance is stable. Sometimes, the fluctuation of the classification precision of test 3 is large in the same data set. The reason may be that it removes some samples with the top of category entropy according to a certain ratio or threshold. If the threshold is set too large, some normal samples will be abandoned mistakenly. On the contrary, if it is set too small, the effect of noise reduction will be influenced.

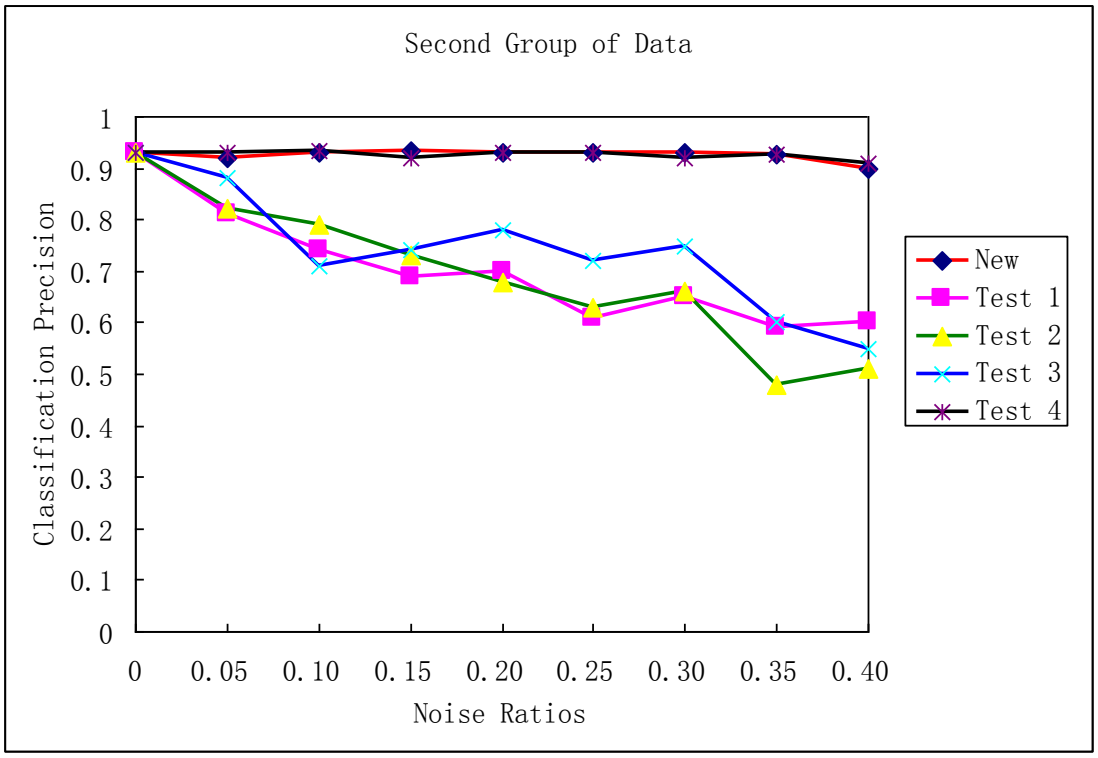

Figure 4.7: Classification results of different methods with various noise ratios in second group of data.

In Figure 4.6, Figure 4.7 and Figure 4.8, when the noise ratio is low, all the methods of noise reduction can guarantee good classification precisions. When the noise ratio is high, basically, the performance of NEW can still remain stable. But, the classification precision of test 1 declines rapidly, the classification precision of test 3 sometimes has large fluctuation.

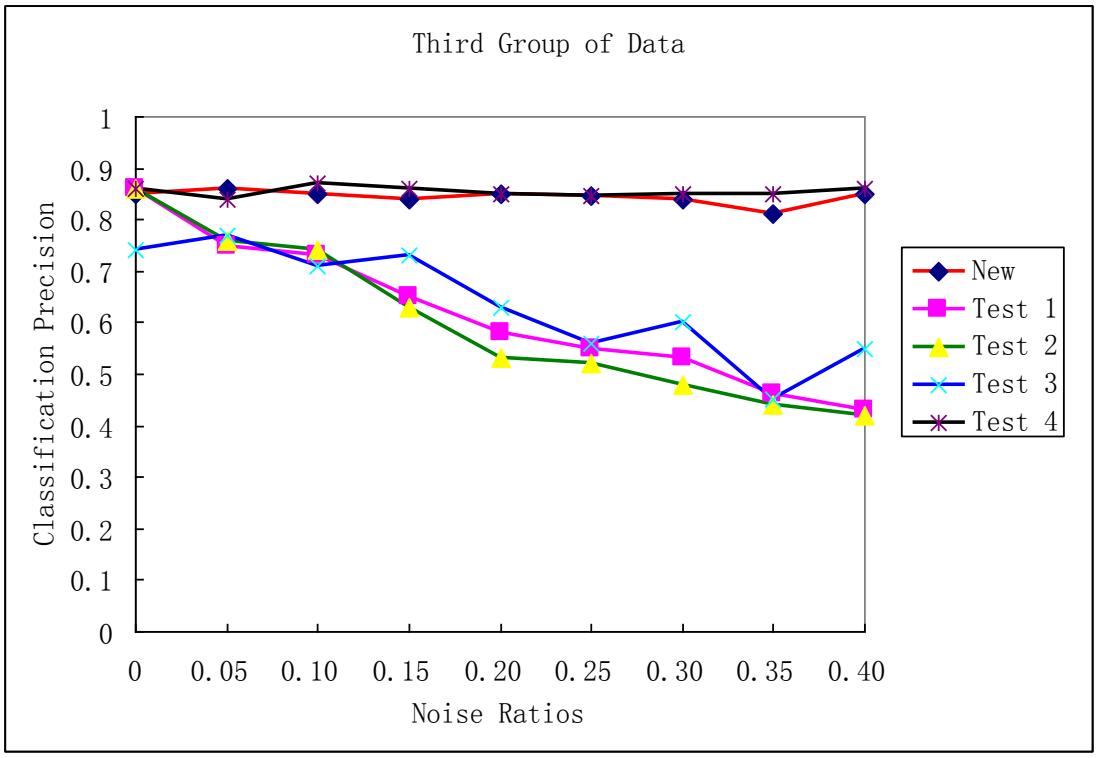

Figure 4.8: Classification results of different methods with various noise ratios in third group of data.

In addition, as can be seen from Figure 4.7 and Figure 4.8 that the classification precision of NEW sometimes is better than test 4. The reason may be that noise samples in training text set are removed by test 4, reducing the size of the training data. The size of training data in NEW is the same as the size of the original training data, which is able to fully reflect all information in the training data and is beneficial to improve the classification precision. Another reason is that there may be some amount of noise samples or the samples which can interfere with the classifier in the original data set. These samples also can be removed by the noise processing in NEW, enhancing the classification precision.

4.4 Summary

At the beginning, this Chapter introduced two current main methods of selecting the optimal number of the LDA model, the standard method of Bayesian statistics and the method of selecting the optimal number of topics based on HDP. Then, according to the idea which calculates the density of samples to measure the correlation among topics based on the density-based clustering algorithm, a method of selecting the optimal number of topics based on DBSCAN was proposed. Its feasibility and validity was proved by the related experiments. Besides, compared with the above two current methods, the proposed method can find the optimal number of topics in the LDA model automatically with relatively small number of iterations, which did not need to be assigned the number of topics manually.

Next, two current main LDA-based methods for noise processing were introduced, the data smoothing based on the generation process of the topic model and the category entropy based on the LDA model. This chapter combined the two methods, and then proposed a noise data reduction scheme. Its basic idea is to filter out noise samples according to the method of category entropy based on the LDA model firstly. And then, the generative process of LDA topic model was used to smooth the data. This strategy not only reduced the influence of noise samples to text classification, but also kept the size of the original training set. The experimental results showed that the proposed method can reduce the noise in the data set greatly. When the noise ratio in the training data set was large, its classification precision was still good. In addition, its classification precision was affected by noise ratio weakly, and its performance was stable.

Chapter 5 Genetic Algorithm based Static Load Balancing for Parallel Latent Dirichlet Allocation

5.1 Introduction

The topic models such as LDA has become the important tools in the field of text mining. It is convenient to query, classify and manage text by latent topic information. However, the traditional stand-alone LDA has been difficult to meet the real needs of massive data because of the limitation of storage and computing capacity [72] [117]. The development of clustered hardware system and parallel computing software technology has brought an opportunity to solve this issue. Parallel programming technology can deal with computing and storage problems well in the cluster, so parallelization of LDA is one of current main methods of processing large-scale data [45] [69]. In addition, load balancing is a common problem in parallel computing. To choose or design a reasonable scheduling strategy and related parameters is able to minimize synchronization waiting time among the processors in the heterogeneous cluster, improving the efficiency of parallel computing [130] [133].

This chapter begins with a brief introduction of four parallel LDA methods: Mahout's parallelization of LDA, Yahoo's parallel topic model, the algorithm of parallel LDA based on Belief Propagation (BP) and Google's PLDA. Then, the basic structure and working process of MapReduce on Hadoop parallel computing platform is introduced briefly. The distributed programming model of MapReduce is used to study the implementation of paralleling AD-LDA. Next, a static load balancing strategy based on genetic algorithm for PLDA is proposed, which can improve the performance of parallel program. Finally, the experiments prove that the efficiency of parallel LDA based on the MapReduce framework is higher than the traditional stand-alone model,

and the proposed load balancing strategy is feasible.

5.2 The Current Main Parallelization Methods of LDA

The essence of LDA is a serial algorithm, so the current parallel approaches inevitably use approximate assumptions to transform it to parallel LDA algorithm [72] [188]. Its basic idea is to make every computer in a cluster model and learn local files only, and transfer, exchange and update the required parameters by network.

At present, there are three main parallel programming technologies, such as shared memory model which is represented by OpenMP [49] [50], message passing model which is represented by Message Passing Interface (MPI) [50] and data parallel model which is represented by MapReduce [52] [66]. They can implement parallel computing of three LDA model parameters inference methods which are Variational Bayes (VB) [31] [32], Gibbs Sampling (GS) [107] [108] and Belief Propagation (BP) [106]. The followings are their parallel algorithms in detail.

5.2.1 Mahout's Parallelization of LDA

In the early phase, the research of parallel topic model focused on the algorithm of parallel variational EM [183]. Parallel sampling is carried out by the parallel execution of E-steps and M-steps. Considering that the computational overhead of E-steps is relatively larger and easier to be parallelized, Nallapati et al mainly parallelized E-steps to improve the efficiency of the algorithm [184]. Wolfe et al proposed the variational EM method which can parallel E-steps and M-steps at the same time, and have proved the effect in a series of network topology [185].

Mahout [186] is one of open source libraries for machine learning algorithms in the Apache project, which can process large-scale data. Its distributed algorithm is mainly

based on the Hadoop Map-Reduce architecture. The distributed LDAEM algorithm in [184] is used in Mahout. Its basic idea is to use variational inference and EM algorithm to approximate the posterior probability of latent variables. In E-steps, the update task of parameter γ and ϕ is assigned to multiple slave nodes. Each slave node will calculate the corresponding statistics according to the updated local parameter γ and ϕ, and return the statistics to master node. Finally, the master node will gather all the statistics and carry out the process of estimating model parameter α and β in M-steps.

The basic process of Hadoop which realizes Mahout's parallelization of LDA is described as follows [184] [187]. Firstly, the input text data should be formatted as the form of key-value pairs, where key stands for the document label and value represents the vector that is made up of words and their frequency. Then, the formatted data is stored in Hadoop Distributed File System (HDFS). HDFS can store the input data in different machines, but execute file management operations according to a unified logical address. In the map phase, E-steps in LDA based on variational inference are carried out in parallel, updating the variational parameters of each document and returning the intermediate results and the normalized items. In the reduce phase, the obtained intermediate results from the map phase will carry out simple vector addition according to values. At the end, the matrix ϕ is normalized to gain final topic-word probability parameter β. Here, it is assumed that the initial parameter α in each document is fixed so that it is not necessary to estimate the variational parameter γ. Logarithmic function is used to represent the intermediate results, which can convert complex operation of vector multiply-divide to simple operation of vector add-subtract, reducing the computational overhead.

5.2.2 Yahoo's Parallel Topic Model

The framework of parallel topic model [188] is a parallel framework of LDA, which was proposed by Yahoo Lab [189]. Its core idea is that the update speed of global statistics (topic-word matrix) and the update speed of local statistics (document-topic matrix) are different in the process of Gibbs sampling. So the global statistics is updated belatedly, reducing the overhead of passing parameters. And, the asynchronous update is adopted among the different machines, which can avoid that the machines with a fast sample rate wait for the machines with a slow sample rate [190].

The framework of Yahoo's parallel topic model divides the algorithm of parallelization into two parts [188] [190]: single machine multi-core parallelization and multi-machine parallelization. The basic idea of single machine multi-core parallel algorithm is that a single document relative to the massive text set affects the global statistic weakly, which is assumed. The global statistics can be updated belatedly after a single document finishing Gibbs sampling, which means that the global statistic is constant when a single document executes Gibbs sampling. Thus, the process of Gibbs sampling can be decomposed into multiple sub-processes. A large number of threads are simultaneously used to carry out Gibbs sampling in parallel to each word in each document by the pipeline process, decreasing the time overhead. The basic idea of multi-machine parallelization is that a local statistics and a shared global statistics are maintained in each machine. Each machine finishes Gibbs sampling of local document according to the local global statistics, asynchronously synchronizes the updated data into the shared global statistics of other machines. Therefore, the update of the global statistics can be completed asynchronously so that the algorithm can keep on the next iteration of Gibbs sampling without waiting for all the machines finishing the current round of Gibbs sampling and completing the update of the global statistics. It can enhance the efficiency of the algorithm's parallelization greatly.

The basic process of achieving Yahoo's parallel topic model is that [188] [191] the input text data is formatted as the form of key-value pairs firstly. And then, key-value pairs will be stored in HDFS. In single multi-core parallelization, the pipeline processing is used to carry out Gibbs sampling to a single document by Intel Threading Building Blocks (Intel TBB) [192], realizing the algorithm. Intel TBB is a multithreaded programming library based on C++, which makes users write software that can take full advantage of the performance of multi-core processor conveniently.

In multi-machine parallelization, the corresponding server and client are established in each machine by the Internet Communications Engine (ICE) [193], to transfer the global statistics and update local global statistics asynchronously. ICE is a content-oriented distributed tool, which was provided by ZeroC company. It provides the API of distributed programming. On the client-side, the data is stored in hash table. The operation request of Put&Get is launched to the server-side by the mechanism of Asynchronous Method Invocation (AMI). On the server-side, the data is stored in the distributed hash table.

5.2.3 The Algorithm of Parallel LDA Based on Belief Propagation

Jia Zeng and Newman used OpenMP and MPI to implement the algorithm of parallel LDA based on BP, which can deal with the bottlenecks of computation time and memory storage on the share-based memory [72] [106]. In brief, OpenMP is an extensible standard, which can support multiple programming languages, cross platforms. And, it provides users with a simple and flexible interface [49]. OpenMP uses the Fork-Join parallel model to achieve the parallelization of tasks. Data interaction is realized by accessing the same shared memory address. MPI is one of the most popular parallel programming mechanisms, which is used in parallel computers, workstations and clusters of distributed storage. MPI divides a large-scale computing task into many small parts first. Then, the small parts will be assigned to

different nodes of the cluster to achieve the parallelization of task. Finally, the model parameters are updated by the Map-Reduce synchronization or Receive-Send asynchronization to achieve the data interaction [50].

The basic idea of MPI implementing the algorithm of parallel BP is to segment the document-word co-occurrence matrix first. And then, the data will be assigned to all nodes of the cluster, obtaining local model parameters by using the algorithm of BP. At the end, the synchronization and updates of the global model parameters are realized by calling a library function MPI_Allreduce in MPI. The basic process of implementation is that D training documents are assigned to P processors in the cluster evenly so that each processor can gain D/P documents. Next, the parameters are initialized. The algorithm of BP is executed in each processor, obtaining local parameters θ and ϕ. After enough iterations, the model is converged at the end. Finally, θ is reduced by calling a library function MPI_Allreduce, gaining the global model parameters [50] [106].

