THE CORRELATED TOPIC MODEL FOR PAPER RECOMMENDATION SYSTEM IN MAPREDUCE FRAMEWORK

MI KHINE OO

UNIVERSITY OF COMPUTER STUDIES, YANGON

MARCH, 2022

The Correlated Topic Model for Paper Recommendation System in MapReduce Framework

Mi Khine Oo

University of Computer Studies, Yangon

A thesis submitted to the University of Computer Studies, Yangon in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

March, 2022

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research and has not been submitted for a higher degree to any other University or Institution.

…..…… ……..……

Date Mi Khine Oo

ACKNOWLEDGEMENTS

To start with, I would like to thank the Union Minister, the Ministry of Science and Technology for the full facilities and supports during the Ph.D. course at the University of Computer Studies, Yangon.

First of all, I would like to express my inmost gratitude to Dr. Mie Mie Khin, Rector, the University of Computer Studies, Yangon, for allowing me to develop this research and giving me excellent guidance during the period of my dissertation. Further, I would like to offer my gratefully thanks to Dr. Mie Mie Thet Thwin, former Rector, the University of Computer Studies, Yangon, for her guidance and encouragement during the Ph.D. study.

Secondly, I would like to express my deepest gratitude to my supervisor Dr. Ah Nge Htwe, Professor, Course Coordinator of the Ph.D. 9th batch, the University of Computer Studies, Yangon, for her caring and encouragement, and providing me with excellent ideas and guidance during the time of writing this dissertation.

Thirdly, I would like to acknowledge and special thanks to Daw Aye Aye Khine, Associate Professor, English Department, the University of Computer Studies, Yangon. I would like to thank her for valuable supports and revising my dissertation from the language point of view.

In addition, I would like to extend my respectful appreciation to Dr. Khine Moe Nwe for her encouragement and special guidance during the development of this research. Also, I would like to offer my sincere gratitude to my former supervisors, Dr. Myo Kay Khaing and Dr. May Aye Khine, for their invaluable suggestions and taking care of me throughout my Ph.D. study.

Finally, I wish to express my appreciation to all the people who greatly contributed and supported throughout the writing of this dissertation.

Last but not least, I am deeply indebted to my beloved parents for their endless love and unconditional support. I would like to give my warmest thanks to my family who endured this long process with me. The completion of my dissertation would not have been possible without the support and nurturing of my family.

ABSTRACT

The emergence of internet technology provides access to an endless supply of data, information and knowledge to many different areas. Since the amount of digital information has been increased rapidly day by day, yielding the big data analytics becomes a problem for recommendation systems. With the availability of increasingly large quantities of digital information in academic area, it is becoming more difficult to find and extract relevant information pertinent to the interests. In addition, more efforts are required to summarize and understand large amount of digital documents. In this research, the correlated topic model (CTM) implemented in MapReduce framework is proposed for generating relevant recommendations within a short response time when the user provided the search query.

Firstly, the full-text documents of publicly available digital libraries are collected to improve the accuracy of recommendations. With the aim of extracting latent semantic topics from a collection of documents, the MapReduce CTM employs a variational Expectation-Maximization (variational EM) algorithm. When learned topics are coherent and interpretable, they may be valuable for the recommendations. To address the poor prediction problem of recommendation system, the information theoretic measure called entropy is proposed to measure the predictability between documents. Finally, when the user enters the search query, the semantic similarity between the search query and extracted topics are calculated for retrieving and recommending relevant documents in the top-N recommendations list.

For the evaluation of the MapReduce CTM model, the topic coherence measures, UCI and UMass, are used to investigate the semantic relatedness of the extracted topics. The results of MapReduce CTM are then compared with another topic model LDA, and observed that the proposed model learns more coherent and specific topics. Moreover, the processing time of MapReduce CTM for extracting the latent topics is also analysed. For the performance evaluation of the proposed paper recommendation system, the precision and recall metrics are used to evaluate the retrieval performance of recommendation system. According to the experimental results, the proposed paper recommendation system with incorporation of MapReduce CTM achieves the best possible performance in the quality of recommendations.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS i

ABSTRACT ii

TABLE OF CONTENTS iii

LIST OF FIGURES vi

LIST OF TABLES ix

LIST OF EQUATIONS xi

1. INTRODUCTION 1

1.1 Big Data …….. 3

1.2 Apache Hadoop Framework ..……. 7

1.3 Problem Definition ……. 9

1.4 Motivation of the Research ……. 11

1.5 Objectives of the Research ……. 11

1.6 Contributions of the Research …… 12

1.7 Organization of the Research ……. 13

2. LITERATURE REVIEW AND RELATED WORK 15

2.1 Recommendation Systems …….. 15

2.1.1 Types of Data ……..……. 18

2.1.1.1 Explicit Feedback …….. 18

2.1.1.2 Implicit Feedback …….. 19

2.1.2 Types of Recommendation Systems …….…….. 19

2.1.2.1 Content-based Filtering Recommendation Systems ….. 20

2.1.2.2 Collaborative Filtering Recommendation Systems ….... 22

2.1.2.3 Hybrid Recommendation Systems …… 26

2.2 Paper Recommendation Systems ..……. 28

2.3 Entropy in Recommendation Systems ……...……. 30

2.4 Chapter Summary …….…….. 32

3. THEORETICAL BACKGROUND 33

3.1 Topic Models …….……..... 33

3.1.1 Latent Semantic Analysis …..……..……... 36

3.1.2 Probabilistic Latent Semantic Analysis …….. 37
3.1.3 Latent Dirichlet Allocation ……. 39
3.1.4 Correlated Topic Model …….. 41
3.2 Apache Hadoop …….……..... 45
3.2.1 Hadoop Distributed File System …..……..…… 46
3.2.2 Hadoop MapReduce ……... 47
3.3 Chapter Summary ……... 50

4. THE ARCHITECTURE OF THE PROPOSED SYSTEM 51
4.1 Data Collection Stage ……. ……... 54
4.2 Offline Learning Stage ……... 54
4.2.1 Data Preprocessing ……..……... 55
4.2.1.1 Tokenization …….. 56
4.2.1.2 Words Elimination ……. 56
4.2.1.3 Stopwords Elimination …….. 57
4.2.1.4 Spell Checking ……... 58
4.2.2 Model Training ……... 58
4.2.2.1 Topics Extraction ……... 59
4.2.2.2 Topics Validation …….. 62
4.2.3 Similarity Computation …….. 63
4.3 Generating Recommendation Stage …….…. …… 64
4.3.1 Keyword Processing ……... 64
4.3.2 Documents Extraction …… 65
4.3.3 Predictability Computation ……. 66
4.4 Chapter Summary ……... 66

5. IMPLEMENTATION OF THE PROPOSED SYSTEM 68
5.1 Hadoop Cluster Configuration …… 68
5.1.1 Overview of Hadoop Cluster …….. 69
5.1.2 Prerequisites for Hadoop Cluster ……..... 70
5.1.3 Installation Steps for Hadoop Cluster ….…… 70
5.2 Implementation of MapReduce CTM ……...…….. 72
5.3 Implementation of Paper Recommendation System ……. 73
5.4 Chapter Summary ……... 74

6. EXPERIMENTAL RESULTS AND EVALUATIONS 75

6.1 Experimental Setup ……. 75

6.2 Datasets ……..…….. 76

6.3 Experimentation for MapReduce CTM …….……. 77

6.3.1 Topic Model Results ……... 78

6.3.2 Topics Coherence Metrics …….. 80

6.3.3 Topics Coherence Evaluations ……... 82

6.3.4 Comparison between MapReduce CTM and Mr. LDA ……. 94

6.3.5 Training Time Comparison …… 97

6.4 Experimentation for Paper Recommendation System ….……... 99

6.4.1 Evaluation Metrics …….. 100

6.4.2 Evaluation Results …….. 101

6.5 Chapter Summary ……... 104

7. CONCLUSION AND FUTURE WORKS 105

7.1 Advantages and Limitations of the Proposed System …….……... 105

7.2 Summary ……..…… 106

7.3 Future Works ……..... 107

AUTHOR'S PUBLICATIONS …….…….. 109

BIBLIOGRAPHY …… 110

ACRONYMS …… 121

LIST OF FIGURES

Figure 1.1	The 3Vs of Big Data ……..	5
Figure 1.2	The 9Vs of Big Data ……..	6
Figure 1.3	Overview of HDFS Architecture ……...	8
Figure 1.4	Processing View of MapReduce ……	8
Figure 1.5	Publications Per Year …….	10
Figure 2.1	Phases of Recommendation System ……..	16
Figure 2.2	General Model of a Recommendation System ……..	17
Figure 2.3	Content-based Filtering Approach …….	20
Figure 2.4	Collaborative Filtering Approach ……..	24
Figure 2.5	Four Aggregation Methods of Hybrid Approach ……...	27
Figure 3.1	Graphical Model of PLSA …….	37
Figure 3.2	Probabilistic Latent Semantic Analysis …….	39
Figure 3.3	Graphical Model of LDA ……...	39
Figure 3.4	Graphical Model of CTM ……..	42
Figure 3.5	High Level Architecture of Hadoop ……..	45
Figure 3.6	Architecture of HDFS ……	46
Figure 3.7	Architecture of MapReduce ……...	48
Figure 3.8	Canonical Example of a MapReduce Job ……..	50
Figure 4.1	Proposed Recommendation System Workflow …….	51
Figure 4.2	Architecture of Proposed System ……..	53
Figure 4.3	Data Collection Steps …….	54
Figure 4.4	Data Pre-processing Steps ……..	55
Figure 4.5	Procedure of Words Elimination ……	57
Figure 4.6	Procedure of Stopwords Elimination …….	57

Figure 4.7	Procedure of Spell-Checking …….	58
Figure 4.8	Sample Input and Outputs of MapReduce CTM ……...	59
Figure 4.9	Procedure of Driver Class ……..	60
Figure 4.10	Workflow of MapReduce CTM …….	60
Figure 4.11	Procedure of Mapper Class ……	61
Figure 4.12	Procedure of Reducer Class ……...	62
Figure 4.13	Topics Validation Process ……..	63
Figure 4.14	Procedure of Recommendation System …….	65
Figure 5.1	An Overview of the Hadoop Cluster …….	69
Figure 5.2	Web User Interface of the NameNode ……...	72
Figure 5.3	Implementation with MapReduce CTM ……	73
Figure 5.4	Recommendation UI of Paper Recommendation System ……..	74
Figure 6.1	Comparison of Mean Coherence Scores for CiteSeerX …….	93
Figure 6.2	Comparison of Mean Coherence Scores for PLOS ONE ……..	93
Figure 6.3	Comparison of Coherence Scores between MapReduce CTM and Mr. LDA for CiteSeerX ……..	95
Figure 6.4	Comparison of Coherence Scores between MapReduce CTM and Mr. LDA for PLOS ONE ……	96
Figure 6.5	Training Time of MapReduce CTM for CiteSeerX ……...	97
Figure 6.6	Training Time of MapReduce CTM for PLOS ONE ……	98
Figure 6.7	Training Time of MapReduce CTM and Mr. LDA for CiteSeerX	99
Figure 6.8	Training Time of MapReduce CTM and Mr. LDA for PLOS ONE ……...	99
Figure 6.9	Precision and Recall with top-100 topic words at cut-off 50 on CiteSeerX ……...	102
Figure 6.10	Precision and Recall with top-100 topic words at cut-off 100 on CiteSeerX ……...	102

Figure 6.11 Precision and Recall with top-100 topic words at cut-off 25 on PLOS ONE ……...…….. 103

LIST OF TABLES

Table 2.1	Popular E-commerce Websites ……	18
Table 2.2	User-Item Ratings Matrix ……	23
Table 3.1	Comparison between Topic Models ……	35
Table 3.2	Notations for PLSA Topic Model ……...	38
Table 3.3	Notations for LDA Topic Model …….	40
Table 3.4	Notations for CTM Topic Model ……	43
Table 5.1	Hadoop Cluster Description ……	70
Table 6.1	Specifications of Machines ……..	75
Table 6.2	Characteristics of Datasets (Before Pre-processing) ……...	76
Table 6.3	Characteristics of Datasets (After Pre-processing) ……..	76
Table 6.4	Comparison of File Sizes …….	77
Table 6.5	Extracted 10 topics for CiteSeerX ……...	78
Table 6.6	Extracted 10 topics for PLOS ONE …….	79
Table 6.7	Mean Coherence Scores of MapReduce CTM for CiteSeerX …….	82
Table 6.8	Extracted 2 topics ordered by UCI for CiteSeerX ……...	83
Table 6.9	Extracted 8 topics ordered by UMass for CiteSeerX ……..	83
Table 6.10	Word Counts for Number of Topics 2 and 8 for CiteSeerX ...…….	84
Table 6.11	Mean Coherence Scores of MapReduce CTM for PLOS ONE …..	85
Table 6.12	Extracted 5 topics ordered by UCI for PLOS ONE ……	85
Table 6.13	Extracted 4 topics ordered by UMass for PLOS ONE ……	86
Table 6.14	Word Counts for Number of Topics 5 and 4 for PLOS ONE ..…...	86
Table 6.15	Mean Coherence Scores of MapReduce CTM for CiteSeerX …….	87
Table 6.16	Extracted 5 topics ordered by UCI for CiteSeerX ……...	87
Table 6.17	Extracted 6 topics ordered by UMass for CiteSeerX ……..	88
Table 6.18	Word Counts for Number of Topics 5 and 6 for CiteSeerX ……	89

Table 6.19 Mean Coherence Scores of MapReduce CTM for PLOS ONE ….. 90

Table 6.20 Extracted 5 topics ordered by UCI for PLOS ONE …… 90

Table 6.21 Extracted 10 topics ordered by UMass for PLOS ONE ..…… 90

Table 6.22 Word Counts for Number of Topics 5 and 10 for PLOS ONE …... 92

Table 6.23 Mean Coherence Scores of Mr. LDA for CiteSeerX and PLOS ONE ……. 95

LIST OF EQUATIONS

Equation 3.1 ……... 38

Equation 3.2 ……... 38

Equation 3.3 ……... 41

Equation 3.4 ……... 41

Equation 3.5 ……... 43

Equation 3.6 ……... 44

Equation 3.7 ……... 44

Equation 4.1 ……... 64

Equation 4.2 ……... 66

Equation 4.3 …….. 66

Equation 6.1 ……..…….. 81

Equation 6.2 ……... 81

Equation 6.3 ……... 81

Equation 6.4 ……... 100

Equation 6.5 ……... 100

CHAPTER 1

INTRODUCTION

Over the years, the Internet technology has become an influential platform for the expeditious flow of information. This modernistic technology has given a new dimension to share the information and has made possible to access the shared information. Especially, the immense contribution of the Internet to computer science and technology field brings bulks of data and information that are sufficiently large and complex, and thus requires massive enhancements in the abilities of human to explore the required information efficiently and effectively. In other words, the enormous increase of data and information causes the need of powerful machine learning techniques and data analytics tools. This is one of the factors that contributes to the emergence of today's big data ecosystem.

Big data is the rapid increasing of heterogeneous, structured and unstructured digital contents that make processing more difficult to understand and extract valuable information. When the information demonstrates volume, variety and velocity, then it is interpreted as big data. Big data mostly involves bunches of unstructured data in comparison with traditional datasets [62]. To summarize the phenomenon of big data, it is about not only the large amounts of data but also running analytics on those large data sets. With the rapid expansion of various big data, users encounter with challenges in discovering and examining information that is relevant to their purposes. Although the researchers can find more information from this growth, they need new ways to deal with the immense inflow of information.

Recommender systems have been widely applied in academic, commercial and industrial fields to figure out the information explosion problem by filtering necessity information from huge amounts of data. The objective is to generate meaningful recommendations to users in accordance with the observed behaviors of users about item to increase satisfaction. Generally, there are basically three categories for generating recommendations. A recommendation system could be a Content-based, a Collaborative Filtering (CF) based or a Hybrid approach [38].

Content-based recommender system matches the profile of the user and some specific characteristics of the item in order to recommend items with similar

characteristics. However, it might not be easy to gather the characteristics or metadata of an item. An alternative to content-based approach is known as collaborative filtering (CF) recommendation approach that filters information based on the collaboration of users or the similarity between items [95]. In traditional CF, since the similarity is calculated based on the past behavior or rating information of users, the features of items that describe the semantic relationships are not considered. Hybrid recommender system combines the content-based and CF approaches to solve the weaknesses of each approach. Among these approaches, CF is the most widely applied technique. For immense data sets, CF methods provide an increase in the accuracy and performance. However, the CF mostly encounters the problems of cold start, sparsity and scalability issues.