The basic idea of OpenMP implementing parallel BP is to distribute large-scale data to multiple processors. Each process runs independently but shares the memory address, so the model parameters are global. Here, the model parameters are atomized so that it can avoid the access conflict to the same address effectively. The corresponding basic process of implementation is that the parameters are initialized first. Then, the text set is assigned to multiple threads to execute the algorithm of BP in parallel. The information content in each word that is in text of a single thread is calculated. Here, the information content which is carried by ϕ is atomized so that multiple threads will not read and write the same memory address at the same time. It can avoid the access conflict. Next, in order to avoid access conflict as well, the model parameters are updated and the information content which is carried by ϕ is atomized again. Finally, the above process is iterated until the algorithm is converged, obtaining the global model parameters [50] [106].

5.2.4 Google's PLDA

Newman et al proposed the parallel LDA model based on Gibbs sampling, Approximate Distributed LDA (AD-LDA) and Hierarchical Distributed LDA (HD-LDA) [72] [117]. Each thread calculates the local model parameters in AD-LDA, and then the global model parameters are updated by reduction. HD-LDA adds a super parameter in topic-word tier, and analyzes many aspects of the performance of the algorithm in detail, such as perplexity, convergence and speed-up ratio. Asuncion et al put forward an asynchronous parallel topic model, and gave the implementation of two main algorithms LDA and HDP [194]. Porteous et al proposed a fast Gibbs sampling method and the corresponding parallel LDA algorithm [71]. Wang et al proposed a parallel LDA algorithm based on MPI and MapReduce, and compared the features and results of the two implemented methods. In addition, the experiments proved its feasibility and efficiency [195].

PLDA is a parallel LDA algorithm, which was proposed by Google based on AD-LDA [195]. It can estimate the posterior distribution of latent variables by Gibbs sampling, and realize the parallel algorithm respectively on MPI and MapReduce distributed programming platforms. In AD-LDA, the word-topic distribution is initialized randomly, and the statistics of word-topic in the whole text set is computed first. Then, the input data will be assigned to different processors. The assigned part of the text set which is regarded as the whole text set is executed Gibbs sampling in each process. After all the processors finish the Gibbs sampling, the word-topic distribution in the whole text set will be gathered to get the global statistics. Finally, the obtained global statistics will become the initial value of the global statistics in the next iteration of Gibbs sampling.

In the implementation of PLDA based on MPI [195], each node will call the AllReduce API of MPI after finishing local Gibbs sampling. After finishing current iteration of Gibbs sampling in all nodes, the change information of the statistics in

each node will be transferred to other nodes, and be accumulated to the global statistics. Meanwhile, the change information of the statistics in other nodes and the updated global statistics are received, which will be regarded as the initial value of the global statistics in the next iteration of Gibbs sampling.

In the MapReduce version, the Hadoop parallel computing platform is used to implement PLDA [195]. In each iteration of Gibbs sampling, each mapper node carry out Gibbs sampling to local data, which is called Map phase. The changes of local statistics and the new topic which is assigned to each word are recorded. After all mapper nodes finishing local Gibbs sampling, the data is transferred among all the nodes, which is called Shuffle phase. The changes of local statistics as the intermediate results are transferred to reducer nodes, which is called Reduce phase. The reducer nodes gather the updates of all mapper nodes and work out the new global statistics, which will be regarded as the initial value of the global statistics in the next iteration of Gibbs sampling. After enough iterations, the global model parameters θ and ϕ will be obtained.

5.2.5 Analysis and Discussion

In Mahout's parallelization of LDA, multiple processors operate the shared variables at the same time result in read-write conflict, which makes the efficiency of parallel variational EM algorithm relatively low. Compared with Mahout's parallelization of LDA and Google's PLDA, Yahoo's parallel topic model can improve the efficiency of parallel computing greatly. But, its basic idea is to assume that to update the global statistics belatedly will not affect the final inference result, which means that the global statistics is constant in a single document sampling process. Unfortunately, the experiments proved that it sacrifices some accuracy when dealing with the real data set.

In fact, when parallel BP algorithm based on MPI deals with large-scale data, the proportion of communication time in each processor increases with the increasing number of processors. Meanwhile, the memory employ is large. Therefore, there is some limitation for its application in ultra-large-scale topic modeling. In addition, the parallel BP algorithm based on OpenMP works on shared memory. So it almost only occupies 1/P in the algorithm based on MPI, which reduces the memory consumption greatly. Here, P is the number of processors. Besides, it does not have the reduced communication time among the MPI threads, decreasing the calculation time greatly. However, all the threads run independently but share the memory address. So, multiple threads may carry out read-write operations on the same memory address at the same time, causing the memory access conflict. In addition, the algorithm based on OpenMP can only be used in the shared memory systems, namely, its scalability is not strong.

Table 5.1: The features comparison of implementing PLDA with MPI and MapReduce.

Distributed programming models	Communication mode	Fault-recovery
MPI	Through memory/network	Not supported
MapReduce	Through GFS/Disk IO	Built in

Table 5.1 shows the features comparison of implementing PLDA with MPI and MapReduce. Without considering computing failure of the processor, the efficiency of MPI-PLDA is higher than MapReduce-PLDA because the data of its iterative calculation is kept and added up in the memory, and the whole process does not need disk IO. But, if the number of processors and the size of processed data increase, the failures in some processors can not be ignored. MapReduce has the function of fault recovery but MPI does not support. Under the circumstances, once a machine fails,

the whole calculation process in MPI-PLDA will be forced to restart. Therefore, this chapter focuses on MapReduce-PLDA, and there will be a detailed research about its algorithm implementation in the next section. In addition, the overhead of disk IO is very large in MapReduce-PLDA because the map phase needs to record the status of the topics being assigned to words in the text set to the hard disk [117] [195].

5.3 Paralleling LDA with MapReduce/Hadoop

Based on the above analysis and comparison, the Hadoop distributed platform and the PLDA algorithm are used to study the implementation process of parallel LDA in this chapter. Gibbs sampling in LDA is a serial algorithm in essence, so to parallelize Gibbs sampling is based on the following assumption: there are m processors, and all the data can be divided into m parts. Each processor executes Gibbs sampling to local data set independently. As long as the global statistics in each processor is updated in time, the final inference result is approximate with the serial inference result. In other words, the effect of distributed sampling on the final result can be ignored, which has been proved in [188].

5.3.1 The Working Process of MapReduce on Hadoop

In the Hadoop parallel platform, its bottom is HDFS. It stores the files of all the storage nodes in a Hadoop cluster [63] [64]. The upper tier of HDFS is the MapReduce engine. MapReduce is composed of two user-defined functions, Map function and Reduce function. The input and output of Mapper and Reducer are stored in HDFS. Programmers only need to focus on specific computing tasks of the Map function and Reduce function, other complex problems in parallel calculation will be processed by the operation system background of MapReduce, such as partition of data set, job scheduling, fault tolerance and communication among nodes [65] [66]. A typical working process of MapReduce is shown as follows [52] [61] [64] [66]:

- Input phase: when a map task runs, the path of input and output and some other operating parameters should be indicated. The InputFormat class analyzes the input files, and splits the large data file into a number of independent data blocks. The data blocks are read in the format of key-value pairs < key1, value1 >.
- Map phase: A user-defined mapper class can carry out any operation on the key-value pair < key1, value1 >. After the operation, the OutputCollect class is called to generate a batch of new intermediate results in the format of key-value pairs < key2, value2 >. The type of key-value pair < key2, value2 > can be different from the input key-value pair < key1, value1 >.
- Shuffle and Sort phase: in order to ensure that the input of Reduce is the sequenced output of Map, the intermediate results with the same key are processed by one Reduce as far as possible. In the Shuffle phase, the intermediate results with the same key are gathered into the same one node as far as possible. In the Sort phase, the input of Reduce will be sorted by the model according to the value of key. Usually, the two phases of Shuffle and Sort are executed in parallel.
- Reduce phase: in all intermediate data, the user-defined Reduce function is carried out to each unique key. The key-value pairs with the same key will be merged, where input parameter is < key2, (list of values2) > and the output is new key-value pairs < key3, value3 >.
- Output phase: the results of Reduce are outputted to HDFS by calling the OutputCollect class.

5.3.2 The Algorithm and Implementation of PLDA

Every iteration of Gibbs sampling in PLDA [195] is regarded as a MarReduce job in Hadoop. When each data node carries out Gibbs sampling, two matrixes will be calculated. One is the word-topic count matrix C^{Word}, where each element C_{wk} represents the frequency of word w being assigned to topic k. Another is the

document-topic count matrix C^{Doc}, where each element C_{kj} expresses the frequency of words in document d_j being assigned to topic k. Briefly, when Hadoop is used to implement PLDA, Gibbs sampling is executed to local data in the Map phase and text distribution, topic distribution and C^{Word} are updated in the Reduce phase.

Hadoop assigns D training texts to P mappers, and each mapper deals with $D_p = D/P$ texts. The corresponding distributions of text W and topic Z are also assigned to P mappers, which is expressed as $\{W_{|1}, \cdots, W_{|P}\}$ and $\{Z_{|1}, \cdots, Z_{|P}\}$. Here, $W_{|P}$ and $Z_{|P}$ stand for that they are only processed in mapper p. Thus, the document-topic count matrix C^{Doc} is also distributed to each mapper, which is represented as $C_{|p}^{Doc}$. Each mapper keeps its own word-topic count matrix C^{Word}. At this moment, text is the key, text distribution and topic distribution are the value.

Each mapper executes Gibbs sampling to local data which is treated as the whole text set. Mapper updates the corresponding topic distribution of each text, and then output text distribution and topic distribution to Data Reducer. Text is key, text distribution and topic distribution are value. After the iteration of Gibbs sampling to all the documents, mapper will output ΔC^{Word} which is the update of C^{Word} to Model Reducer. Word is key, and ΔC^{Word} is value. Therefore, when PLDA is realized by the Hadoop parallel platform, two reducers are used to process and output the updated text distribution, topic distribution and C^{Word}. In the Reduce phase, the output of map is respectively transferred to Data Reducer and Model Reducer, which identify the input key and invoke different reduce algorithms accordingly.

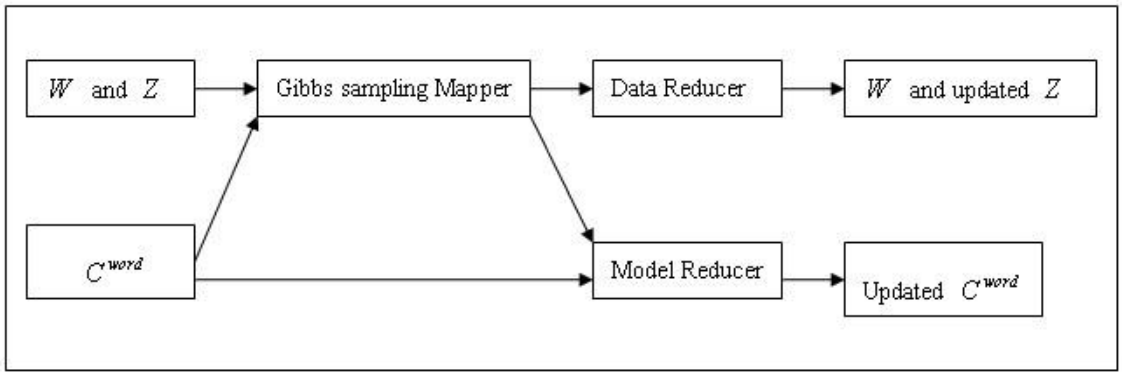

Figure 5.1: The framework of one Gibbs sampling iteration in MapReduce-LDA.

As shown in Figure 5.1, Data Reducer saves the output of map text distribution and topic distribution to HDFS after Gibbs sampling iteration. For each word, Model Reducer will gather and output the updated C^{Word} of this word. After all mappers completing Gibbs sampling, Data Reducer and Model Reducer will aggregate the intermediate results, obtaining the updated global statistics which is treated as the initial value of next iteration of Gibbs sampling. The input and output of the MapReduce job in PLDA are the same in terms of format, both containing text distribution, topic distribution and word-topic count matrix C^{Word}. So the Gibbs sampling iteration of PLDA in Hadoop can be ensured, namely each output of reduce can be the input of map.

5.4 A Static Load Balancing Strategy Based on Genetic Algorithm for PLDA in Hadoop

In the algorithm of PLDA, after each data node finishing Gibbs sampling to local data, data exchange will be executed to achieve synchronization of parallel algorithms. If the load of each node is imbalanced, the nodes with a small load will wait for the

nodes with a large load, which will influence the parallel efficiency. Therefore, this chapter proposes a static load balancing strategy based on a genetic algorithm for PLDA in a heterogeneous cluster. Data partitioning is executed before the start of the task. According to the divisible load theory, the synchronization waiting time among data nodes should reduce as far as possible, enhancing the performance of parallel execution [127] [128] [129] [131] [133].

5.4.1 The Algorithm Design

First of all, in Hadoop, the total time T of a mapper processing a data block includes the following four parts:

- Data copying time t_c: it is the time of a data block being copied from HDFS to a local hard disk. In reality, it depends on the realistic network bandwidth and writing speed of the hard disk.
- Data processing time t_p: it is the computing time of the mapper to process a data block.
- Merging time of intermediate data t_m: in the Shuffle phase, it is the operating time that the intermediate data which is outputted by mapper with the same key is merged to transfer to reducer as the input.
- Buffer releasing time t_b: it is the time of releasing the filled buffer until empty.

Therefore, $T = t_c + t_p + t_m + t_b$. (5.1)

Then, before the algorithm design, the following assumptions are made. D_s is the size of the data block. H_w is the writing speed of the hard disk, and its unit is MB/S.

B_n is the realistic network bandwidth, and its unit is MB/S. P_s is the speed of the processor running the mapper process, and its unit is MB/S. B_s is the size of the buffer of the mapper. R_a is the ratio of the size of intermediate data to the size of the original data block. N_f is the frequency of processing the intermediate data. N_b is the number of times of the buffer being filled up. V_b is the amount of the processed data when the buffer is filled up. S_h is the sort factor of Hadoop.

Thus, $t_c = \frac{D_s}{\min(H_w, B_n)}$ (5.2)

where t_c depends on the practical hard disk and network bandwidth. Their speed will affect the time of the data being copied from HDFS to the local hard disk of the mapper. In addition, it is obvious to gain Equation (5.3).

$$t_p = \frac{D_s}{P_s} \quad (5.3)$$

When a buffer is being filled, the processor needs to continue to write the intermediate data to the buffer, and meanwhile the process of releasing maintains that the sorted data is written to the hard disk from the buffer. So, the speed of filling a buffer can be expressed as $P_s \times R_a - H_w$. Then, the time of a buffer being filled up is $\frac{B_s}{P_s \times R_a - H_w}$. As a result, when a buffer is filled up, the processor will generate the intermediate results with the size of V_b which can be represented as Equation (5.4).

$$V_b = P_s \times R_a \times (\frac{B_s}{P_s \times R_a - H_w}) \quad (5.4)$$

The total amount of intermediate data being generated from the original data block

with the size of D_s can be expressed as $D_s \times R_a$. Therefore, the number of times of a buffer being filled up can be expressed by Equation (5.5).

$$N_b = \frac{D_s \times R_a}{V_b} \tag{5.5}$$

If the time of a buffer being released once is $\frac{B_s}{H_w}$, the time of a buffer being released for N_b times will be $N_b \times \frac{B_s}{H_w}$. So,

$$t_b = N_b \times \frac{B_s}{H_w} \tag{5.6}$$

The frequency of processing the intermediate data N_f can be defined by Equation (5.7).

$$N_f = \frac{N_b}{S_h} - 1 \tag{5.7}$$

When the intermediate data begins to be merged, the time of all the intermediate data being written into the hard disk is $\frac{D_s \times R_a}{H_w}$. If the process of merging happens N_f times, the operation time of disk IO will be $N_f \times \frac{D_s \times R_a}{H_w}$. Therefore,

$$t_m = N_f \times \frac{D_s \times R_a}{H_w} \tag{5.8}$$

In Hadoop MapReduce, the total time of processing data blocks in one processing wave T_{all} depends on the maximum consumed time of k participating mappers, which can be shown in Equation (5.9).

$$T_{all} = \max(T_1, T_2, \cdots, T_k) \tag{5.9}$$

because the finished processing mappers have to wait for other mappers to complete

processing.