From a collection of documents, the probabilistic topic models reveal the underlying thematic structures through extracting topics. With the help of these extracted topics, the whole document collection can be summarized and categorized without human annotation effort. Latent Dirichlet Allocation (LDA) proposed in Blei et al. [12], one of the most widely known topic models, uses statistical methods to infer the latent topics involved in the document collection. A limitation of LDA is the incapability of modeling topic correlations because LDA models the topic proportions by using a Dirichlet distribution. The distribution assumes that the presence of one topic is not correlated with the presence of another because of its independence structure. However, the latent topics can have relations between them and correlate with each other in many realistic applications. The Correlated Topic Model (CTM), which was first proposed in Blei et al. [11], resolves this limitation by replacing the Dirichlet distribution with the logistic normal distribution to uncover the correlations of latent topics.

In order to discover the latent topics, the CTM may have a challenge in calculating the posterior distribution of topics over the observed words. Different inference algorithms have been proposed to figure out the model parameters estimation, including Gibbs Sampling and Variational Expectation-Maximization (variational EM) [40]. Gibbs sampling is a Markov Chain Monte Carlo algorithm which iteratively draws instances from probability distributions. Variational EM algorithm relies in computing the maximum likelihood estimation of latent variables. In this proposed approach, the variational EM algorithm is practiced for the analysis.

In summary, the proposed approach is intended to contribute a solution to the above problems by proposing a paper recommendation system which incorporates CTM. The CTM is utilized to learn the semantic relationship between items in order to recommend the latent documents. For the inferencing process of CTM, the variational EM algorithm is proposed using Map and Reduce functions leveraging MapReduce paradigm in a Hadoop cluster. Then, the extracted topic distributions resulted from the MapReduce CTM are used to compute the similarity between the topics. The similarity calculation is performed by comparing the latent topic distributions of two items. In addition, the entropy is applied during the process of recommendation generation to improve the predictability of the paper recommendation system.

Besides, since big data produces enormous data size, the performance may be slow if the recommendation process has been done in a single system. To manage this slowness problem, a distributed environment is required to boost the recommendations generation process. An open-source big data platform Hadoop provides distributed processing in minimum amount of time by using MapReduce implementation [60]. Therefore, the proposed system has been implemented on Hadoop platform. The experiments are executed using the crawled datasets from the academic publicly available digital libraries, which are called as big data.

1.1 Big Data

Big data is a collection of unstructured data that is huge in volume, complex and grow exponentially with time that it becomes difficult to collect, store and process using traditional data processing approaches and software technologies. The so-called big data enables to collect more and more data through disparate forms, provide a set of tools to analyze the data, infer various structures in the data and interpret that knowledge in various forms. Big data comes from different sources, such as Websites, social networks, desktop and mobile applications, scientific experiments and repositories, and machine-generated data used in the internet of things environments.

Zikopoulos et al. [116] stated that Big data is a combination of structured, semi-structured and unstructured data.

- Structured data has formal schema as well as similar formats and predefined lengths. It has been formatted and transformed into a well-defined data model,

and are generated by humans, machines and automatic data generators. Examples of structured data include SQL relational databases, barcodes and spreadsheets.

- Semi-structured data or partially structured data need not to have a predefined length or type. It has some consistent and definite characteristics that make it easier to analyze. Examples include XML and JSON data.
- Unstructured data does not fit into any pre-defined format and does not have a predefined data model, and thus it is hard to analyze. Satellite images, audios, videos, social media data and IoT data are examples of unstructured data.

Many definitions have evolved for big data, for example, TechAmerica Foundation [97] presents big data as follows:

> "Big data is a term that describes large volumes of high velocity, complex and variable data that require advanced techniques and technologies to enable the capture, storage, distribution, management, and analysis of the information."

Similarly, Dumbill [22] defines big data as:

> "Big data is data that exceeds the processing capacity of conventional database systems. The data is huge and massive, moves at a very high speed, or does not fit the structures of existing database architectures. To gain value from these data, there must be an alternative way to process it."

Dave [20] observed that big data can be represented by three standard characteristics, or the three Vs as shown in Figure 1.1. The three Vs of big data includes:

- Volume – The huge amount of data that is expanding enormously. It is the massive amount of data generated from different fields around the world in every second.
- Velocity – The high speed of data being created and generated at faster pace. There is a continuous flow of data at which the data is to be processed to meet the challenges of big data.
- Variety – The nature of data that comes from heterogeneous sources. This data can be structured and semi-structured and mostly unstructured data.

Furthermore, in McNulty [62], four more Vs are defined to represent big data:

- Variability – The spread or dispersion of data. The data whose structure changes constantly with inconsistent speed.
- Veracity – The inconsistency, uncertainty and abnormality in data. The quality or trustworthiness of data to derive useful insights.
- Visualization – Different ways of representing the data to be easily understandable, comprehensible and accessible for decision making purposes.
- Value – The most important part of big data. Data needs to be processed and converted correctly into something valuable to extract value from massive data.

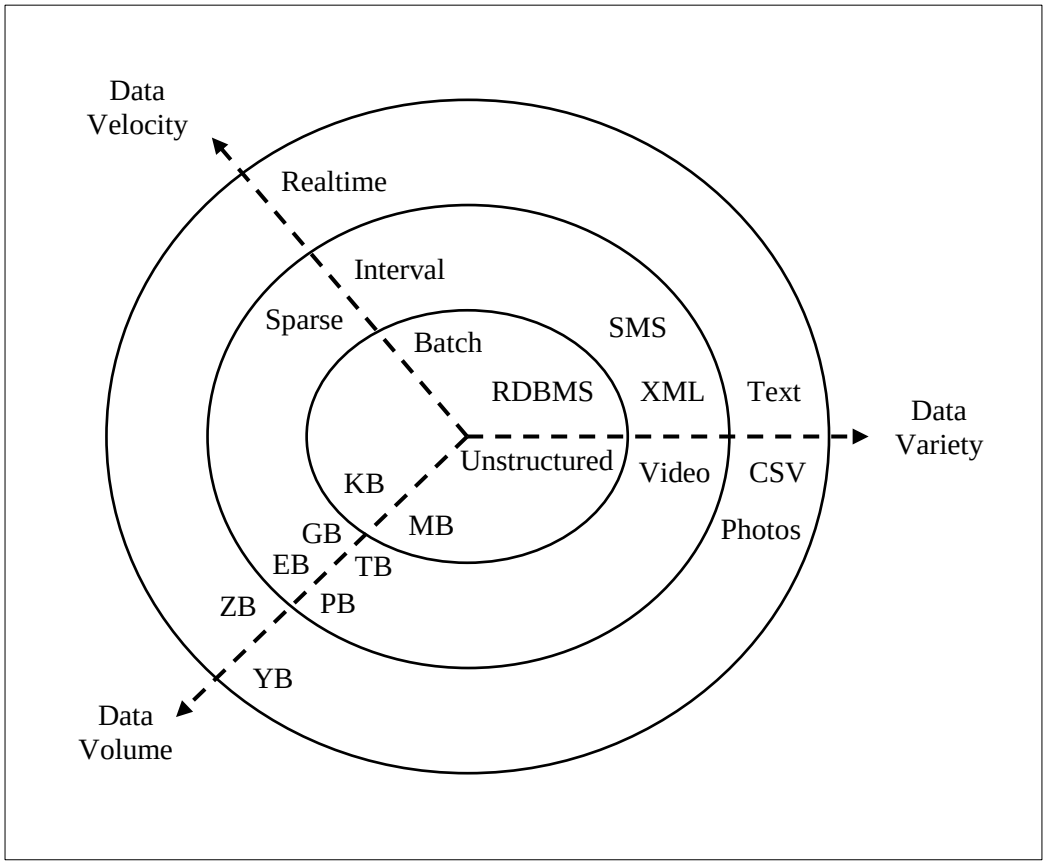


Figure 1.1 The 3Vs of Big Data

In addition to the above 7Vs, Wu et al. [109] transformed the 7Vs characteristics of big data into the multi-V (9Vs) characteristics by adding two new characteristics including validity and verdict. The 9Vs define big data in three aspects of domains: data domain, business intelligent domain and statistics domain. Figure 1.2 shows the 9Vs of big data characteristics.

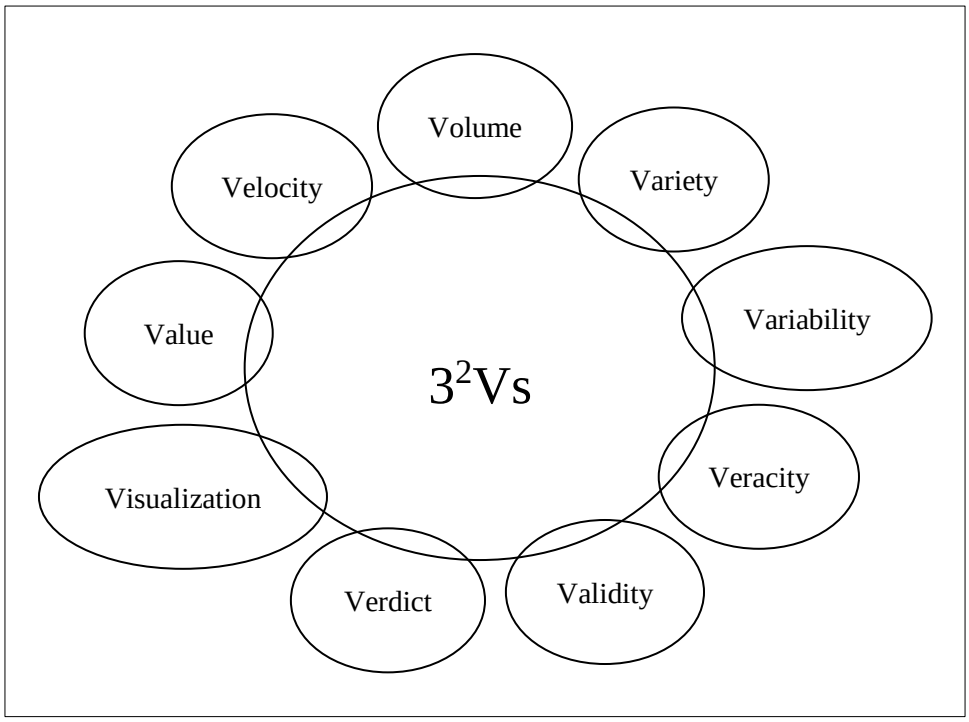


Figure 1.2 The 9Vs of Big Data

In Fan et al. [23], there are many research challenges in the field of big data. Nowadays, the improvement of information technology brings enormous challenges to data collecting, storing, processing, organizing and maintaining of the explosive growing big data. Singh et al. [90] discussed a list of key challenges in the analytics of big data as follows:

- Distributed data source: As big data constitutes various data sets which come from multiple disparate sources, and thus, big data has heterogeneity in data type, structure, and locations.
- Unstructured nature of data: Almost 80% of big data are semi-structured and unstructured data, and for this reason, this data is required to transform or consolidate into an appropriate structured data in order to make the processing more efficient.
- Time evolving data: Since big data is changing constantly and increasing dramatically over time, effective algorithms and methods are needed to adapt with this time evolving data.

- Analytics architecture: The analytics architecture for adopting big data should be able to perform scalable and speedy processing to improve efficiency in managing heterogeneous big data sets and respond with a timely manner.
- Data compression: Huge sets of data can contain redundant data, and consequently, they take more space for storage. To reduce the storage cost of the system, data compression is needed.
- Privacy: Maintaining and securing sensitive and private data plays a crucial role in big data. Hence, proper protection measures, such as data encryption and data segregation, are required when transferring data between various sources.

1.2 Apache Hadoop Framework

Big data refers not only to the increasing volume of data but also the technology to store, process and analyze that huge data. Apache Hadoop, one of the most widely used big data technologies, is an open-source framework to work with big data. An entire ecosystem of big data tools and technologies is built around Hadoop to provide distributed storage and parallel processing of large amounts of unstructured data on a cluster of nodes, where each node offers local computation and storage. Anshudeep [2] presented that the architecture of Hadoop ecosystem includes the following core components:

- Hadoop Common – A pre-defined set of utilities and libraries that provide underlying capabilities required by the other pieces of Hadoop within the Hadoop ecosystem.
- Hadoop Distributed File System (HDFS) – A file system that manages storage of and access to data distributed across the various nodes of a Hadoop cluster.
- Hadoop YARN – Hadoop's cluster resource manager, responsible for allocating system resources to applications and scheduling jobs.
- Hadoop MapReduce – A programming framework and processing engine used to run large-scale batch applications in Hadoop systems.

From the Hadoop's infrastructural point of view, a cluster of the HDFS architecture comprises of a single NameNode (Master node) and a number of DataNodes (Slave nodes). The NameNode is responsible for executing file system namespace (NS) operations and determining the mapping of blocks to DataNodes. The DataNodes are responsible for accomplishing block creation, deletion and replication

upon instructions from the NameNode [4]. Figure 1.3 shows the architecture of the HDFS file system.

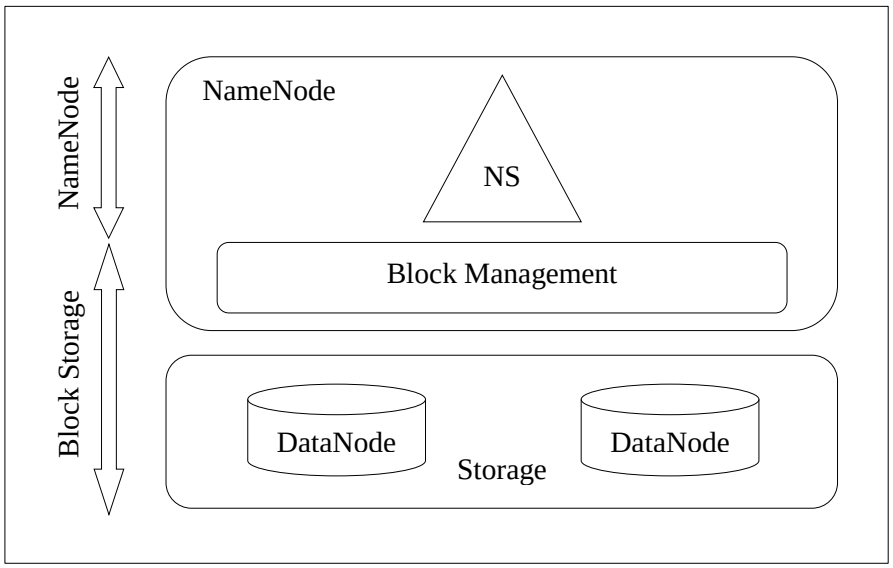


Figure 1.3 Overview of HDFS Architecture

From the programming point of view, the MapReduce is a parallel programming framework to distribute processing for big data in a Hadoop cluster containing multiple nodes [26]. The data processing can be done on both structured and unstructured data by using two distinct phases, Map and Reduce. The Map phase accepts input data from HDFS, splits them into chunks, assigns a map task for each chunk and produces intermediate results as <key, value> pairs. The Reduce phase combines intermediate outputs from the Map phase and produces the final output. Figure 1.4 illustrates the processing view of Map and Reduce phases.

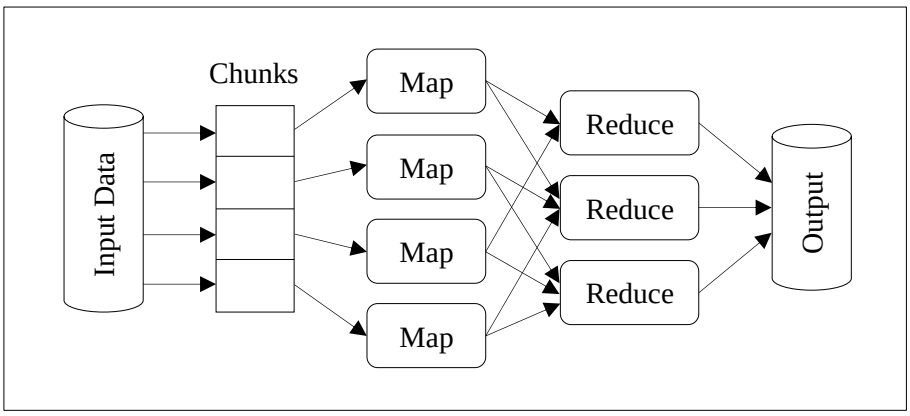


Figure 1.4 Processing View of MapReduce

1.3 Problem Definition

With the tremendous increase of Internet technology, the information overload problem becomes a major point of concern. The availability of vast amount of data and information that is beyond the manageable limits of the user expresses the problem of information overload. In fact, this problem occurs when the system is unable to handle and process this huge data in a systematic manner. Therefore, a powerful recommendation system is required to tackle the information overload problem. Apart from anything else, the reason is that when having more data, the analysis can be more accurate. Again, this increased accuracy leads to a more beneficial recommendation system as a final result.

Among the various approaches of recommendation system, the CF stands with great success such that this approach purely relies on user-item interactions as well as users' past behaviors expressed in the form of ratings. In most CF-based recommendation systems, the data sparsity is one of the main issues that prevent generating accurate predictions. For the reason the rating matrix becomes sparse is that the user provides ratings only to a small number of all items. Moreover, the cold-start problem is another prominent issue when a new item with lack of user interactions is placed into the system. To simplify, the two issues are related to large items dimensionality. Therefore, a precomputing of all the recommendations is a desirable way to alleviate these issues.