In order to realize the minimization of T_{all}, according to the divisible load theory [132] [133] [134], it is assumed that all the mappers which are involved in the calculation complete local data processing at the same time. In other words, it can be expressed as the following Equation.

$$T_1 = T_2 = \cdots T_k \tag{5.10}$$

5.4.2 The Design and Implementation of Genetic Algorithm

A genetic algorithm is a self-adapting and global optimizing probability search algorithm, which can make computer simulate the genetic and evolutionary behaviors in living nature [122]. The basic idea of this algorithm is that the initial population is composed by coding first. According to the fitness function, the genetic operators such as selection, crossover and mutation are used in the evolution of population. Finally, the global optimal solution will be obtained [197] [198].

5.4.2.1 Encoding Scheme

When solving practical optimization problems, genetic algorithms can process chromosome (individual) in the form of genes string. Thus, a genetic algorithm has to convert the solutions of an optimization problem into the form of chromosomes. In fact, the coding process is that the problem space is mapped to the chromosome space [199].

Usually, genetic algorithms adopt binary coding. Its advantages are simple operations of coding and decoding, which will be beneficial to achieve crossover and mutation operations. However, when Hadoop is used to implement PLDA in a heterogeneous computing environment, a huge amount of text data will be processed. In this case,

binary coding makes the length of chromosome too long and requires too much memory, which will influence the performance of the whole computing platform. In addition, binary coding cannot reflect the inherent structure of the problem directly. When it is used to solve the high-dimensional optimization problem, the accuracy is not high.

In order to overcome the disadvantages of binary coding, this chapter uses decimal coding. It can treat the size of the data block which is assigned to each mapper as chromosomes so as to execute genetic operations directly.

5.4.2.2 The Initialization of Population

Usually, the size of population is set from 20 to 200 [125]. After many experiments, the size of population is set to 100 in this chapter. After setting the size of population, its initialization process is to generate 100 chromosomes randomly. This chapter uses decimal coding rather than indirect coding, so the length of each chromosome and the value of genes depend on the assigned data block from the Hadoop platform to each mapper.

5.4.2.3 Fitness Function

Generally, converting an objective function into a fitness function should follow two principles [120] [122]:

- The fitness must be non-negative.
- The change direction of the objective function in the optimization process is the same with the change direction of the fitness function in the group evolution process.

According to the algorithm design in Section 5.4.1, the fitness function of Hadoop-based parallel LDA can be expressed as Equation (5.11).

$$f(T) = \sqrt{\sum_{i=1}^{k} (\overline{T} - T_i)^2} \tag{5.11}$$

where T_i stands for the processing time of the i-th mapper. $\overline{T}$ means the average time of k mappers processing data, which can be expressed as $\overline{T} = \frac{\sum_{i=1}^{k} T_i}{k}$.

As long as the value of fitness $f(T)$ of the individual is smaller than a given fitness threshold, which means that k mappers almost complete local data processing at the same time, the iteration of the genetic algorithm will be converged and end. Otherwise, the obtained new generation of population will replace the previous generation, and return to the genetic operations to continue the loop execution until the iterative convergence. In addition, the maximum number of generations for the algorithm can be set as another iterative termination condition, avoiding that the algorithm cannot converge. As long as the iterations reach the given maximum number of generations, the genetic algorithm will be terminated.

5.4.2.4 Crossover

The crossover operator of genetic operations is the main method to generate new individuals, which determines the global search ability of genetic algorithm and plays a central role in genetic algorithm. Crossover is an operation that part of genes string in two parent individuals is exchanged and recombined to generate new individuals in some way [124]. Crossover operator can maintain the features of excellent individuals in the original population in a certain extent, and it also can make the algorithm explore the new gene space. So the new individuals are generated in the next generation, maintaining the diversity of individuals in the new population [200].

This chapter adopts single-point crossover, which is often used in simple genetic algorithms. The specific operations are as follows: the crossover probability P_c is set

to 0.5 in this chapter. According to this P_c, two individuals are selected randomly from the population as parent individuals. If the length of the gene string is L, and a crossover point is determined randomly, there may be L-1 crossover locations. When the chromosomes cross, the code value of the crossed genes in two selected individuals will be exchange and two new individuals will be generated. But, the new individuals may not be all retained in the next generation. They need to be compared with the original parent individuals. If the new individuals can adapt the fitness function better, they will be kept in the next generation.

5.4.2.5 Mutation

The basic idea of a mutation operator is that the genetic values of some locations in the individual gene string in the population are changed to generate a new individual. Mutation is a kind of local random search methods [200]. When it is combined together with a crossover operator, some permanent loss of information caused by the crossover operator can be avoided and the effectiveness of genetic algorithms can be guaranteed. Besides, it can make the search avoid falling into local optimum [201].

Furthermore, in order to prevent some genes in the population from being stuck in a constant state, mutation operation is required [122]. It makes genetic algorithms maintain the diversity of the population, ensuring that the population can continue to evolve and preventing the premature convergence [200]. Therefore, mutation in genetic algorithms is a helper method to generate new individuals, and an effective measure to avoid premature convergence of the genetic algorithm [118] [119].

Crossover needs two parent chromosomes to match, but mutation only needs one parent chromosome. In general, for each genetic value of the offspring individuals generated in crossover operation, a pseudo-random number between 0 and 1 is produced. If it is smaller than P_m, mutation operation will be executed. The basic

steps of simple mutation operation applied in this chapter are as follows:

Step 1: Mutation probability P_m is set to 0.01, to mutate the genetic value of the above gene.

Step 2: If the individual mutates, the genetic value of its gene will be replaced by new value, to generate a new individual.

5.4.2.6 The optimal retention strategy

This chapter adds the optimal retention strategy into simple genetic algorithm, to prevent that the highest fitness individuals may be destroyed in the process of crossover and mutation. Before crossover and mutation, two individuals with the highest fitness in the current generation are maintained, and they do not participate in genetic operations. After individuals in the current generation complete genetic operations, the two maintained individuals are used to compare and replace two individuals with the lowest fitness in the population, which can ensure better population evolution and improve the computational efficiency of genetic algorithms.

5.5 Experiment and Analysis

5.5.1 Evaluating PLDA in Hadoop

In this section, the Hadoop parallel framework is used to be the experiment platform. But, because the experimental condition is limited, such as the cluster is very small and the computing capacity is insufficient, the test data set is not large but it is real. The purpose of the experiment is to verify the feasibility and effectiveness of the PLDA algorithm, and find the load balancing problem. So, this section does not involve the genetic algorithm based load balancing strategy.

5.5.1.1 The Experimental Environment

Due to the limited experimental condition, nine ordinary desktop computers are used to constitute a Hadoop cluster in the local area network. Hadoop platform is the Master/Slave structure, so one of them is used as the master node to do the work of NameNode and JobTracker. And, other eight machines are the slave nodes to do the work of DataNode and TaskTracker.

In a homogeneous environment, for each desktop computer, CPU is Intel(R) Core(TM) 2 Quad Q6600 2.40 GHz, ram is 4GB, and the hard disk is SATA 500GB with 7200rpm. In a heterogeneous environment, based on the homogeneous environment, the hardware of two slave nodes will be changed. For example, CPU is Intel（R） Pentium(R) IV 2.8GHz, ram is 1GB, and hard disk is SATA 80GB with 5400rpm. Furthermore, the details about the configuration of the experimental environment are shown in Table 5.2.

Table 5.2: The configuration of the experimental environment.

Network bandwidth	1000Mb/s
Operating system	Linux Fedora 12
Experiment platform	Hadoop version 0.20.204
Java execution environment	JDK 1.6.25
Integrated Development Environment（IDE）	Eclipse (Linux version)

In the above hardware system, the specific steps of building the Hadoop platform are described as follows [189] [196]:

Step 1: The operating system Linux Fedora 12 is installed in the nine desktop computers.

Step 2: As shown in Table 5.3, each computer is named and is connected to the switchboard, where hadoop1 is the Master node and others are Slave nodes. The file /etc/hosts of each machine is modified to ensure that the host name of each machine and the corresponding IP address can be analyzed correctly with each other. For example, in the Namenode, namely in Master hadoop1, the IP address of all the computers in the cluster and the corresponding host names need to be added into the hosts file. In a Datanode, namely in other eight Slave nodes, the local IP address, IP address of Namenode and its corresponding host name need to be added into the hosts file.

Table 5.3: The configuration of nodes in a Hadoop cluster.

Computer name	IP address	Function
hadoop1	192.168.1.100	Master
hadoop2	192.168.1.101	Slave
hadoop3	192.168.1.102	Slave
hadoop4	192.168.1.103	Slave
hadoop5	192.168.1.104	Slave
hadoop6	192.168.1.105	Slave
hadoop7	192.168.1.106	Slave
hadoop8	192.168.1.107	Slave
hadoop9	192.168.1.108	Slave

Step 3: JDK is the Java kernel to run Hadoop, so JDK 1.6 or later version has to be installed in Linux before Hadoop is installed in each machine. This experiment chooses and installs jdk-6u25-linux-i586-rpm.bin, namely JDK 1.6.25.

Step 4: In each computer, the Eclipse integrated development environment with

MapReduce Plugins is installed, which can develop the MapReduce program, configure the Hadoop cluster file, and operate HDFS very conveniently.

Step 5: Generally, SSH (Secure Shell) service is installed when installing Linux. In the Hadoop cluster, both NameNode and JobTracker use SSH to start and stop the daemon process of each node, so for the convenient operations, SSH starting automatically when starting up is set and SHH logging in without password is configured in each node.

Step 6: Hadoop is installed in each machine in the cluster, and its version is 0.20.204. According to the introduction in the Hadoop project official website, the configuration files of the Hadoop cluster are modified. For instance, JDK path of hadoop-env.conf is modified and hadoop-site.conf which is the main configuration file of Hadoop is configured.

Step 7: After completing configuration, all the configuration files are copied into all nodes in the cluster. At this moment, to set up a Hadoop cluster environment is finished.

Moreover, in the experiment of evaluating PLDA, the default configuration is used in Hadoop. Here, the default FIFO scheduler is used to be the Hadoop scheduler. Each slave node can run up to 2 Map tasks at the same time. Accordingly, each node is able to run at most 2 Reduce tasks at the same time.

5.5.1.2 Experimental Data

This experiment uses the real text data set 20 Newsgroups to be the experimental data. It is composed of 20 different classes of news, and collected 18828 samples. All the samples are respectively in 20 folders, and the size of the total file is 32.8 MB [164]. At present, 20 Newsgroups has been widely applied to the text application

technologies with machine learning, including text classification and text clustering [163]. In the text set, text preprocessing and the parameter setting for implementing the LDA algorithm are the same with the processing operations in Chapter 3.

5.5.1.3 The Experiment in the Homogeneous Environment

Test 1: To verify the feasibility and effectiveness of Hadoop implementing the PLDA algorithm.

The text set of 20 Newsgroups is real, but the size of the data set is small. In order to test the computing performance of Hadoop realizing the PLDA algorithm in different number of nodes, the data set with 640MB is constructed by copying. In the whole Hadoop cluster, the nodes can be added or removed very easily because of the scalability of Hadoop. 1 to 8 data nodes are used to process the above constructed data set respectively.

Figure 5.2 shows the computing time of different number of data nodes dealing with the same amount of documents. As shown in Figure 5.2, along with the increase of data nodes, the computing time of processing the same data reduces. To increase the number of data nodes can significantly improve the computational efficiency of the Hadoop cluster processing the same size of data. In addition, NameNode and JobTracker are only responsible for task scheduling, and do not participate in the calculation, thus the moved nodes do not include NameNode and JobTracker. Therefore, the feasibility and effectiveness of the PLDA algorithm have been improved again in Hadoop.

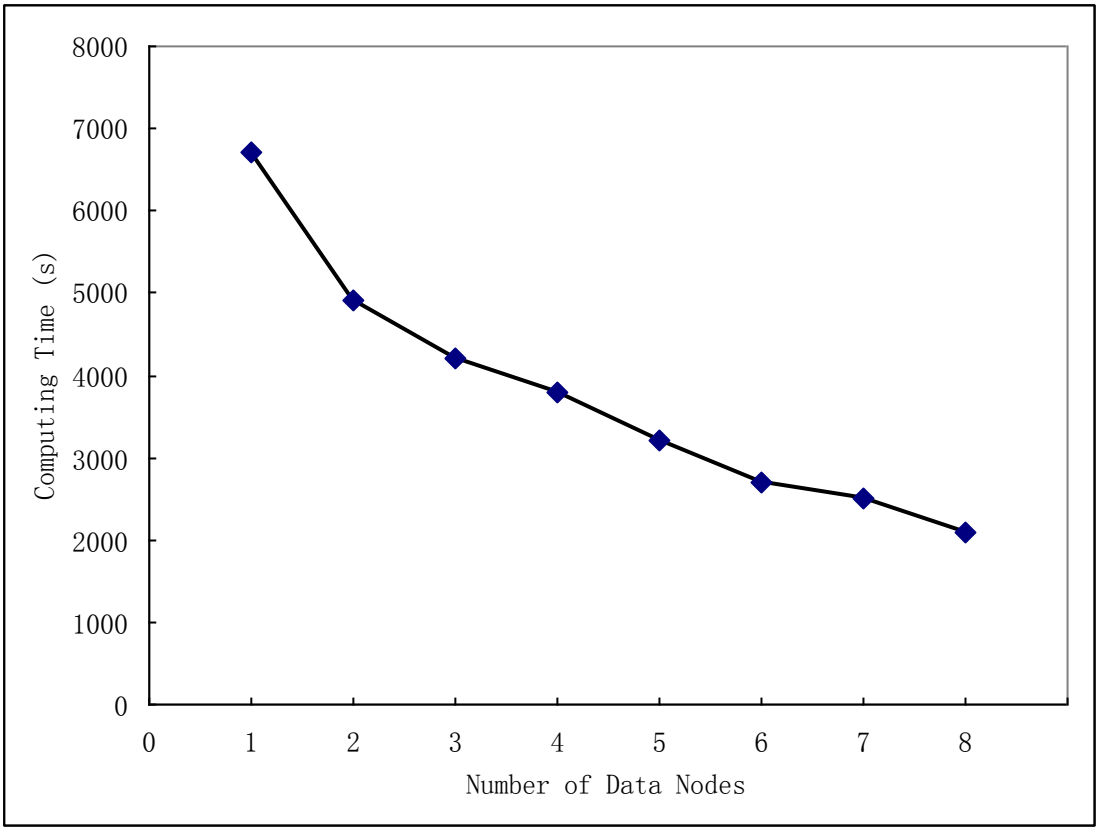

Figure 5.2: The computing time with different number of data nodes in a homogeneous cluster.

Test 2: The comparison of single machine and one data node in the cluster processing the data.

In the same hardware and software configuration environment, the performance of single machine and one data node in the Hadoop cluster will be compared when they process the same scale of data. In the contrast experiment, the size of data set is increasing gradually, such as 1.1MB, 2.7MB, 5.3MB, 16.2MB, 32.8MB and 53.7MB，so that cross-validation is executed successfully.

Table 5.4: The experimental result of single machine and one data node in the cluster processing the data.