In addition, huge amounts of data are now flowing in from various sources to every field, especially to research and science field. Figure 1.5 depicts the total number of conference papers and journal articles being published in the past 18 years [102]. The amount of electronically available scientific documents motivates researchers to propose new computational and statistical methods to collect, analyze and interpret these documents. In other words, the more data are stored digitally, the more difficult to understand these data and search the specific document. To overcome this problem, topic models are proposed to explore and understand the latent themes of a large document collection. Topic models have the potential to improve the search and retrieve processes of recommendation system by discovering the hidden semantic themes from a collection of documents.

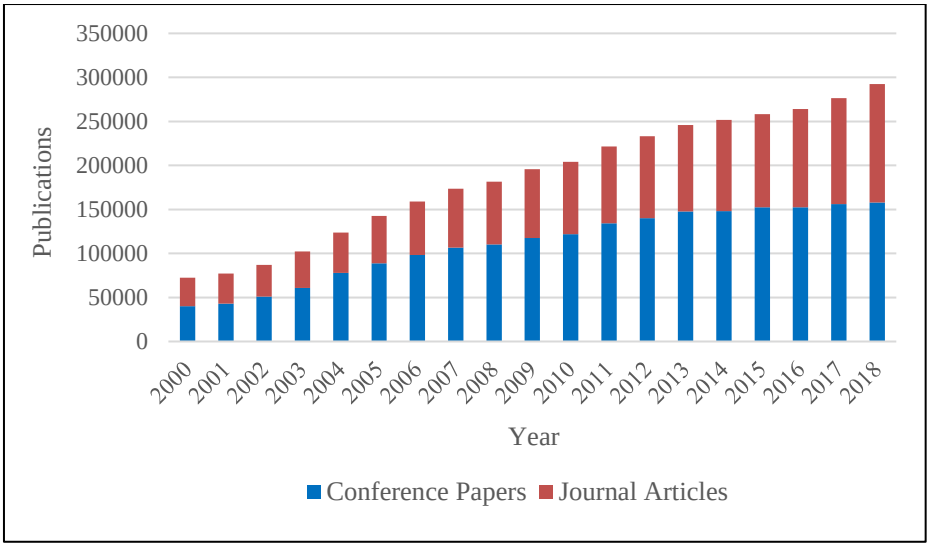


Figure 1.5 Publications Per Year

Besides, there is an increasing interest to process and store the massive amounts of data, also called big data, which are generated from the today's Information Age. The existing traditional data storage systems are not appropriate and impossible for keeping this growth of information. Therefore, a powerful infrastructure is needed to store, manage and retrieve valuable insight from big data. The Hadoop is used as a computing and storage framework which is connected to a cluster of computing nodes. Each node has storage capacity and compute power to enable the processing and analytics of big quantities of data.

The notable problem definitions related to the research work consist of:

Problem 1. Generating recommendations for the user by only considering the user's explicit preferences as well as the implicit preferences.

Problem 2. Recommending the most relevant items to the user within optimal response time.

Problem 3. Recommending the new unexpected and latent items semantically related to the provided user's query.

The main idea of this research work is to develop a paper recommendation system with the incorporation of CTM implementing Hadoop platform that can able to satisfy the tastes and needs of the users of online community.

1.4 Motivation of the Research

With the development of network and storage technology, the amount of digital information is exclusively increasing. In order to deal with continuously growing amounts of data, the design of recommendation system has become important. Therefore, a powerful recommendation system is needed to achieve the performance over big data. Moreover, the primary barrier of recommendation techniques over big data is generating relevant and meaningful recommendations from unstructured data within acceptable time. For this reason, a paper recommendation system is facilitated to extract valuable information and generate predictions for the users.

Topic modeling enables to discover the hidden topic structure of the data the data without requiring any supervision, which is rather useful for handling the big data at a scale that are hardly annotated with human power. However, the real-world applications of topic modeling are limited because of the scalability issues. When the size of documents collection arrives at a quite huge level, the scalability of topic modeling is needed to scale to the large document collections by using a distributed computing framework. Therefore, it is highly desirable to implement parallelized topic modeling algorithms in the MapReduce programming framework. In this system, a distributed learning algorithm for CTM is proposed to increase the scalability by adopting the variational EM inference approach.

1.5 Objectives of the Research

The main objective for this research is to build a high-quality paper recommendation system to generate predictions for users by incorporating the semantic relationships between the items. The next aim of this research is to improve the recommendation system's accuracy and performance by implementing the proposed variational EM algorithm in MapReduce framework. The last one is proposing a collaborative filtering algorithm with entropy measure to produce more precise predictions.

Other objectives of this research work are raised as follows:

i. To develop a paper recommendation system for big data that is able to recommend the unseen and latent articles.

ii. To study the effect of using CTM in a paper recommendation system when generating recommendations.
iii. To implement a CTM in MapReduce to extract the structure of the data without any explicit understanding of the data.
iv. To develop a CTM model in MapReduce to solve the bottlenecks of big data in text analytics.
v. To apply the entropy in generating recommendations to increase the accuracy of the recommendation system.
vi. To generate semantically related recommendations to satisfy the needs of users of online community.

1.6 Contributions of the Research

This research focuses on proposing a paper recommendation system and implementing the system in a Hadoop cluster to recommend unseen and latent articles in an effective and efficient manner. The main contribution of this research is, in the area of applying the Correlated Topic Model (CTM), to enhance the performance of paper recommendation system. Recommending scientific papers to the users of the recommendation system is a rigorous and demanding task because of the following challenges:

- The majority of the users do not rate most of the items and thus the ratings matrix becomes very sparse. This is called the sparse matrix problem.
- A new item cannot be recommended to the users when it is introduced to a CF system with no ratings. This is known as the cold-start problem.
- When the user's keywords have similar meanings, that can cause the recommendation system difficult to generate predictions. Also called the synonymy problem.
- When the real-world datasets are too large to process in a single system and thus affect the scalability of the recommendation system. This is also known as the scalability problem.

The proposed paper recommendation system integrates the Correlated Topic Model (CTM) to provide the best recommendation results to the user. Firstly, the variational Expectation-Maximization (variational EM) algorithm is utilized for

inferencing of CTM. Secondly, the similarity between items is calculated using the topic distributions obtained from the CTM. Thirdly, the entropy is calculated to ensure the predictability of items. And finally, the recommendations are generated and the performance evaluations are executed.

The proposed system mainly considers the following contributions:

i. A web crawler is developed to collect only the full-text documents from diverse domains in order to increase the accuracy of the recommendation system.
ii. The CTM is used to extract the latent topics of the documents collection that can predict unseen items and provide meaningful recommendations.
iii. A variational Expectation-Maximization (variational EM) algorithm implemented in MapReduce framework is proposed to distribute the inferencing process and to adapt with big data.
iv. The Cosine similarity is used to calculate the topic similarities using the topic distributions of the resulting topics.
v. The information theoretic measure called entropy is used in generating recommendations to identify the predictability relationship of one item to another.
vi. The topic coherence measures, UCI and UMass, are used to evaluate the performance of the proposed MapReduce CTM model and compare it with an existing topic model.
vii. The precision and recall metrics are applied to evaluate the quality and performance of the proposed paper recommendation system.

1.7 Organization of the Research

This dissertation is organized and structured with seven chapters.

Chapter 1 outlines the study areas, the motivations, the research issues and the goals and aims of the study. An overview of the methodology and the contributions of the research work are also presented.

Chapter 2 reviews the different approaches of recommendation systems exist in the literature to build the foundation of this study. A thorough studying of the previous researches related to the paper recommendation systems that have been conducted thus

far are also delivered. Additionally, the related works of entropy in recommendation systems are reflected.

Chapter 3 focuses on the theoretical background of the four topic models and highlights the various types of topic models. In addition, the architecture of Apache Hadoop and its principles and components are also completely covered. Overall, this chapter discusses the basic concepts that are important for understanding this research.

Chapter 4 presents the methodology which includes the overall architecture about how the proposed approach was built using MapReduce framework and how the proposed paper recommendation system was implemented.

Chapter 5 describes the detailed implementations of the proposed paper recommendation system and splits into two parts, one for the configuration setting for offline learning stage and the other for the implementation of generating recommendations stage, with rigorous illustrations.

Chapter 6 explains the experimental results and evaluations of the proposed approach. It has mainly two experimentations. The former is the experiments and evaluations of MapReduce CTM along with the comparison of another topic model. The latter is the experiments and evaluations of proposed paper recommendation system.

Finally, Chapter 7 brings to the conclusion of the research work, points out the advantages and limitations of this work and accomplishes with future research directions of this work.

CHAPTER 2

LITERATURE REVIEW AND RELATED WORK

This chapter studies the details of recommendation approaches and describes the works of previous researchers in the development of recommendation systems to accommodate with big data and Hadoop technologies. There have been much done in the recommendation systems area over the past years that have used a broad range of statistics, such as machine learning, information retrieval and other techniques, to advance the state-of-the-art recommendation systems.

In this chapter, first, a thorough study of recommendation systems based on their algorithms and methods is discussed. Then, the literature review of paper recommendation systems is completed by discussing the previous studies. There are not many academic papers related to paper recommendation systems in MapReduce framework. Finally, the use of entropy in recommendation purpose is provided. Based on the interdisciplinary nature of this study, the three related areas are covered in this literature review:

i. Recommendation systems
ii. Paper recommendation systems
iii. Entropy in recommendation systems

2.1 Recommendation Systems

Recommendation systems are changing the way of communication between Internet-based services and users. The recommendation systems are a subset of information filtering systems that the preferences or interests of the user that are given to the items are expecting to be involved in recommendations production [103]. They have a great number of applications in many fields, such as economic, e-commerce, education and research. A recommendation system can be defined as a set of tools and algorithms which aim to suggest the most relevant and appropriate items to the user by filtering useful information from a large data collection. The suggestions made by recommendation systems are related with decision making process, such as items to be bought [46]. In a recommendation system, the term ‘item’ is used to indicate what the system recommends to the user, such as a product, a movie, etc.

The output from a recommendation system can be either rating prediction or item recommendation. Most of the existing recommendation systems are designed to produce one of these outputs. A typical recommendation system is composed of three phases. The primary phases of a recommendation system are depicted in Figure 2.1 and are explained as follows:

1. **Information Collection**: The user's information including attributes, preferences or behaviours are gathered to create a profile of the user or a model for the next phases. The collection of information is influential so that the recommendation system generates appropriate outputs to the user. This phase can be completed with either explicit or implicit feedbacks or both.
2. **Learning**: The input data obtained from the information collection phase are utilized and exploited by employing a learning approach or algorithm.
3. **Prediction/Recommendation**: The system predicts a rating or recommends a list of top items to the user that the user may have preferences. In the case of rating prediction, the user ratings for the items that they have not yet rated are predicted. And in the task of item recommendations, a list of top-N items for each user that is anticipated to be liked by the user is generated.

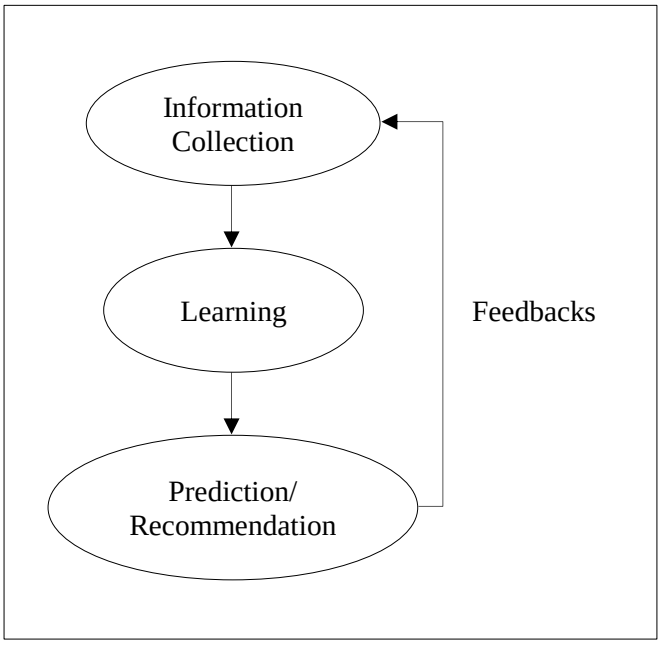


Figure 2.1 Phases of Recommendation System

The following Figure 2.2 illustrates the general model of a typical recommendation system [41]. In a model of a recommendation system, the users and items are emphasized as a critical role. The user's profile is matched with the items' descriptions and other users' profiles to generate the recommended items and neighbouring users, respectively. Broadly speaking, a recommendation system focuses on the type of data, the techniques that are used to generate recommendations, the customized design of the system and the graphical user interface (GUI).

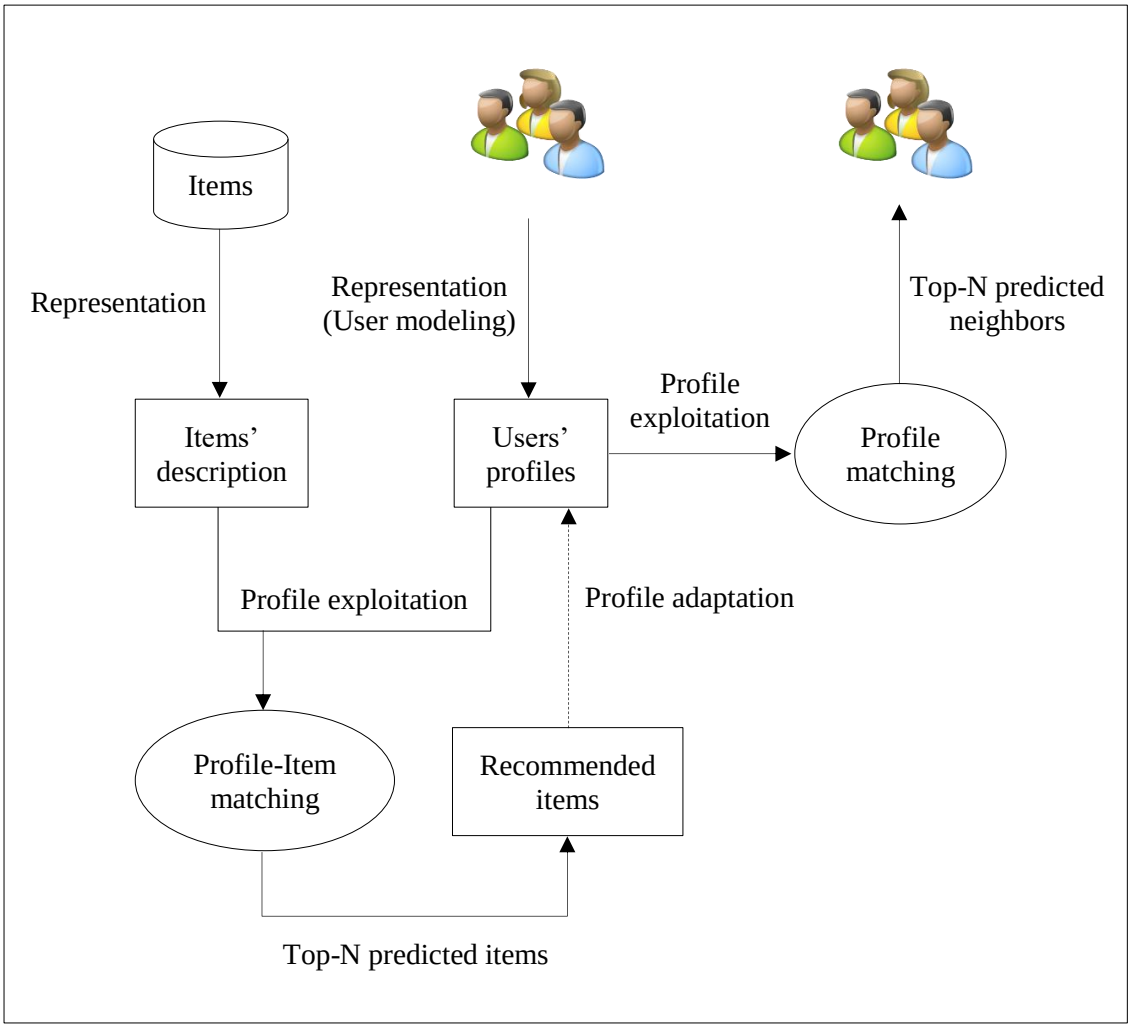


Figure 2.2 General Model of a Recommendation System

In recent years, the recommendation systems become a very popular and successful research topic in many e-commerce applications. The most popular e-commerce websites are described in the following Table 2.1.