Times of experiments	File size (MB)	The number of sample documents	The overhead of a single machine running serial LDA (S)	The overhead of one data node running parallel LDA in a Hadoop cluster (S)
1	1.1	320	14	132
2	2.7	780	46	146
3	5.3	1600	109	173
4	16.2	6400	378	208
5	32.8	18828	Reporting insufficient memory	253
6	53.7	29500	Reporting insufficient memory	335

Table 5.4 shows that when the single machine runs the serial LDA algorithm, with the growth of the input data, the computing time increases rapidly and the algorithm performance decreases significantly. The reason is that with the increase of the input data, the consumption of memory and other resource is too large on the single machine, resulting in the machine performance declines sharply, even until reporting that the memory is not enough to complete computing tasks.

However, the Hadoop cluster can complete the computing tasks, which means that the PLDA algorithm has the ability of dealing with large-scale data in the Hadoop cluster. But when the data set is small, the processing efficiency of PLDA is much lower than the processing efficiency of the single serial LDA in the Hadoop cluster. Because the start and scheduling of Job, the communication among the nodes and disk IO need to consume a certain amount of time, and its proportion of the total consumption is large. The actual computation time in the proportion of the total consumption is small.

In addition, a large number of small data files make the storage efficiency of HDFS low, and its retrieval speed is slower than that with the large files. When running MapReduce operations, the data is assigned to each mapper according to block in Hadoop. These small data files will consume much computing power. Thus, the PLDA algorithm is designed to process the large data set which the single machine cannot handle. Meanwhile, it is also proved that the Hadoop platform is suitable for processing large-scale data.

5.5.1.4 The Experiment in the Heterogeneous Environment

In order to test the performance of the algorithm, the constructed and preprocessed text data of 20 Newsgroups is divided into five different sizes of data sets, such as 300MB, 400MB, 500MB, 600MB and 700MB. Eight data nodes in the heterogeneous and homogeneous cluster respectively execute contrast tests to the above five different sizes of data. Their specific configuration is as shown in Section 5.5.1.1.

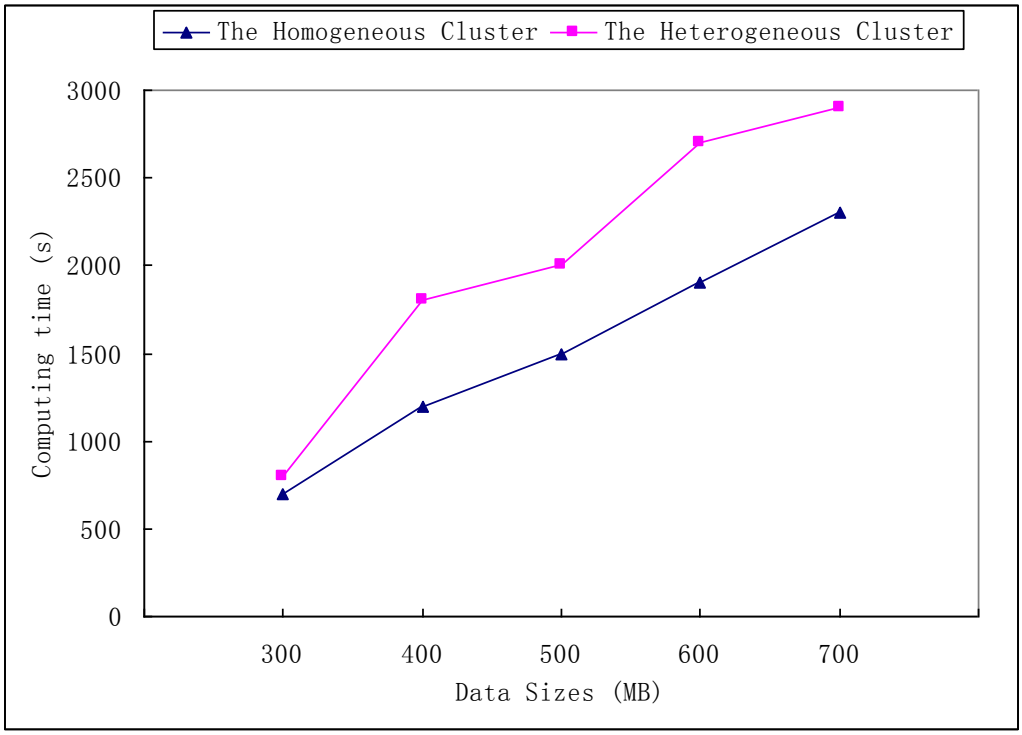

Figure 5.3: A comparison of computing time of dealing with different sizes of data with eight data nodes.

Figure 5.3 shows the running time of processing different sizes of the text set when the same number of data nodes is used in the homogeneous and heterogeneous cluster. It can be seen that the total running time of the homogeneous cluster increases basically linearly with the increase of the data set. On the other hand, the total running time of the heterogeneous cluster is not stable enough, a little fluctuation, but the PLDA algorithm still can be achieved in the heterogeneous cluster.

In addition, the computing efficiency of the homogeneous cluster is better than that of the heterogeneous cluster when processing the large data set. The reason is that the default scheduling approach of the Hadoop platform is built on the homogeneous cluster. And, when PLDA is realized in Hadoop, the data set is distributed equally according to the number of mappers in the cluster. When the heterogeneous cluster deals with a large data set, especially the number of nodes and the performance difference among the nodes are large, the load imbalance will be generated. The nodes with strong performance and fast speed of processing have to wait for the nodes with poor performance and slow speed of processing, affecting the computational efficiency of the whole cluster.

5.5.2 The Experiment of Load Balancing in a Simulated Environment

The extra overhead generated from task scheduling in a Hadoop cluster is inevitable, and its proportion in the total overhead is not small. However, if there are dozens or even hundreds of nodes in the Hadoop cluster, the proportion of the extra scheduling overhead in the total overhead will reduce to a very low and acceptable level. In addition, if the amount of data is greater and the number of data nodes is more in the Hadoop platform, its advantages will be more obvious.

Due to the limited experimental conditions, the number of nodes in the cluster is small and the size of the test data is not large. So, it is difficult to show the advantages of the

Hadoop platform fully, and the load balancing problem is not obvious. Therefore, HSim [203] is used to simulate the large-scale cluster processing massive data in this section, verifying the effectiveness of the static load balancing strategy based on the genetic algorithm.

HSim is a simulator, which can simulate a Hadoop environment [202] [203]. Its structure, basic idea and working process are the similar with Hadoop. Its aim is to simulate the behavior of the Hadoop framework accurately. Briefly, the framework of Hadoop is modeled and simulated by HSim in four aspects, such as node parameters, cluster parameters, Hadoop system parameters and HSim's own parameters. It can support the simulation of homogeneous and heterogeneous Hadoop computing environment. Besides, the performance of HSim has been verified by the published benchmark results and the published and customized MapReduce applications. The experimental result showed that HSim has high accuracy and stability when simulating the Hadoop applications. Therefore, HSim is able to simulate a variety of MapReduce jobs.

However, HSim is not perfect, and also has some limitations. In a simulated large cluster, the performance of HSim will be affected. But in a simulated cluster within 100 nodes, HSim can work well [203]. Thus, in order to evaluate the load balancing algorithm, a cluster with 40 nodes is simulated in this simulation experiment.

A high-performance computer is used to run HSim. Its hardware configuration is as follows: CPU is Intel（R）Core（TM）i7-880 up to 3.73 GHz，RAM is 16GB and hard drive is 2TB SATA with 7200rpm. Its software configuration is the same with Section 5.5.1.1. Table 5.5 shows the configuration of the simulated Hadoop environment when evaluating the performance of the load balancing strategy for PLDA.

Table 5.5: The configuration of the simulated Hadoop environment.

Number of simulated data nodes	40
Number of processors in each node	1
Number of cores in each processor	2
Number of hard disk in each node	1
Number of mappers in each node	2
Number of reducer in each node	2
The processing speeds of processors	Depending on H(d)
Reading speed of hard disk	80MB/s
Writing speed of hard disk	40MB/s
Sort factor	100

In the simulated Hadoop environment, the processing speed of data node depends on the heterogeneity of the Hadoop cluster. The total computing capacity of the cluster can be expressed as $D_t = \sum_{i=1}^{n} d_i$, where n stands for the number of data nodes in the cluster. d_i represents the processing speed of the i-th data node. If the total computing capacity of the Hadoop cluster is D_t, the heterogeneity of the Hadoop cluster can be defined as Equation (5.12).

$$H(d) = \sqrt{\sum_{i=1}^{n} (\bar{d} - d_i)^2} \tag{5.12}$$

where $\bar{d}$ means the average processing speed of all the data nodes in the Hadoop cluster, which can be shown as $\bar{d} = \frac{\sum_{i=1}^{n} d_i}{n} = \frac{D_t}{n}$.

Test 1: For the same data, the performance comparison with different heterogeneities.

First of all, the data with the size of 10GB is tested with the different heterogeneity in the simulated Hadoop cluster, namely the value of $H(d)$ is from 0 to 2.41. Figure 5.4 shows that when the value of $H(d)$ is less than 1.21, the simulated cluster is close to a homogeneous environment. In the PLDA algorithm, the performance difference between the static load balancing strategy based on genetic algorithm and the default FIFO scheduling in Hadoop is not obvious. But with the increase of the value of $H(d)$, namely the increase of the heterogeneity of the simulated cluster, the proposed strategy can reduce the overheads greatly and improve the computational efficiency.

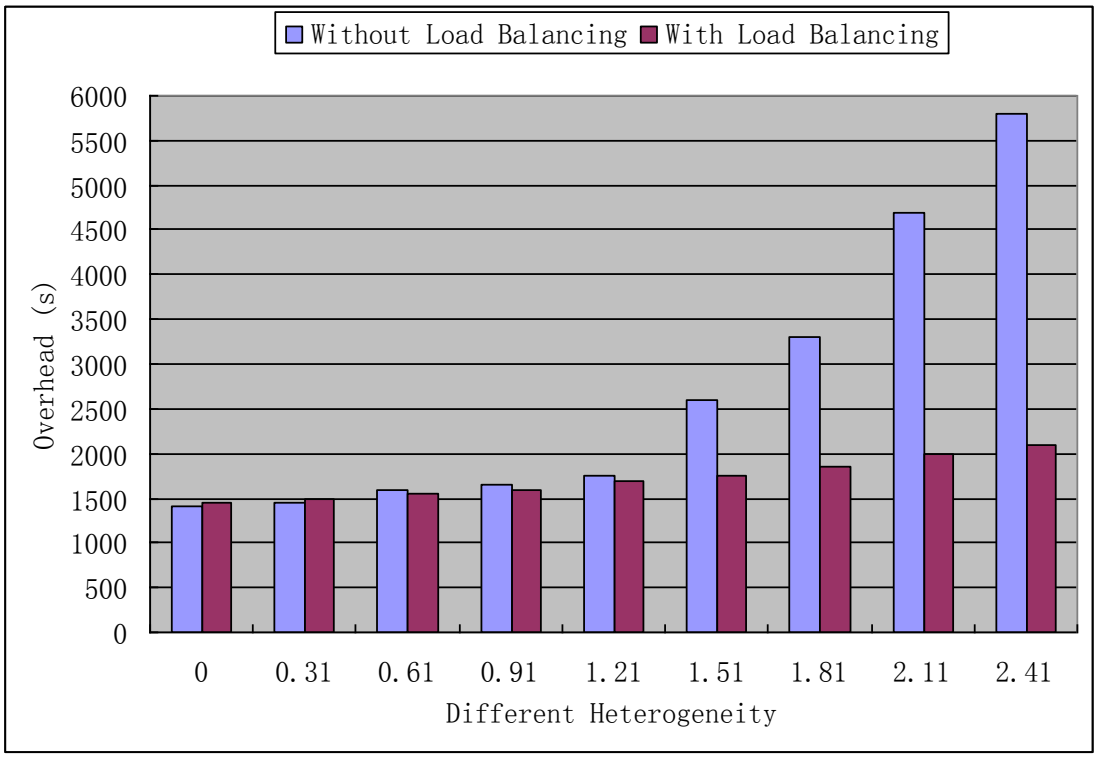

Figure 5.4: The performance comparison of the load balancing strategy with the different heterogeneity in a simulated environment.

Test 2: For the same heterogeneity, the performance comparison with different sizes of data.

The size of the data set is from 10GB to 100GB when $H(d)$ is set to 2.11. This test is to evaluate the performance of the proposed strategy dealing with different sizes of data sets. As shown in Figure 5.5, with the increase of the size of the data set, its effect is not very significant. But, compared with the default FIFO scheduling in Hadoop, it can reduce the overheads of the PLDA algorithm. The reason is that the proposed static load balancing strategy is based on genetic algorithm. It also needs some computing time when running. Furthermore, with the increase of the processed data set, the overhead of the load balancing strategy itself is also on the rise.

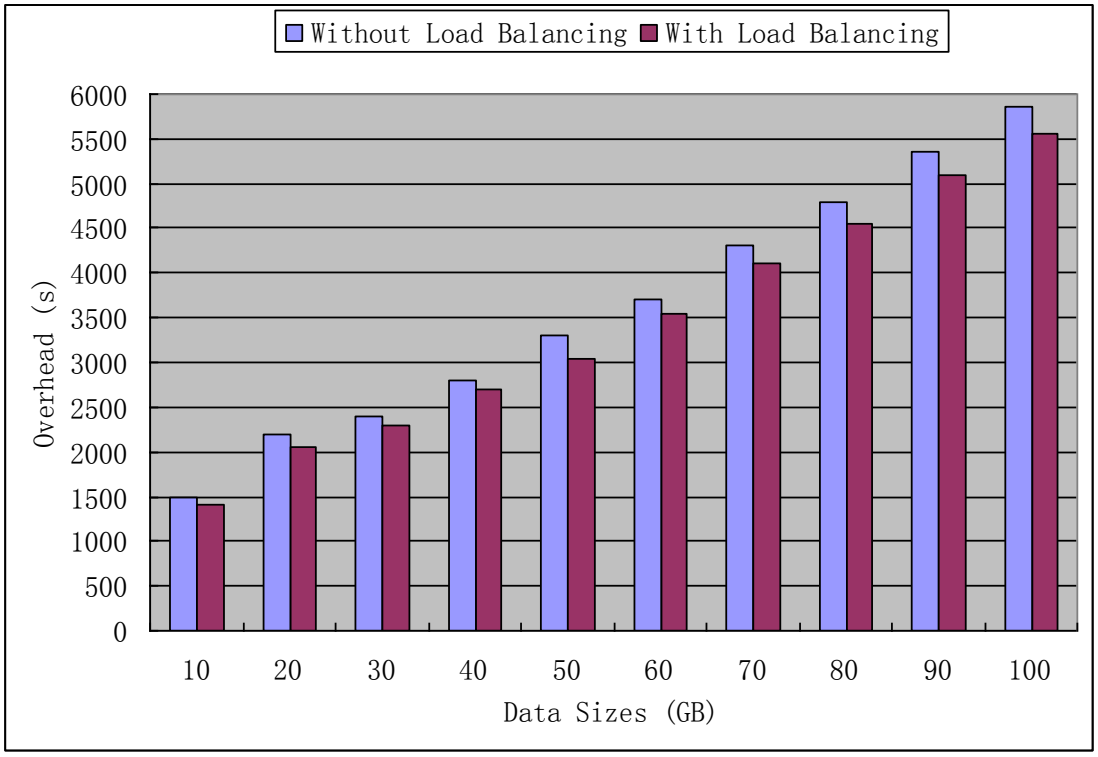

Figure 5.5: The performance comparison of PLDA with different sizes of data in a simulated environment.

However, the PLDA algorithm still can get benefits from the load balancing strategy. For example, when $H(d)$ is 2.11, the overhead of the proposed strategy itself is 673s. The overhead of one processing wave of mappers with load balancing is 1408s. But, the overhead of one processing wave of mappers without load balancing is 4711s. So, the algorithm performance improves by 55.8%. Therefore, in a heterogeneous environment, the proposed static load balancing strategy based on the proposed genetic algorithm is effective.

Test 3: The stability of the static load balancing strategy.

In fact, the convergence of genetic algorithm in the proposed static load balancing strategy will also affect the efficiency of the PLDA algorithm in Hadoop. In order to analyze the convergence of the genetic algorithm, the overheads and the evolutionary generations of PLDA processing a data set with the size of 10GB are tested in the simulated Hadoop environment.