Table 2.1 Popular E-commerce Websites

Websites	Category
Amazon, eBay, Alibaba	Shopping
Pinterest, Shopee	Social shopping
TripAdvisor	Travel
Netflix, YouTube	Movie
Spotify, Pandora, Last.fm	Music

The types of data inputted to typical recommendation systems and the explanations and practical reviews for the state of the art of the types of recommendation system adopted in several application domains are presented in the following subsections.

2.1.1 Types of Data

As mentioned, the recommendation systems play a vital role in many e-commerce websites and applications. Generally, the algorithmic task of a recommendation system is to predict the relevant items for a user by creating a list of recommended items. The recommendations are made based on various kinds of available data, depends on the needs and situations.

Typically, two types of data are collected and processed by a recommendation system, which are explicit feedbacks (votes, ratings) and implicit feedbacks (clicks, views, purchases, time stayed). A comparative analysis between these two feedbacks can be found in Montaner et al. [66]. There are many different recommendation systems proposed for modeling one of these feedbacks. Only few have focused on using both types of feedbacks [17, 45, 51, 55]. In the following, these two feedbacks will be explained in more detail with their characteristics.

2.1.1.1 Explicit Feedback

Explicit feedback, a real number given by a user to express his/her preference about an item is liked or disliked, is hard to collect because the additional input is required from the users [5]. In order to collect the explicit feedbacks, the

recommendation system requires to utilize the GUI to provide the user's preference for the items. Mostly, the explicit feedbacks are given as numeric ratings from users only when they have time to enter the ratings. Therefore, the amount of explicit data collected is extremely sparse, scarce and quantifiable.

Furthermore, the accuracy of the recommendation system depends on the quantity of users' preferences. However, the explicit feedback data is more accurate than the implicit one because the value of explicit feedback gives evidence of the user's actual interests. The explicit feedbacks are more widely used in many recommendation systems. Examples of explicit feedback include ratings of products on Amazon and ratings of movies on Netflix.

2.1.1.2 Implicit Feedback

Implicit feedback, an indirect way of inferring the interests and preferences of users through monitoring and observing users' actions and behaviours, is easier to collect in greater quantities without requiring any additional input from the users [39]. Examples of implicit feedback include viewing the detailed product description of an item or putting the item on a wish list on Amazon. The implicit data does not have negative feedback, that is, only positive instances are discovered.

Compared with explicit feedback data, the implicit feedback data can be available in abundance. However, the implicit data can be less accurate and can have redundancy because the data can be extracted from the navigation of the database, transaction logs and server logs. Therefore, the recommendation system is needed to be efficient to automatically monitor the different actions of users.

2.1.2 Types of Recommendation Systems

In this subsection, the major categories of recommendation system are reviewed with their related works. The types of recommendation approaches can be classified into content-based filtering (CBF), collaborative filtering (CF) and hybrid recommendation systems.

2.1.2.1 Content-based Filtering Recommendation Systems

A content-based filtering (CBF) recommendation system recommends items to the users based on the correlation between a series of discrete characteristics or features of the items and the profiles of users, each profile describes each user's preferences [74]. That is, the CBF approach uses additional information about an item in order to recommend additional items with similar characteristics to the active user [81]. This kind of recommendation system has been used in a diversity of domains, such as suggesting web pages for browsing, products for buying, news articles for reading, videos for watching, and music for listening. Figure 2.3 shows the general concept of CBF recommendation system.

For instance, in a music recommendation system, the additional information may be the title, artist, album or other characteristics of the song (item). The idea is to construct a model based on the available characteristics of the item. Although the details of CBF systems have differences, in this example, the CBF first describes the items that may be recommended, then creates a profile of the user to describe the types of items that the user likes, and finally compares items to the user profile to determine relevant items to recommend.

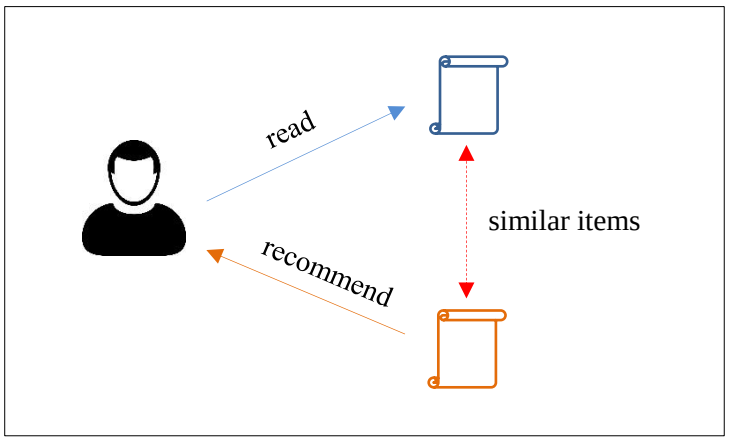


Figure 2.3 Content-based Filtering Approach

To sum up, the CBF approach is computationally fast and interpretable, and easily adapts to new items. However, collecting the addition information of items may be complicated because selecting the important features of items is difficult in CBF approaches [91], for example, the characteristics of multimedia data may not be able to

automatically extract, such as graphics and sound. The adoption of the CBF recommendation approach has some advantages:

- The CBF exploits only the active user's ratings to build the user's profile, it does not depend on other users.
- The CBF is understandable, that is, the explanation of recommendations is provided by listing the same features of an item that is occurred in the recommendation results.
- The CBF can recommend new items that are not yet rated by any user.

Nonetheless, Lops et al. [56] reported that the CBF recommendation systems have several shortcomings:

- The CBF cannot provide suitable recommendations to the user if the item does not contain enough information to separate items the user likes from items the user does not like.
- The CBF cannot provide reliable recommendations to the new user when few ratings are available or the user's information is not available.
- The CBF can recommend too similar items to the user, that is, if the characteristics of two different items are expressed with the same words, then these two items will not be distinguished.

The CBF recommendation systems, developed since the mid 90's, were the first approach to recommender systems and are used in several application areas to help users to find information on the Web. In Mooney et al. [67], a content-based book recommendation system for text categorization was developed that was able to recommend previously unrated items to the users. The system utilized a machine learning methodology to extract text data from the web pages. The user was asked to provide a rating for the selected books and the user model was constructed using a Bayesian learning algorithm. In Meteren et al. [64], a recommender system PRES was described that utilizes CBF techniques in order to recommend documents consisting of textual information about home improvements. The system compared the contents of each document with a user's profile to make recommendations.

For multimedia-related recommendations, a music recommendation system was proposed in Kim et al. [43] using a CBF method that was implemented by a ubiquitous computing technology. The system considers additional parameters that affect the

preferences of the user and creates a model based on these parameters to provide more accurate music recommendations. In Ma [58], a content-based movie recommendation system was developed for the Vionel movie website. The system extracted the features of the movies, and uses them to create the model and to calculate similarity between movies.

Under the explosive growth of web logs, RSS feeds and other sources of Internet contents, a content-based RSS aggregator (Cobra) proposed in Rose et al. [83] is implemented using CBF method to crawl, filter and aggregate vast numbers of RSS feeds, and to provide a personalized recommendation to the user based on the user's interests. For the offers in e-commerce services, a coupon recommendation system of Xia et al. [111] was proposed to provide a personalized recommendation in order to select the coupon efficiently and to improve the clickthrough rate. The system utilized the noisy implicit feedback to build the CBF coupon recommendation model.

2.1.2.2 Collaborative Filtering Recommendation Systems

A collaborative filtering (CF) recommendation system recommends items to the active user based upon the basis of past preferences of other users with similar tastes [14]. In other words, CF constructs the user-item ratings matrix that represent the users' historical preferences on the items, such as rated, liked, clicked, watched, etc. Then CF matches the users with similar preferences by computing the similarities of the users' profiles to predict ratings or recommend items [46]. The past user-item interactions are presented in a user-item ratings matrix. The following facts are required to consider regarding with the user-item interactions:

- The type of interaction to collect should be defined with respect to the kind recommendation system to develop, either explicit or implicit.
- The number of interactions between the users and items should be larger so that the better results are generated.

An example of user-item ratings matrix is shown in Table 2.2. Each element (cell) of the matrix is the value of the rating representing the evaluation of the user to the item, which could be a number (0 or 1) or just a number between 1 to 5 to represent the different evaluation degrees.

Table 2.2 User-Item Ratings Matrix

	Item 1	**Item 2**	**…**	**Item I**
User 1	1	0	…	1
User 2	0	0	…	1
…	…	…	…	…
User U	1	1	…	0

The main idea that dominates the CF recommendation system is that the past user-item interactions are adequate to observe other similar users or similar items and the predictions are made based on these estimated proximities [81]. Then, the primary tasks of CF approach are to collect a large amount of information of users' preferences, behaviors or activities and to predict what users will like based on their similarity to other users. The general concept of CF recommendation system is depicted in Figure 2.4.