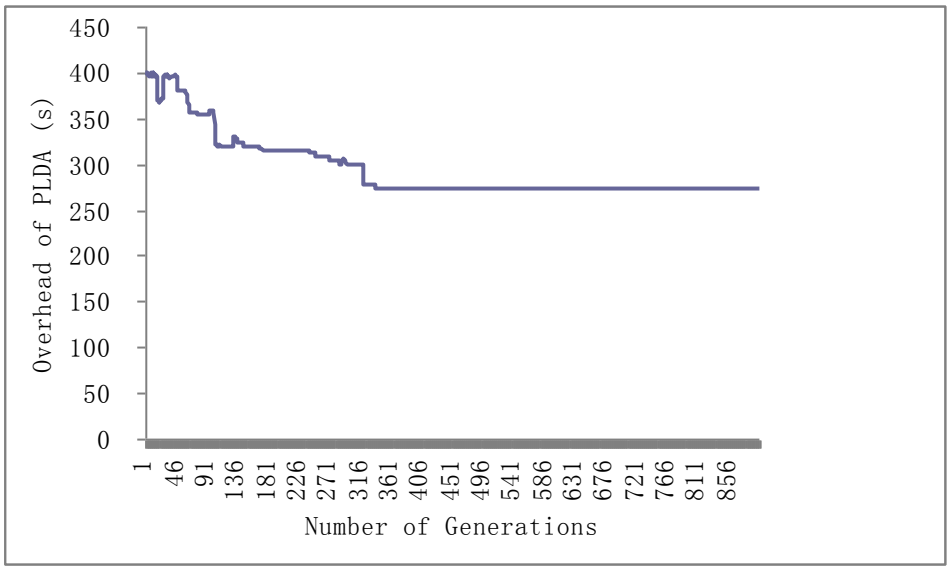

Figure 5.6: The convergence of genetic algorithm in the load balancing strategy.

As shown in Figure 5.6, the overhead of PLDA is long in the early evolution, and there are significant fluctuations. With the increase of the evolutionary generations, the genetic algorithm is able to keep the potential good genes. When the number of iterations reaches 400, the algorithm is basically converged and the optimal individuals in the population can be searched. At this moment, the PLDA algorithm reaches a stable performance. Thus, the proposed load balancing strategy is effective.

5.6 Summary

Firstly, this chapter introduced and compared the current main parallel LDA methods, such as Mahout's parallelization of LDA, Yahoo's parallel topic model, the algorithm of parallel LDA based on Belief Propagation (BP) and Google's PLDA. Then, the PLDA algorithm and its implementation process in Hadoop were described in detail.

After studying the current main Hadoop task scheduling algorithm, this chapter brought genetic algorithms into the Hadoop platform, and proposed a static load balancing strategy based on a genetic algorithm for PLDA. The biggest feature of this strategy was that in a heterogeneous computing environment, the data was split according to the divisible load theory before the start of the task, which can minimize the synchronization waiting time among the nodes. Meanwhile, the population searching technology of genetic algorithms was used to make the population evolve to contain or approximate the optimal solution, improving the performance of the Hadoop cluster.

In the real homogeneous and heterogeneous Hadoop cluster environment, the experiment showed that when the PLDA algorithm dealt with a huge number of data in Hadoop, its efficiency was higher than the traditional serial algorithm. So, it is an effective way to process the massive data. In the Hadoop simulation environment, the experiment proved that the proposed static load balancing strategy was effective and stable, and it can enhance the computational efficiency of the Hadoop cluster.

Chapter 6 Conclusion and Future Works

6.1 Conclusion

In the information era, with the rapid development of computer technologies, web pages, emails, databases, digital libraries and other electronic texts have been increasing fast. Most of the information data is in an unstructured or semi-structured form. It is very difficult to discover the knowledge from it, but it is quite significant. How to find the useful and required information quickly and accurately from lots of information systems has become a challenge in the fields of information science and technology at present. The probability topic model such as LDA provides an effective tool for text analysis. It treats the documents as the probability distribution on topics so that the probability model is established for the text generation process and the probability model parameters are trained. Briefly, it can extract the latent topic information from text effectively.

In LDA, the core procedure is to estimate the posterior probability distribution of topics. The main approximation methods include the variational inference based on the EM algorithm and Markov Chain Monte Carlo method based on Gibbs sampling. The advantage of the former is that the variational parameters and iteration are used to approximate the Dirichlet parameters. Because its convergence speed is fast, the number of iterations is less. But, the calculation in each iteration is complex, especially the computing overhead is enormous when dealing with massive text. The advantage of the latter is to construct the Markov chain of the posterior probability. The global statistics is updated by sampling, and the number of iterations is fixed. The sampling algorithm in the iteration is simple, and the computing overhead is small. But, according to the nature of Markov chains, lots of iterations are required to achieve a stable state. These two algorithms have their own advantages and

disadvantages respectively. However, comparatively speaking, the scalability of LDA based on Gibbs sampling is better, and it is easier to realize the parallelization of LDA. Therefore, all the research work focused on the LDA based on Gibbs sampling in this thesis.

Text classification is one of the key technologies of text mining. It can deal with the problem of heterogeneous and messy text information to some extent, reducing the search space, speeding up the retrieval speed and improving the query accuracy. So, it is an effective way to classify the text information automatically, and it is a new challenge and a new development opportunity for the information industry. However, text classification technologies are facing some problems at present. For example, the feature dimension is too high, the number of text categories and the number of samples are too much, there is lots of noise and the number of samples in each class is unbalanced. Besides, the representation of text feature and the selection of text classification algorithm impact the entire text classification system greatly.

Aiming at some problems in the traditional text classification system, the probability topic models were studied and their advantages and disadvantages were compared in Chapter 2 in this thesis, especially the LDA topic model. LDA is a total probability generation model, so it has a clear internal structure. And, it can take advantage of effective probability inference algorithms to calculate. The size of its parameter space has nothing to do with the number of training documents so that LDA is more suitable for handling large-scale corpus. In addition, it is a hierarchical model, and it is more stable than other topic models. Besides, the SVM classification algorithm has a unique excellent performance for text classification.

In Chapter 3, the LDA model with good performance of text representation and the effect of dimension reduction and SVM algorithm with strong classification ability were combined, and a text classification method based on the LDA model was proposed. In the framework of SVM, the LDA model was used to model the text set.

Then, Gibbs sampling in Markov Chain Monte Carlo (MCMC) was used to infer and compute the model parameters indirectly, to obtain the probability distribution of text on the topic set. Finally, SVM was trained in the latent topic-document matrix, constructing a text classifier.

The classification experiment in 20newsgroups which is a real text set verified the validity and superiority of the proposed classification method based on the LDA model. The experimental result showed that it was effective to overcome the damaged problem of classification performance which was caused by the traditional feature selection method. And, it can improve the classification accuracy when reducing the data dimension. LDA as a method of feature selection, when its effect of dimension reduction was compared with SVM alone, feature vectors obtained a significant dimension reduction, saving the training time and enhancing the classification efficiency. In addition, compared with the combination of LSA and SVM, the proposed method had a better classification performance in the text set with many classes.

When LDA is used to model a whole text set, the number of topics T will influence the performance of LDA modeling the text set greatly. In other words, the given number of topics and the initial value of topic distribution play an important role in the parameters learning process of LDA. In Chapter 4, the current main selection methods of the optimal number of topics based on the LDA model were introduced, such as the selection method based on HDP and the standard method in Bayesian statistics. The former uses the nonparametric feature of DP to solve the selection problem of the optimal number of topics in LDA. But, it needs to build a HDP model and a LDA model respectively for the same data set. Obviously, it will take lots of time and computing when dealing with large-scale text sets or real text classification problems. The latter needs to specify a series of values of T manually, where T is the number of topics in LDA. After executing Gibbs sampling, the values of $\log p(w|T)$

can be obtained in sequence so that the value of T in LDA can be determined.

In Chapter 4, the relation between the optimal model and the topic similarity was proved theoretically. In the LDA model, the model was optimal when the average similarity among the topics was the smallest. Based on this constraint condition, the selection of the value of T and the parameter estimation of the LDA model were integrated in a framework. So, according to the basic idea of computing the sample density in DBSCAN, a selection algorithm of the optimal number of topics based on DBSCAN was proposed. By calculating the density of each topic, the most unstable topic model structure can be found. It was iterated until the model was stable. Finally, the optimal number of topics in LDA for the text set can be found. The experimental result showed that the proposed selection method of the optimal number of topics based on DBSCAN was effective. Compared with two current main methods, it can determine the value of T automatically with relatively little iterations. Especially, it did not need to set the number of topics manually in advance.

In the real applications of text classification, the training data inevitably contains noise. The noise samples would influence the final classification result significantly. At present, there are two main LDA-based noise processing methods, such as the data smoothing method based on the LDA model and the noise identification method based on the category entropy. Both of them can effectively remove some noise samples to some extent, but they cannot completely delete all the noise data from the data set. In addition, it is inevitable to remove some normal samples as noise mistakenly.

Therefore, the data smoothing method based on the LDA model and the noise identification method based on the category entropy were combined in Chapter 4. And then, a noise data reduction scheme was put forward. Its basic idea was to bring the concept of information entropy into text classification, and the noise samples were filtered according to the category entropy. Then, the generation process of the LDA

topic model was used to smooth the data, weakening the influence of the remaining noise samples after filtering noise. Meanwhile, it can keep the original size of the training set. The experiment with the real data set showed that when the noise ratio was large in the training data set, the classification accuracy of the proposed strategy was still good. In other words, the classification accuracy was affected by the noise ratio weakly, and the performance was stable.

Because the capacities of storage and computing are limited, the traditional single LDA cannot meet the needs of the real large-scale data set. So, the parallelization of LDA has become one of main methods and research priorities of processing the massive text data. In Chapter 5, the current main methods of paralleling LDA were introduced, such as Mahout's parallelization of LDA, Yahoo's parallel topic model, the algorithm of parallel LDA based on Belief Propagation (BP) and Google's PLDA. According to three main parallel programming technologies which are OpenMP, MPI and MapReduce, the following three inference methods of LDA model parameters realize their parallel computing, such as Variational Bayes (VB), Gibbs Sampling (GS) and Belief Propagation (BP). After comparison and analysis, the PLDA algorithm and Hadoop platform were chosen and studied in this thesis.

And then, three current main Hadoop task scheduling algorithms were analyzed. FIFO is simple, but its limitation is large. Fair scheduling has some improvement, but it cannot support large memory operations. Capacity scheduling is able to optimize the resource allocation effectively and enhance the operating efficiency. However, it has to configure lots of parameter values manually, which is quite difficult for the normal users who do not understand the details of the Hadoop cluster. In addition, all of them are suitable for the homogeneous environment. But in the heterogeneous environment, they will waste a lot of resources and the overheads will be large, affecting the overall performance of the Hadoop platform.

Therefore, this thesis brought a genetic algorithm which is an intelligent optimization

algorithm into the Hadoop platform [204] [205] [206]. And, considering that different nodes in the cluster have different computing speed, different communication capability and different storage capacity, a static load balancing strategy based on a genetic algorithm for PLDA was proposed. Its basic idea was to assume that the total time T of a mapper processing a data block contained four parts, i.e., the data copying time t_c, data processing time t_p, merging time of intermediate data t_m and buffer releasing time t_b. In a heterogeneous Hadoop cluster, T_{all} which is the total time of processing data blocks in one processing wave depends on the maximum consumed time of k participating mappers, namely $T_{all} = \max(T_1, T_2, \cdots, T_k)$.

According to the optimal principle of the divisible load theory, each data node should complete local Gibbs sampling at the same time in theory, namely $T_1 = T_2 = \cdots T_k$. Based on a genetic algorithm, the optimal principle was designed and accomplished in Chapter 5. It is mainly embodied in the determination of the fitness function in the genetic algorithm. The mean square error is used to measure the difference among the total time T of each mapper processing the data in the fitness function. As long as the value of the fitness is smaller than a given fitness threshold, which means that k mappers almost completes local data processing at the same time, the iteration of the genetic algorithm will be converged and end. Thus, an optimal or near optimal solution can be found to determine the size of one data block for each mapper, minimizing the completion time of the parallel LDA. In fact, to partition the data before the start of the task and to minimize the synchronization waiting time among each data node as far as possible can improve the performance of the Hadoop cluster in a heterogeneous environment.

Finally, the experiment was set up in a real Hadoop cluster environment, which had one name node and eight data nodes. In the homogeneous computing environment and the heterogeneous computing environment, the feasibility and validity of Hadoop

implementing the PLDA algorithm were proved respectively. Especially via the comparison of single machine and one data node in the cluster processing the data, the experiment proved that the PLDA algorithm is designed to deal with the large data set which single machine cannot handle. Due to the limited experimental condition, HSim, which is a simulator, was used to simulate a Hadoop cluster with 40 nodes to process 10GB to 100GB data. The validity of the proposed static load balancing strategy based on the genetic algorithm was verified in three aspects, which are different heterogeneities, different sizes of data and stability, where the heterogeneity of the Hadoop heterogeneous cluster was also determined and achieved by the mean square error. For example, for the same data but different heterogeneities, the PLDA with the proposed load balancing strategy is faster than the PLDA without the load balancing strategy obviously when the heterogeneity is bigger than 1.21. And, for the same heterogeneity but different sizes of data, the PLDA with the proposed load balancing strategy is still always faster than the PLDA without the load balancing strategy. In the simulated environment, if the value of the mean square error is smaller, the heterogeneity of the Hadoop cluster is smaller. On the contrary, if the value of the mean square error is bigger, the heterogeneity of the Hadoop cluster is bigger.

6.2 Future Works

The research work of this thesis mainly focused on the LDA probability topic model and the Gibbs sampling algorithm, which is one of posterior inference methods. In order to improve the accuracy, on the one hand, the LDA model and SVM were combined to construct a new text classifier. On the other hand, the optimal number of topics in LDA can be found automatically and the noise data can be removed from the text set effectively. In the aspect of enhancing the processing capacity and speed, the Hadoop platform was used to realize the parallelization of LDA, making the LDA model deal with the real large-scale text data effectively. This thesis has acquired the above research results, but there are still some problems and limitations. Therefore,

the following research work will need to be done in the future.

6.2.1 The Supervised LDA

More and more data with users label appears in the Internet, so to model the data with label has become people's actual needs [207]. The traditional LDA model is an unsupervised learning model. It only models words without considering other label information. The supervised LDA algorithm can provide an effective modeling tool for the data with label, and bring the text label information into the traditional LDA model [208]. It is able to calculate the probability distribution of latent topics in each class, overcoming the defect that the traditional LDA model has to allocate latent topics in a single class forcibly [209]. Besides, it can improve the representation capability of the LDA model [210]. Thus, to study and apply the supervised LDA will become one of future works. At present, the representative supervised LDA algorithms include author-topic model [211] [212] [213] and Labeled-LDA [214] [215].

In the author-topic model [211] [212], the relationship between the documents and the authors is modeled by the topics, which can apply to search the authors with the similar topics and other tasks. It brings the variables which can represent author information into the traditional LDA. Multiple labels, namely author information, are assigned to each document, and each of the authors set is from the author labels set. In this model, the authors in the authors set determine the document d. For each position in the document, an author is selected from the authors set randomly. And then, the selected author will choose the topic z. Next, according to the probability distribution of the topics on words, words are selected. Finally, Gibbs sampling is used to estimate the model parameters [213].

There are a large number of documents with users label in the Internet. Labeled-LDA

[214] can model the words in the documents with labels and the generation process of labels. According to the model parameters, the label distribution of new documents is estimated. It brings in the categories information, which has one more tier than the standard LDA model. The tier can be called text category tier. This algorithm is mainly used to determine the specific labels which are assigned to each word in the document with multi-label. And then, the distribution of each label in the whole document will be estimated. Because the label of each document is observable, the parameters of the label distribution have nothing to do with other parameters in the model [215]. Therefore, Gibbs sampling of the traditional LDA can be used to estimate the model parameters.

6.2.2 The Improved PLDA — PLDA+

The PLDA algorithm has two limitations: firstly, the word-topic matrix is not partitioned. So, the whole matrix has to be stored in a data node memory. Secondly, the overhead of data exchange in the Reduce phase is high when dealing with the large-scale data. In order to solve these limitations, [216] put forward an enhanced version of the PLDA algorithm, which was called PLDA+. It not only can partition the document-topic matrix, but also can split the word-topic matrix.

In brief, the nodes in the cluster are divided into two groups in the PLDA+ algorithm [216]. The task of a group of nodes is to maintain the split document-topic matrix and carry out Gibbs sampling. Besides, the topics are assigned to the words in the document. The task of another group of nodes is to maintain the split word-topic matrix, and accept and accumulate the update of the word-topic matrix from the previous group of nodes. Thus, the PLDA+ algorithm can halve the bottleneck of data exchange, reducing the communication time among the nodes and enhancing the speed and performance of the algorithm. In addition, assigning the above two kinds of tasks to two groups of nodes make it possible to deal with the problem of load

balancing more flexibly [217].

6.2.3 Dynamic Load Balancing Problem

Load balancing strategy is a task scheduling strategy [218] [219], and is a key issue of obtaining the high performance and improve the computing scalability and application scale in a distributed computing environment. The research work of this thesis focused on the static load balancing strategy in a Hadoop cluster. It assumed that the load can be determined and split before running. But in a real large-scale cluster such as cloud computing platform, if the load is measured only when running and is split dynamically, or the difference of the operational capability among the nodes or the load fluctuation is large, the Dynamic Load Balancing (DLB) strategy should be considered [220].

The research showed that compared with the static load balancing strategy, the DLB strategy usually can obtain 30% to 40% performance improvement [221]. Currently, according to the control position feature of the DLB strategy, the existing DLB strategies can be divided into three categories, such as the distributed DLB strategy, the centralized DLB strategy and the hybrid hierarchy DLB strategy [220].

The static load balancing strategy in this thesis was based on genetic algorithms. So, to make it change into DLB strategy for the dynamic load environment will be one of the future works. For example, the selection of crossover probability P_c and mutation probability P_m in genetic algorithms directly affects the convergence and the performance of the algorithm. Usually, the traditional P_c and P_m are static manual setting. Srinivas et al proposed a method of dynamic parameters setting, where P_c and P_m can change automatically along with the fitness to adapt the dynamic computing environment and decrease the limitation of the manual selection

parameters [222]. If P_c and P_m increase, the change of the individuals will increase, but the values of the fitness of the individuals will decline. It is beneficial to guarantee the global solutions, but not conducive to make the population converge to the optimal solution. If P_c and P_m decrease, the values of the fitness of the individuals will increase, but the change of the individuals will decline. It can help the population converge to the optimal solution, but cannot ensure the global optimal solution [223] [224]. Therefore, it is necessary to set P_c and P_m dynamically.

6.2.4 The Application of Cloud Computing Platform

Due to the limited experimental condition, the number of nodes in the cluster is not large, and the computing and storage capacity is not strong. Thus, a simulator HSim was used to implement the proposed static load balancing strategy in this thesis. In the simulation environment, it has good performance and stability, but there will be a challenge when processing the real massive data. At present, there are many large cloud computing platforms, such as Google's Google App Engine, Amazon's Amazon EC2 (Amazon Elastic Compute Cloud) and Amazon S3 (Amazon Simple Storage Service), IBM's Blue Cloud and Microsoft's Windows Azure [225] [226] [227]. Therefore, applying the proposed load balancing strategy to the above large cloud computing platforms and obtaining good performance will become one of the future works.

6.2.5 The Application of Interdisciplinary Field

LDA probability topic model not only can be applied to the field of text analysis, but also can be applied to images, audio, video and other structured information analysis. For example, it can be applied to image annotation and image classification. The key is to define the words and the documents reasonably. Usually, the pixels block in the

image is treated as the word, the single image is treated as the document, and all the image libraries are regard as the document set [228] [229]. At the same time, the features of the pixels block are used to carry out the coding, generating the word list of the image library. Besides, the LDA model can be used to learn the image library [230]. Compared to text, the image itself has some characteristics, such as the high dimensions in image elements, the strong relevance and the low dispersion. Therefore, the establishment of topic models and the selection of model parameters are worthy of the further study, which will also become the emphasis of the future research.

From the research and analysis of the LDA model and its parallel algorithm in this thesis, it can be seen that LDA is a fast and effective text analysis tool. In addition, the derivative algorithm based on LDA topic model is an important issue in the field of machine learning. At present, when the LDA model and its related algorithms deal with the real massive text data, lots of work will be required further, including the study of theoretical knowledge and the analysis of application scenarios. On the whole, it is a field of scientific research with practicability, which is worthy of intensive study.

References

References

[1] Liu, B. (2011) *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data.* 2^{nd} edn. New York: Springer Publishing.

[2] Han, J.W., Kamber, M. and Pei, J. (2006) *Data Mining: Concepts and Techniques.* 2^{nd} edn. Burlington: Morgan Kaufmann.

[3] Tan, P.N., Steinbach, M. and Kumar, V. (2005) *Introduction to Data Mining.* Boston: Addison-Wesley.

[4] Kantardzic, M. (2011) *Data Mining: Concepts, Models, Methods, and Algorithms.* 2^{nd} edn. Hoboken: Wiley-IEEE Press.

[5] Feldman, R. and Sanqer, J. (2006) *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data.* Cambridge: Cambridge University Press.

[6] Berry, M.B. and Koqan, J. (2010) *Text Mining: Applications and Theory.* 1^{st} edn. Colorado: Wiley.

[7] Song, M. and Wu, Y.F.B. (2008) *Handbook of Research on Text and Web Mining Technologies.* Hershey: Information Science Reference.

[8] Francis, L. (2006) Taming Text: An Introduction to Text Mining. *Casualty Actuarial Society Forum*, pp. 51-88.

[9] Karanikas, H. and Theodoulidis, B. (n.d.) Knowledge Discovery in Text and Text Mining Software. *Centre for Research in Information Management Department of Computation, UMIST, Manchester, UK* [Online]. Available at: http://www.crim.co.umist.ac.uk [Accessed 26 October 2011].

[10] Lent, B., Agrawal, R. and Srikant, R. (1997) Discovering Trends in Text Databases. *IBM Almaden Research Center San Jose, California, USA* [Online]. Available at: http://www.almaden.ibm.com/cs/projects/iis/hdb/Publications/papers/kdd97_trends.pdf [Accessed 26 October 2011].

[11] Miner, G., Elder, J., Hill, T., Nisbet, R., Delen, D. and Fast, A. (2012) *Practical Text Mining and Statistical Analysis for Non-structured Text Data Applications.* Waltham: Academic Press.

[12] Zanasi, A. (2007) *Text Mining and its Applications to Intelligence, CRM and Knowledge Management.* Ashurst: WIT Press.

[13] Srivastava, A. and Sahami, M (2009) *Text Mining: Classification, Clustering, and Applications.* 1st edn. USA: Chapman and Hall/CRC.

[14] Berry, M.W. (2003) *Survey of Text Mining: Clustering, Classification, and Retrieval.* 2nd edn. New York: Springer Publishing.

[15] Franke, J., Nakhaeizadeh, G. and Renz, I. (2003) *Text Mining: Theoretical Aspects and Applications.* 1st edn. Berlin: Physica-Verlag HD.

[16] Chemudugunta, C. (2010) *Text Mining with Probabilistic Topic Models: Applications in Information Retrieval and Concept Modeling.* Saarbrücken: LAP LAMBERT Academic Publishing.

[17] Blei, D.M. (2004) *Probabilistic models of text and images.* California: UNIVERSITY OF CALIFORNIA.

[18] Blei, D.M. (2012) Probabilistic topic models. *Communications of the ACM,* 55 (4), pp. 77-84.

[19] Blei, D.M. and Lafferty, J.D. (2009) Topic models. *Text Mining: Theory and Applications,* pp. 71-93.

[20] Deerwester, S., Dumais, S.T., Furnas, G.W. and Landauer, T.K. (1990) Indexing by Latent semantic *Analysis. Journal of the American Society for Information Science, 41 (6),* pp. 391-407.

[21] Torkkola, K. (2002) Discriminative features for document classification. *16th International Conference on Pattern Recognition,* 1, pp. 472-475.

[22] Foltz, P.W. (1990) Using latent semantic indexing for information filtering. *COCS '90 Proceedings of the ACM SIGOIS and IEEE CS TC-OA conference on Office information systems,* pp. 40-47.

[23] McCarey, F., Cinnéide, M. and Kushmerick, N. (2006) Recommending library methods: an evaluation of the vector space model (VSM) and latent semantic indexing (LSI). *ICSR'06 Proceedings of the 9th international conference on Reuse of Off-the-Shelf Components, pp. 217-230.*

[24] Chen, L., Tokuda, N. and Nagai, A. (2003) *A new differential LSI space-based probabilistic document classifier.* Information Processing Letters, 88 (5), pp. 203-212.

[25] Hofmann, T. (1999) Probabilistic latent semantic indexing. *SIGIR '99 Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 50-57.

[26] Hofmann, T. (1999) Probabilistic latent semantic analysis. *UAI'99 Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pp. 289-296.

[27] Brants, T. (2005) Test Data Likelihood for PLSA Models. *Information Retrieva*, 8 (2), pp. 181-196.

[28] Buntine, W. (2009) Estimating Likelihoods for Topic Models. *ACML '09 Proceedings of the 1st Asian Conference on Machine Learning: Advances in Machine Learning*, pp. 51-64.

[29] Lu, Y., Mei, Q.Z. and Zhai, C.X. (2011) Investigating task performance of probabilistic topic models: an empirical study of PLSA and LDA. *Information Retrieval*, 14 (2), pp. 178-203.

[30] Blei, D.M., Ng, A.Y. and Jordan, M.I. (2003) Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3, pp. 993-1022.

[31] Minka, T. and Lafferty, J. (2002) Expectation-propagation for the generative aspect model. *UAI'02 Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pp. 352-359.

[32] Blei, D.M. and Jordan, M.I. (2006) Variational inference for Dirichlet process mixtures. *Bayesian Analysis*, 1 (1), pp. 121-144.

[33] Kao, A. and Poteet, S.R. (2010) *Natural Language Processing and Text Mining.* 1^{st} edn. New York: Springer Publishing.

[34] Bellegarda, J. (2008) *Latent Semantic Mapping: Principles and Applications.* San Rafael: Morgan and Claypool Publishers.

[35] Zhong, N. and Liu, J.M. (2004) *Intelligent Technologies for Information Analysis.* 2^{nd} edn. New York: Springer Publishing.

[36] Vinokourov, A. and Girolami, M. (2002) A probabilistic framework for the hierarchic organisation and classification of document collections. *Journal of Intelligent Information Systems*, 18(2/3), pp. 153–172.

[37]Lucas, P., Gámez, J.A. and Cerdan, A.S. (2007) *Advances in Probabilistic Graphical Models.* 1^{st} edn. New York: Springer Publishing.

[38] Chemudugunta, C., Smyth, P. and Steyvers, M. (2007) Modeling general and specific aspects of documents with a probabilistic topic model. *In Proceedings of the Conference on Advances in Neural Information Processing Systems (NIPS'07),* pp. 241–248.

[39] Girolami, M. and Kabán, A. (2003) On an equivalence between PLSI and LDA. *SIGIR '03 Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval,* pp. 433-434.

[40] Masada, T., Kiyasu, S. and Miyahara, S. (2008) Comparing LDA with pLSI as a dimensionality reduction method in document clustering. *LKR'08 Proceedings of the 3rd international conference on Large-scale knowledge resources: construction and application,* pp. 13-26.

[41] Dongarra, J., Foster, I., Fox, G.C., Gropp, W., Kennedy, K., Torczon, L. and White, A. (2002) *The Sourcebook of Parallel Computing.* 1^{st} edn. Burlington: Morgan Kaufmann.

[42] Grama, A., Karypis, G., Kumar, V. and Gupta, A. (2003) *Introduction to Parallel Computing.* 2^{nd} edn. Boston: Addison-Wesley.

[43] Hwang, K., Dongarra, J. and Fox, G.C. (2011) *Distributed and Cloud Computing: From Parallel Processing to the Internet of Things.* Burlington: Morgan Kaufmann.

[44] Amati, G., D'Aloisi, D., Giannini, V. and Ubaldini, F. (1997) A framework for filtering news and managing distributed data. Journal of Universal Computer Science, 3 (8), pp. 1007–1021.

[45] Pacheco, P. (2011) *An Introduction to Parallel Programming.* Burlington: Morgan Kaufmann.

[46] Gebali, F. (2011) *Algorithms and Parallel Computing.* Hoboken: Wiley-Blackwell.

[47] Fountain, T.J. (2006) *Parallel Computing: Principles and Practice.* Cambridge: Cambridge University Press.

[48] Bischof, C., Bucker, M., Gibbon, P., Joubert, G. and Lippert, T. (2008) *Parallel Computing: Architectures, Algorithms and Applications.* Amsterdam: IOS Press.

[49] Chandra, R., Menon, R. Daqum, L., Kohr, D., Maydan, D. and McDonald, J. (2000) *Parallel Programming in OpenMP.* Burlington: Morgan Kaufmann.

[50] Tora, S. and Eguchi, K. (2013) MPI/OpenMP hybrid parallel inference for Latent

Dirichlet Allocation. *IEICE TRANSACTIONS on Information and Systems,* 96 (5), pp. 1006-1015.

[51] Dean, J. and Ghemawat, S. (2004) Mapreduce: Simplified data processing on large clusters. *In Proceedings of the ACM USENIX Symposium on Operating Systems Design and Implentation (OSDI'04),* pp. 137–150.

[52] Miner, D. and Shook, A. (2012) *MapReduce Design Patterns: Building Effective Algorithms and Analytics for Hadoop and Other Systems.* 1st edn. Sebastopol: O'Reilly Media.

[53] Janert, P.K. (2010) *Data Analysis with Open Source Tools.* 1st edn. Sebastopol: O'Reilly Media.

[54] Dumbil, E. (2012) *Planning for Big Data.* 1st edn. Sebastopol: O'Reilly Media.

[55] Wang, L.Z. et al., (2008) Scientific Cloud Computing: Early Definition and Experience. *10th IEEE International Conference on High Performance Computing and Communications (HPCC),* pp. 825-830.

[56] Armbrust, M. et al., (2010) A view of cloud computing. *Communications of the ACM,* 53 (4), pp. 50-58.

[57] Zhang, L.J. and Zhou, Q. (2009) CCOA: Cloud Computing Open Architecture. *IEEE International Conference on Web Services (ICWS),* pp. 607-616.

[58] Zhang, S., Zhang, S.Z., Chen, X.B. and Huo, X.Z. (2010) Cloud Computing Research and Development Trend. *Second International Conference on Future Networks (ICFN),* pp. 93-97.

[59] Gong, C.Y. et al., (2010) The Characteristics of Cloud Computing. *2010 39th International Conference on Parallel Processing Workshops (ICPPW),* pp. 275-279.

[60] White, T. (2010) *Hadoop: The Definitive Guide.* 2nd edn. Sunnyvale: Yahoo Press.
15

[61]Perera, S. and Gunarathne, T. (2013) *Hadoop MapReduce Cookbook.* Birmingham: PACKT PUBLISHING.

[62] Lam, C. (2010) *Hadoop in Action.* 1st edn. Greenwich: Manning Publications.

[63] Kala Karun, A. and Chitharanjan, K. (2013) A review on hadoop — HDFS infrastructure extensions. *2013 IEEE Conference on Information & Communication*

Technologies (ICT), pp. 132-137.

[64] Taylor, R.C. (2010) An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics. Proceedings of the 11th Annual Bioinformatics Open Source Conference (BOSC), 11 (12), pp. 1-6.

[65] Lin, X.L., Meng, Z.D., Xu, C. and Wang, M. (2012) A Practical Performance Model for Hadoop MapReduce. *2012 IEEE International Conference on Cluster Computing Workshops*, pp. 231-239.

[66] Chu, C.T. et al., (2006) Map-Reduce for Machine Learning on Multicore. *Advances in NIPS*, 19, pp. 281-288.

[67] Brown, R.A. (2009) Hadoop at home: large-scale computing at a small college. *Proceedings of the 40th ACM technical symposium on Computer science education*, pp. 106-110.

[68] Duan, S.Q., Wu, B., Wang, B. and Yang, J. (2011) Design and implementation of parallel statiatical algorithm based on Hadoop's MapReduce model. *2011 IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS)*, pp. 134-138.

[69] Gomes, R., Welling, M. and Perona, P. (2008) Memory bounded inference in topic models. *In Proceedings of the International Conference on Machine Learning (ICML'08),* pp. 344–351.

[70] Xie, J. et al., (2010) Improving MapReduce performance through data placement in heterogeneous Hadoop clusters. *2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW)*, pp. 1-9.

[71] Porteous, I. et al., (2008) Fast collapsed gibbs sampling for latent dirichlet allocation. *In Proceedings of the International SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'08),* pp. 569–577.

[72] Newman, D., Asuncion, A., Smyth, P. and Welling, M. (2009) Distributed Algorithms for Topic Models. *The Journal of Machine Learning Research*, 10, pp. 1801-1828.

[73] Tang, N. (2011) *Information Extraction from the Internet.* 1^{st} edn. CreateSpace Independent Publishing Platform.

[74] Indurkhya, N. and Damerau, F.J. (2010) *Handbook of Natural Language Processing.* 2^{nd} edn. USA: Chapman and Hall/CRC.

[75] Yang, Y., Slattery, S. and Ghani, R. (2002) A study of approaches to hypertext categorization. *Journal of Intelligent Information Systems,* 18(2/3), pp. 219–241.

[76] Chen, W.L. et al., (2004) Automatic word clustering for text categorization using global information. *AIRS'04 Proceedings of the 2004 international conference on Asian Information Retrieval Technology,* pp. 1-11.

[77] Guyon, I. and Elisseeff, A. (2003) An introduction to variable and feature selection. *The Journal of Machine Learning Research,* 3, pp. 1157-1182.

[78] Yang, Y. and Pedersen, J.O. (1997) A Comparative Study on Feature Selection in Text Categorization. *Proc. 14th Int'l Conf. Machine Learning,* pp. 412-420.

[79] Forman, G. (2003) An extensive empirical study of feature selection metrics for text classification. *The Journal of Machine Learning Research,* 3, pp. 1289-1305.

[80] Tenenbaum, J.B., Silva, V. and Langford, J.C. (2000) A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science,* 290, pp. 2319-2323.

[81] Yuan, H.Y., Gao, X.J. and Lei, F. (2010) The Feature Extraction and Dimension Reduction Research Based on Weighted FCM Clustering Algorithm. *2010 International Conference on Electronics and Information Engineering (ICEIE),* 2, pp-114-117.

[82] Griffiths, T.L. and Steyvers, M. (2004) Finding scientific topics. *The National Academy of Sciences of the USA,* 101 (1), pp. 5228-5235.

[83] Alon, N. and Spencer, J.H. (2008) *The Probabilistic Method.* 3^{rd} edn. Hoboken: Wiley-Blackwell.

[84] Schutze, H., Hull, D. and Pedersen, J. (1995). A comparison of classifiers and document representations for the routing problem. *In Proceedings of SIGIR-95, 18th International Conference on Research and Developement in Information Retrieval,* pp. 229–237.

[85] Lafferty, J.D. and Blei, D.M. (2005) Correlated topic models. *Advances in neural information processing systems,* pp. 147-154.

[86] Teh, Y.W., Jordan, M.I., Beal, M.J. and Blei, D.M. (2007) Hierarchical Dirichlet Processes. *Journal of the American Statistical Association,* 101 (476), pp. 1566-1581.

[87] Zhou, J.Y., Wang, F.Y. and Zeng, D.J. (2011) Hierarchical Dirichlet Processes and Their Applications: A Survey. *Acta Automatica Sinica,* 37 (4), pp. 389-407.

[88] Teh, Y.W., Jordan, M.I., Beal, M.J. and Blei, D.M. (2005) Sharing Clusters among Related Groups: Hierarchical Dirichlet Processes. *Advances in Neural Information Processing Systems*, pp. 1385-1392.

[89] Zhu, X.Q., Wu, X.D. and Chen, Q.J. (2003) Eliminating class noise in large datasets. *In Proceedings of the Twentieth International Conference on Machine Learning*, pp. 920-927.

[90] Gamberger, D., Lavrac, N. and Dzeroski, S. (2000) Noise detection and elimination in data preprocessing: Experiments in medical domains. *Applied Artificial Intelligence: An International Journal*, 14 (2), pp. 205-223.

[91] Ramaswamy, S., Rastogi, R. and Shim, K. (2000) Efficient algorithms for mining outliers from large data sets. *SIGMOD '00 Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 29 (2), pp. 427-438.

[92] Liu, D.X., Xu, W.R. and Hu, J.N. (2009) A feature-enhanced smoothing method for LDA model applied to text classification. *International Conference on Natural Language Processing and Knowledge Engineering*, pp. 1-7.

[93] Zhou, D. and Wade, V. (2009) Smoothing methods and cross-language document re-ranking. *CLEF'09 Proceedings of the 10th cross-language evaluation forum conference on Multilingual information access evaluation: text retrieval experiments*, pp. 62-69.

[94] Garcia, M., Hidalgo, H. and Chavez, E. (2006) Contextual Entropy and Text Categorization. *Fourth Latin American Web Congress*, pp. 147-153.

[95] Xu, L. (2003) Data smoothing regularization, multi-sets-learning, and problem solving strategies. *Neural Networks - 2003 Special issue: Advances in neural networks research*, 16 (5-6), pp. 817-825.

[96] Jia, R., Tao, T. and Ying, Y. (2012) Bayesian Networks Parameter Learning Based on Noise Data Smoothing in Missing Information. *ISCID '12 Proceedings of the 2012 Fifth International Symposium on Computational Intelligence and Design*, 1, pp. 136-139.

[97] Zhang, X.W., Parisi-Presicce, F., Sandhu, R. and Park, J. (2005) Formal model and policy specification of usage control. *ACM Transactions on Information and System Security (TISSEC)*, 8 (4), pp. 351-387.

[98] Arora, S. (2012) Learning Topic Models - Going beyond SVD. *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 1-10.

[99] Sebastiani, F. (2002) Machine learning in automated text categorization. *ACMComputing Surveys,* 34(1), pp. 1-47.

[100] Bishop, C.M. (2006) *Pattern recognition and machine learning.* New York: Springer Publishing.

[101] Ha-Thuc, V. and Srinivasan, P. (2008) Topic models and a revisit of text-related applications. *PIKM '08 Proceedings of the 2nd PhD workshop on Information and knowledge management,* pp. 25-32.

[102] Lertnattee, V. and Theeramunkong, T. (2004) Effect of term distributions on centroid-based text categorization. *Informatics and computer science intelligent systems,* 158 (1), pp. 89-115.

[103] Debole, F. and Sebastiani, F. (2003) Supervised term weighting for automated text categorization. *SAC '03 Proceedings of the 2003 ACM symposium on Applied computing,* pp. 784-788.

[104] Xue, D.J. and Sun, M.S. (2003) Chinese text categorization based on the binary weighting model with non-binary smoothing. *ECIR'03 Proceedings of the 25th European conference on IR research,* pp. 408-419.

[105] Heath, M.T. (1997) *Scientific computing an introductory survey.* New York: McGraw-Hill.

[106] Zeng, J., Cheung, W.K. and Liu, J.M. (2013) Learning Topic Models by Belief Propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 35 (5), pp. 1121-1134.

[107] Wang, C., Thiesson, B., Meek, C. and Blei, D.M. (2009) Markov topic models. *The Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS),* pp. 583-590.

[108] Casella, G. and George, E.I. (1992) Explaining the Gibbs sampler. *The American Statistician,* 46 (3), pp. 167-174.

[109] Bigi, B. (2003) Using Kullback-Leibler distance for text categorization. *ECIR'03 Proceedings of the 25th European conference on IR research,* pp. 305-319.

[110] Kullback, S. and Leibler, R.A. (1951) On Information and Sufficiency. *The annals of Mathematical Statistics,* 22 (1), pp. 79-86.

[111] Heinrich, G. (2009) Parameter Estimation for Text Analysis. *Technical Report No. 09RP008-FIGD, Fraunhofer Institute for Computer Graphics,* pp. 1-32.

[112] Bertsekas, D. P. (1999) *Nonlinear Programming.* Cambridge: Athena Scientific.

[113] Bonnans, J. F. (2006) *Numerical optimization: theoretical and practical aspects.* New York: Springer Publishing.

[114] Kschischang, F.R., Frey, B.J. and Loeliger, H.A. (2001) Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory,* 47 (2), pp. 498-519.

[115] Neal, R.M. (1993) *Probabilistic inference using Markov chain Monte Carlo methods.* Toronto: University of Toronto.

[116] Seneta, E. (2006) *Non-negative Matrices and Markov Chains.* 2^{nd} edn. New York: Springer Publishing.

[117] Newman, D., Asuncion, A., Smyth, P. and Welling, M. (2007) Distributed inference for latent dirichlet allocation. *In Proceedings of the Conference on Advances in Neural Information Processing Systems (NIPS'07),* pp. 1081–1088.

[118] Mitchell, M. (1998) *An Introduction to Genetic Algorithms.* Cambridge: MIT Press.

[119] Goldberg, D.E. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning.* 1^{st} edn. Boston: Addison Wesley.

[120] Affenzeller, M., Winkler, S., Waqner, S. and Beham, A. (2009) *Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications.* 1^{st} edn. USA: Chapman & Hall/CRC.

[121] Sivanandam, S.N. and Deepa, S.N. (2007) *Introduction to Genetic Algorithms.* 2^{nd} edn. New York: Springer Publishing.

[122] Srinivas, M. and Patnaik, L.M. (1994) Genetic algorithms: a survey. *Computer,* 27 (6), pp. 17-26.

[123] Potts, J.C., Giddens, T.D. and Yadav, S.B. (1994) The development and evaluation of an improved genetic algorithm based on migration and artificial selection. *IEEE Transactions on Systems, Man and Cybernetics,* 24 (1), pp. 73-86.

[124] Burjorjee, K.M. (2013) Explaining optimization in genetic algorithms with uniform crossover. *FOGA XII '13 Proceedings of the twelfth workshop on Foundations of genetic algorithms XII,* pp. 37-50.

[125] Lobo, F.G. and Lima, C.F. (2005) A review of adaptive population sizing

schemes in genetic algorithms. *GECCO '05 Proceedings of the 2005 workshops on Genetic and evolutionary computation,* pp. 228-234.

[126] Miqdalas, A., Pardalos, P.M. and Storøy, S. (2012) *Parallel Computing in Optimization.* Berlin: Kluwer Academic Publishers.

[127] Shokripour, A. and Othman, M. (2009) Survey on Divisible Load Theory and its Applications. *International Conference on Information Management and Engineering (ICIME) 2009,* pp. 300-304.

[128] Bharadwaj, V., Ghose, D. and Robertazzi, T. (2003) Divisible Load Theory: A New Paradigm for Load Scheduling in Distributed Systems. *Cluster Computing,* 6 (1), pp. 7-17.

[129] Robertazzi, T.G. (2003) Ten reasons to use divisible load theory. *Computer,* 36 (5), pp. 63-68.

[130] Li, X., Veeravalli, B. and Chi, C.K. (2001) Divisible load scheduling on a hypercube cluster with finite-size buffers and granularity constraints. *First IEEE/ACM International Symposium on Cluster Computing and the Grid,* pp. 660-667.

[131] Heien, E.M., Fujimoto, N. and Hagihara, K. (2008) Static Load Distribution for Communication Intensive Parallel Computing in Multiclusters. *16th Euromicro Conference on Parallel, Distributed and Network-Based Processing,* pp. 321-328.

[132] Noel, P.A. and Ganesan, S. (2010) Performance analysis of Divisible Load Scheduling utilizing Multi-Installment Load Distribution with varying sizes of result load fractions. *2010 IEEE International Conference on Electro/Information Technology (EIT),* pp. 1-6.

[133] Ghatpande, A., Nakazato, H., Watanabe, H. and Beaumont, O. (2008) Divisible Load Scheduling with Result Collection on Heterogeneous Systems. *IEEE International Symposium on Parallel and Distributed Processing (IPDPS) 2008,* pp. 1-8.

[134] Sohn, J. and Robertazzi, T.G. (1998) Optimal time-varying load sharing for divisible loads. *IEEE Transactions on Aerospace and Electronic Systems,* 34 (3), 907-923.

[135] Lin, X., Lu, Y., Deogun, J. and Goddard, S. (2007) Real-Time Divisible Load Scheduling for Cluster Computing. *13th IEEE Real Time and Embedded Technology and Applications Symposium (RTAS),* pp. 303-314.

[136] Yang, Y., Zhang, J. and Kisiel, B. (2003) A scalability analysis of classifiers in

text categorization. *Proceedings of SIGIR-03, 26th ACM International Conference on Research and Development in Information Retrieval,* pp. 96–103.

[137] Grover, C. et al., (2004) A Framework for Text Mining Services. *School of Informatics, University of Edinburgh and National e-Science Centre, Edinburgh* [Online]. Available at: http://www.ltg.ed.ac.uk/np/publications/ltg/papers/Grover2004Framework.pdf [Accessed 26 October 2011].

[138] Salton, G., Wong, A. and Yang, C.S. (1997) A vector space model for automatic indexing. *Readings in information retrieval,* pp. 273-280.

[139] Sheikholeslami, G., Chatterjee, S. and Zhang, A. (1998) Wavecluster: A multi-resolution clustering approach for very large spatial databases. *Proceedings of the 24th VLDB Conference,* pp. 428-439.

[140] Sun, C.K., Gao, B., Cao, Z.F. and Li, H. (2008) HTM: a topic model for hypertexts. *EMNLP '08 Proceedings of the Conference on Empirical Methods in Natural Language Processing,* pp. 514-522.

[141] Joachims, T. (1999) Making large-scale support vector machine learning practical. *In Advances in kernel methods: support vector learning,* pp. 169-184.

[142] Tong, S. and Koller, D. (2002) Support vector machine active learning with applications to text classification. *Machine Learning Research,* 2 (3/1), pp. 45-66.

[143] Liu, T.Y. (2005) Support vector machines classification with a very large-scale taxonomy. *ACM SIGKDD Explorations Newsletter - Natural language processing and text mining,* 7 (1), pp. 36-43.

[144] Joachims, T. (1998) Text categorization with support vector machines: learning with many relevant features. *Proceedings of ECML-98, 10th European Conference on Machine Learning,* pp. 137–142.

[145] Joachims, T. (2001) A statistical learning learning model of text classification for support vector machines. *SIGIR '01 Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval,* pp. 128-136.

[146] Tan, S.B. et al., (2005) A novel refinement approach for text categorization. *CIKM '05 Proceedings of the 14th ACM international conference on Information and knowledge management,* pp. 469-476.

[147] Xue, X. and Zhou, Z. (2009) Distributional Features for Text Categorization.

IEEE Transactions on Knowledge and Data Engineering, 21 (3), pp. 428-442.

[148] Goncalves, T. and Quaresma, P. (2006) Evaluating preprocessing techniques in a Text Classification problem. pp. 841-850, [Online]. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.131.1271&rep=rep1&type=pdf [Accessed 26 October 2011].

[149] Srividhya, V. and Anitha, R. (2010) Evaluating Preprocessing Techniques in Text Categorization. *International Journal of Computer Science and Application Issue 2010,* pp. 49-51.

[150] Schiefele, U. (1996) Topic Interest, Text Representation, and Quality of Experience. *Contemporary Educational Psychology*, 21 (1), pp. 3-18.

[151] Frank, S.L., Koppen, M., Noordman, L.G.M. and Vonk, W. (2007) Modeling Multiple Levels of Text Representation. *Higher level language processes in the brain: inference and comprehension processes,* pp. 133-157.

[152] Dumais, S.T., Platt, J., Heckerman, D. and Sahami, M. (1998) Inductive learning algorithms and representations for text categorization. *Proceedings of CIKM-98, 7th ACM International Conference on Information and Knowledge Management*, pp. 148–155.

[153] Yan, J. et al., (2006) Effective and Efficient Dimensionality Reduction for Large-Scale and Streaming Data Preprocessing. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING,* 18 (2), pp. 1-14.

[154] Conrad, J.G. and Utt, M.H. (1994) A system for discovering relationships by feature extraction from text databases. SIGIR '94 Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 260-270.

[155] Mooney, R.J. and Nahm, U.Y. (2003) Text Mining with Information Extraction. *Proceedings of the 4th International MIDP Colloquium,* pp.141-160.

[156] Drucker, H., Vapnik, V. and Wu, D. (1999) Support vector machines for spam categorization. *IEEE Transactions on Neural Networks,* 10(5), pp. 1048–1054.

[157] Lewis, D.D. (1995) Evaluating and optimizing autonomous text classification systems. *SIGIR '95 Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval,* pp. 246-254.

[158] Halkidi, M., Vazirgiannis, M. and Batistakis, Y. (2000) Quality Scheme Assessment in the Clustering Process. *PKDD '00 Proceedings of the 4th European*

Conference on Principles of Data Mining and Knowledge Discovery, pp. 265-276.

[159] Andrieu, C., Freitas, N.D., Doucet, A. and Jordan, M. (2003) An Introduction to MCMC for Machine Learning. *Machine Learning In Machine Learning*, 50 (1-2), pp. 5-43.

[160] Chang, C.C. and Lin, C.J. (2011) LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2 (3), pp. 1-39.

[161] LIBSVM -- A Library for Support Vector Machines (2011) Available at: http://www.csie.ntu.edu.tw/~cjlin/libsvm/ [Accessed: 27 May 2011].

[162] Apte, C., Damerau, F. and Weiss, S. (1994). Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 12 (3):233–251.

[163] Jo, T. and Jo, G.S. (2008) Table Based Single Pass Algorithm for Clustering Electronic Documents in 20NewsGroups. *IEEE International Workshop on Semantic Computing and Applications*, pp. 66-71.

[164] MIT Computer Science and Artificial Intelligence Laboratory: Twenty news groups dataset (2011) Available at: http://qwone.com/~jason/20Newsgroups/ [Accessed: 03 April 2011].

[165] McCallum, A., Mimno, D.M. and Wallach, H.M. (2009) Rethinking LDA: Why Priors Matter. *Advances in Neural Information Processing Systems*, pp. 1973-1981.

[166] Hsu, C.W. and Lin, C.J. (2002) A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13 (2), pp. 415-425.

[167] Li, W. and McCallum, A. (2006) Pachinko allocation: DAG-structured mixture models of topic correlations. *ICML '06 Proceedings of the 23rd international conference on Machine learning*, pp. 577-584.

[168] Yakhnenko, O. and Honavar, V. (2009) Multi-modal hierarchical Dirichlet process model for predicting image annotation and image-object label correspondence. *Proceedings of the SIAM International Conference on Data Mining*, pp. 281-294.

[169] Karypis, G., Han, E.H. and Kumar, V. (1999) Chameleon: hierarchical clustering using dynamic modeling. *Computer*, 32 (8), pp. 68-75.

[170] Qian, W.N., Gong, X.Q. and Zhou, A.Y. (2003) Clustering in very large databases based on distance and density. *Journal of Computer Science and Technology*,

18 (1), pp. 67-76.

[171] Zhou, S.G. et al., (2000) Combining Sampling Technique with DBSCAN Algorithm for Clustering Large Spatial Databases. *PADKK '00 Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Current Issues and New Applications,* pp. 169-172.

[172] Ester, M., Kriegel, H.P., Sander, J. and Xu, X.W. (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. *In Second International Conference on Knowledge Discovery and Data Mining,* pp. 226-231.

[173] Ashour, W. and Sunoallah, S. (2011) Multi density DBSCAN. *IDEAL'11 Proceedings of the 12th international conference on Intelligent data engineering and automated learning,* pp. 446-453.

[174] Borah, B. and Bhattacharyya, D.K. (2004) An improved sampling-based DBSCAN for large spatial databases. *Proceedings of International Conference on Intelligent Sensing and Information Processing,* pp. 92-96.

[175] Duan, L. et al., (2007) A local-density based spatial clustering algorithm with noise. *Information Systems,* 32 (7), pp. 978-986.

[176] Van Kley, E.R. (1994) Data smoothing algorithm. *Position Location and Navigation Symposium,* pp. 323-328.

[177] Siebes, A. and Kersten, R. (2012) Smoothing categorical data. *ECML PKDD'12 Proceedings of the 2012 European conference on Machine Learning and Knowledge Discovery in Databases,* pp. 42-57.

[178] Shannon, C.E. (2001) A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review,* 5 (1), pp. 3-55.

[179] Li, W.B., Sun, L., Feng, Y.Y. and Zhang, D.K. (2008) Smoothing LDA model for text categorization. *AIRS'08 Proceedings of the 4th Asia information retrieval conference on Information retrieval technology,* pp. 83-94.

[180] Nigam, K., Lafferty, J. and McCallum, A. (1999) Using Maximum Entropy for Text Classification. *IJCAI-99 Workshop on Machine Learning for Information Filtering,* pp. 61-67.

[181] Fu, L. and Hou, Y.X. (2009) Using non-extensive entropy for text classification. *ICIC'09 Proceedings of the 5th international conference on Emerging intelligent computing technology and applications,* pp. 908-917.

[182] Halkidi, M., Batistakis, Y. and Vazirgiannis, M. (2001) On Clustering Validation Techniques. *Journal of Intelligent Information Systems,* 17 (2-3), pp. 107-145.

[183] Huang, H., Bi, L.P., Song, H.T. and Lu, Y.C. (2005) A variational EM algorithm for large databases. *Proceedings of 2005 International Conference on Machine Learning and Cybernetics,* 5, pp. 3048-3052.

[184] Nallapati, R., Cohen, W. and Lafferty, J. (2007) Parallelized Variational EM for Latent Dirichlet Allocation: An Experimental Evaluation of Speed and Scalability. *Seventh IEEE International Conference on Data Mining Workshops,* pp. 349-354.

[185] Wolfe, J., Haghighi, A. and Klein, D. (2008) Fully distributed EM for very large datasets. *ICML '08 Proceedings of the 25th international conference on Machine learning,* pp. 1184-1191.

[186] Apache Mahout: Scalable machine learning and data mining (2012) Available at: http://mahout.apache.org/ [14 January 2012].

[187] Zhai, K., Boyd-Graber, J., Asadi, N. and Alkhouja, M.L. (2012) Mr. LDA: a flexible large scale topic modeling package using variational inference in MapReduce. *Proceedings of the 21st international conference on World Wide Web,* pp. 879-888.

[188] Smola, A. and Narayanamurthy, S. (2010) An architecture for parallel topic models. *Proceedings of the VLDB Endowment,* 3 (1-2), pp. 703-710.

[189] Yahoo! Hadoop Tutorial (2012) Available at: http://developer.yahoo.com/hadoop/tutorial/ [10 February 2012].

[190] Dou, W.W., Wang, X.Y., Chang, R. and Ribarsky, W. (2011) ParallelTopics: A probabilistic approach to exploring document collections. *2011 IEEE Conference on Visual Analytics Science and Technology (VAST),* pp. 231-240.

[191] Zhang, D. et al., (2010) PTM: probabilistic topic mapping model for mining parallel document collections. *CIKM '10 Proceedings of the 19th ACM international conference on Information and knowledge management,* pp. 1653-1656.

[192] Intel Threading Building Blocks (Intel TBB) (2012) Available at: https://www.threadingbuildingblocks.org/ [15 February 2012].

[193] The Internet Communications Engine (ICE) (2012) Available at: http://www.zeroc.com/ice.html [18 February 2012].

[194] Asuncion, A., Smyth, P. and Welling, M. (2008) Asynchronous distributed learning of topic models. *In Proceedings of the Conference on Advances in Neural*

Information Processing Systems, pp. 1-8.

[195] Wang, Y., Bai, H., Stanton, M., Chen, W. and Chang, E. (2009) PLDA: Parallel latent dirichlet allocation for large-scale applications. *In Algorithmic Aspects in Information and Management*, pp. 301–314.

[196] Apache Hadoop (2012) Available at: http://hadoop.apache.org/ [28 January 2012].

[197] Guo, P.G., Wang, X.Z. and Han, Y.S. (2010) The enhanced genetic algorithms for the optimization design. *2010 3rd International Conference on Biomedical Engineering and Informatics (BMEI)*, 7, pp. 2990-2994.

[198] Jiang, W.J., Luo, D.T., Xu, Y.S. and Sun, X.M. (2004) Hybrid genetic algorithm research and its application in problem optimization. Fifth World Congress on Intelligent Control and Automation (WCICA) 2004, 3, pp. 2122-2126.

[199] Yazdi, H.M. (2013) *Genetic Algorithms In Designing And Optimizing Of Structures*. Saarbrücken: LAP LAMBERT Academic Publishing.

[200] Nadi, F. and Khader, A.T. (2011) A parameter-less genetic algorithm with customized crossover and mutation operators. *GECCO '11 Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pp. 901-908.

[201] Vafaee, F., Turán, G. and Nelson, P.C. (2010) Optimizing genetic operator rates using a markov chain model of genetic algorithms. *GECCO '10 Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pp. 721-728.

[202] Hammoud, S., Li, M.Z., Liu, Y. and Alham, N.K. (2010) MRSim: A discrete event based MapReduce simulator. *2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, 6, pp. 2993-2997.

[203] Liu, Y., Li, M.Z., Alham, N.K. and Hammoud, S. (2013) HSim: A MapReduce simulator in enabling Cloud Computing. *Future Generation Computer Systems*, 29 (1), pp. 300-308.

[204] Kumar, P. and Verma, A. (2012) Scheduling using improved genetic algorithm in cloud computing for independent tasks. *ICACCI '12 Proceedings of the International Conference on Advances in Computing, Communications and Informatics*, pp. 137-142.

[205] Adamuthe, A.C. and Bichkar, R.S. (2011) Minimizing job completion time in grid scheduling with resource and timing constraints using genetic algorithm. *ICWET '11 Proceedings of the International Conference & Workshop on Emerging Trends in*

Technology, pp. 338-343.

[206] Ramirez, A.J., Knoester, D.B., Cheng, B.H.C. and McKinley, P.K. (2009) Applying genetic algorithms to decision making in autonomic computing systems. *ICAC '09 Proceedings of the 6th international conference on Autonomic computing,* pp. 97-106.

[207] Nigam, K., McCallum, A.K., Thrun, S. and Mitchell, T.M. (2000) Text classification from labeled and unlabeled documents using EM. *Machine Learning,* 39(2/3), pp. 103–134.

[208] Blei, D.M. and McAuliffe, J.D. (2010) Supervised topic models. *Arxiv preprint arXiv:1003.0783,* pp. 1-22.

[209] Perotte, A.J., Wood, F., Elhadad, N. and Bartlett, N. (2011) Hierarchically Supervised Latent Dirichlet Allocation. *Advances in Neural Information Processing Systems,* pp. 2609-2617.

[210] Zhu, J., Ahmed, A. and Xing, E.P. (2012) MedLDA: maximum margin supervised topic models. *The Journal of Machine Learning Research,* 13 (1), pp. 2237-2278.

[211] Rosen-Zvi, M., Griffiths, T., Steyvers, M. and Smyth, P. (2004) The author-topic model for authors and documents. *UAI '04 Proceedings of the 20th conference on Uncertainty in artificial intelligence,* pp. 487-494.

[212] Rosen-Zvi, M. et al., (2010) Learning author-topic models from text corpora. *ACM Transactions on Information Systems (TOIS),* 28 (1), pp. 401-438.

[213] Steyvers, M., Smyth, P., Rosen-Zvi, M. and Griffiths, T. (2004) Probabilistic author-topic models for information discovery. *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining,* pp. 306-315.

[214] Ramage, D., Hall, D., Nallapati, R. and Manning, C.D. (2009) Labeled LDA: a supervised topic model for credit attribution in multi-labeled corpora. *EMNLP '09 Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing,* 1 (1), pp. 248-256.

[215] Wang, Tao., Yin, G., Li, X. and Wang, H.M. (2012) Labeled topic detection of open source software from mining mass textual project profiles. *SoftwareMining '12 Proceedings of the First International Workshop on Software Mining,* pp. 17-24.

[216] Liu, Z.Y., Zhang, Y.Z., Chang, E.Y. and Sun, M.S. (2011) PLDA+: Parallel

latent dirichlet allocation with data placement and pipeline processing. *ACM Transactions on Intelligent Systems and Technology (TIST),* 2 (3), pp. 2601-2618.

[217] Chang, E.Y., Bai, H.J. and Zhu, K.H. (2009) Parallel algorithms for mining large-scale rich-media data. *MM '09 Proceedings of the 17th ACM international conference on Multimedia,* pp. 917-918.

[218] Shu, W. and Kale, L.V. (1989) A dynamic scheduling strategy for the Chare-Kernel system. *Proceedings of the 1989 ACM/IEEE Conference on Supercomputing*, pp. 389-398.

[219] Viswanathan, S., Veeravalli, B., Yu, D.T. and Robertazzi, T.G. (2004) Design and analysis of a dynamic scheduling strategy with resource estimation for large-scale grid systems. Fifth IEEE/ACM International Workshop on Grid Computing, pp. 163-170.

[220] Cariño, R.L. and Banicescu, I. (2008) Dynamic load balancing with adaptive factoring methods in scientific applications. *The Journal of Supercomputing,* 44 (1), pp. 41-63
Rotaru, C. Brudaru, O. (2012) Dynamic segregative genetic algorithm for optimizing the variable ordering of ROBDDs. *GECCO '12 Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*, pp. 657-664.

[221] Kameda, H., Fathy, E.Z.S., Ryu, I. and Li, J. (2000) A performance comparison of dynamic vs. static load balancing policies in a mainframe-personal computer network model. *Proceedings of the 39th IEEE Conference on Decision and Control*, 2, pp. 1415-1420.

[222] Srinivas, M. and Patnaik, L.M. (1994) Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics,* 24 (4), pp. 656-667.

[223] Yang S.X. (2008) Genetic algorithms with memory-and elitism-based immigrants in dynamic environments. *Evolutionary Computation,* 16 (3), pp. 385-416.

[224] Milton, J. and Kennedy, P.J. (2010) Static and dynamic selection thresholds governing the accumulation of information in genetic algorithms using ranked populations. *Evolutionary Computation,* 18 (2), pp. 229-254.

[225] Peng, J.J. et al., (2009) Comparison of Several Cloud Computing Platforms. *2009 Second International Symposium on Information Science and Engineering (ISISE)*, pp. 23-27.

[226] Dikaiakos, M.D. et al., (2009) Cloud Computing: Distributed Internet Computing for IT and Scientific Research. *IEEE Internet Computing,* 13 (5), pp. 10-13.

[227] Buyya, R. et al., (2009) Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems,* 25 (6), pp. 599-616.

[228] Wang, X.G. and Grimson, E. (2007) Spatial Latent Dirichlet Allocation. *Advances in NIPS,* 20, pp. 1577-1584, [Online]. Available at: http://machinelearning.wustl.edu/mlpapers/paper_files/NIPS2007_102.pdf [Accessed 14 March 2013]

[229] Fei-Fei, L. and Perona, P. (2005) A Bayesian hierarchical model for learning natural scene categories. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) 2005,* 2, pp. 524-531.

[230] Vaduva, C., Gavat, I. and Datcu, M. (2013) Latent Dirichlet Allocation for Spatial Analysis of Satellite Images. IEEE Transactions on Geoscience and Remote Sensing, 51 (5), pp. 2770-2786.