In various ways, the CF recommendation approaches have been implemented in many different application areas. The Tapestry in Goldberg et al. [28], one of the earliest implementations of CF approach, required explicit feedbacks from the user to filter and browse electronic documents (emails) that arrive in a huge stream in an email system. In Konstan et al. [44], the GroupLens project designed and implemented a CF system for Usenet news to assist users to locate news articles from a massive database. The project took the benefits by building CF into a pre-existing Usenet news system. In Terveen et al. [98], the CF was employed in PHOAKS recommendation system to solve the difficulty of finding related information on the WWW.

Other CF-based recommendation systems, such as Ringo, Bellcore's Video Recommender and Jester, were also developed for music, videos and jokes recommendations, respectively. Ringo in Shardanand et al. [89] is one of the first music recommendation systems that implemented CF to exploit the similarities between the tastes of different users in order to provide personalized music recommendations. Bellcore's in Hill et al. [34] is a video recommender system that interacted with the user via e-mail and provided recommendations to the user in an e-mail reply. Jester in Goldberg et al. [29] is an online joke recommender system based on CF that facilitated the study of social information filtering. Moreover, the Amazon.com proposed an item-item CF algorithm to produce personalized recommendations for each user. Their

algorithm scales to massive data, and the high-quality recommendations are produced in real time [53].

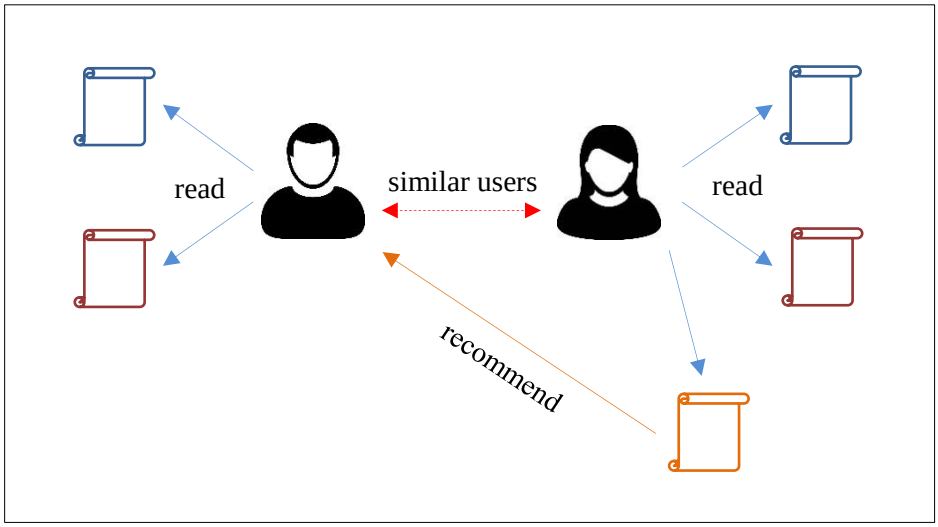


Figure 2.4 Collaborative Filtering Approach

The types of CF methods can be divided into two categories: the memory-based and model-based approaches [41]. The memory-based approach works with values of the user-item interactions directly, assumes no latent model is built, and uses a nearest neighbors search to give rise predictions. As no latent model is assumed, this method has theoretically a low bias but a high variance. The model-based approach assumes a latent model is trained to reconstruct the user-item interactions and makes predictions based on this trained model without having to use the entire data all the time. As a latent model is assumed, this method has theoretically a higher bias but a lower variance.

In memory-based approach, the users' rating data are used to calculate the similarity between pairs of users (user-based CF) or similarity between pairs of items (item-based CF) and to produce a prediction for the user by taking the weighted average of all the ratings. The similarity computation plays a vital role in this approach. Various similarity measures, such as correlation-based, vector-based and probability-based similarity methods, have been developed to calculate similarities between users or items. The memory-based approaches are easy to create and easy to facilitate with new data. Furthermore, this approach has a good adaption with co-rated items and the results are explainable. However, it can have problems when dealing with the large and sparse data.

To continue in deeply, the memory-based CF can be accomplished in two methods: the user-based and item-based CFs [78]. The user-based CF fills a user-item matrix and recommends items based on the users who are most similar to the active user. The item-based CF fills an item-item matrix and recommends items based on similar items.

Alternatively, in model-based approach, the models are developed by applying some different data mining and machine learning algorithms and techniques to produce new predictions for the users. Using these models has hastened to give similar results as in the memory-based approach. The matrix factorization [13] is one of the most popular technique because it uses the dot product to learn the latent characteristics of users and items to give a new prediction. Many other algorithms, such as singular value decomposition, regression, clustering, latent semantic models and neural networks, are widely used in this approach. In addition, the dimensionality reduction techniques can be used as a complementary to reduce the size of matrix from high to lower dimensional space. This reduced matrix can be applied by the algorithms of memory-based approach. Similarity calculation of the reduced matrix is scalable when handling with large datasets.

To summarize these two approaches, the big difference is that the memory-based method uses the entire data all the time to make predictions, while the model-based method uses the data to train a model and then use it to make predictions. Therefore, all data should be in memory for memory-based methods, whilst model-based methods may use smaller data once the model is built so that to improve the scalability and prediction speed of the overall system. But the prediction quality of model-based method is not as accurate as the memory-based one because all the complete data is not used in model-based methods.

As the CF works with past interactions to produce recommendations, it can face with the cold start problem, which means any items are recommended to new users or no new items are recommended to any users [81]. This problem can be dealt with some considerations: random strategy (random items are recommended to new users or new items are recommended to random users), maximum expectation strategy (popular items are recommended to new users or new items are recommended to most active users), or exploratory strategy (a set of various items is recommended to new users or a new item is recommended to a set of various users).

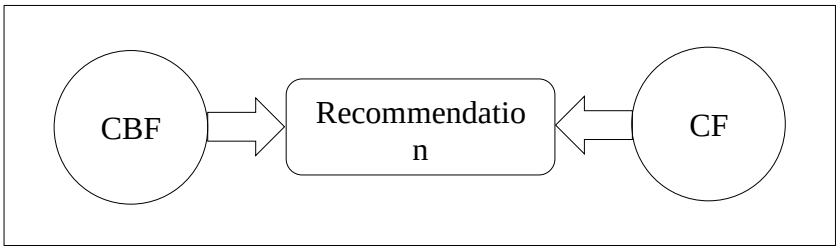
Most of the large-scale applications comprise a greater number of users and items. As a consequence, the user-item ratings matrix can be extremely sparse, and thus, the CF suffers from the sparsity problem. Many researches have been conducted to alleviate this problem. In Sarwar et al. [86], an item-based CF algorithm is proposed to solve the sparsity and scalability issues. More, the dimensionality reduction approach is proposed to reduce the dimension of the user-item ratings matrix by forming clusters of users and items and generate predictions by using these clusters [85]. Also, the statistical and information retrieval techniques can be employed to reduce the dimensionality of the matrix. Necessarily, the dimensionality reduction figures out the sparsity problem by generating a denser user-item ratings matrix that regards only the most relevant users and items.

2.1.2.3 Hybrid Recommendation Systems

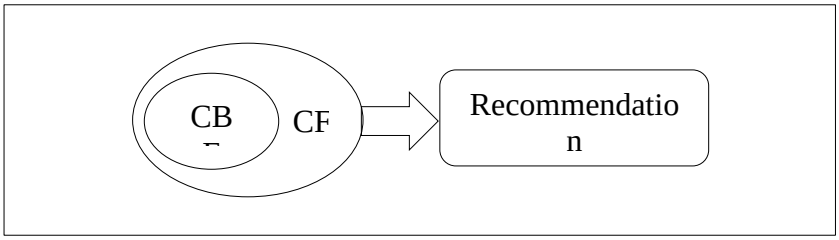
In the literature, many different ways are proposed to create a hybrid recommendation system. As stated earlier, the recommendation systems based on CBF and CF have their own advantages and disadvantages. Therefore, the hybrid approach attempts to combine the CBF and CF approaches to produce the best recommendation results. In this way, the hybrid recommendation systems can avoid the certain problems of each approach and gain the insights from these approaches. Until now, the combination of the CBF and CF is the most researched and applied recommendation system.

Basically, there are four different ways that can be used to aggregate CBF and CF to become a hybrid approach. These ways are illustrated in Figure 2.5 and are arranged as follows:

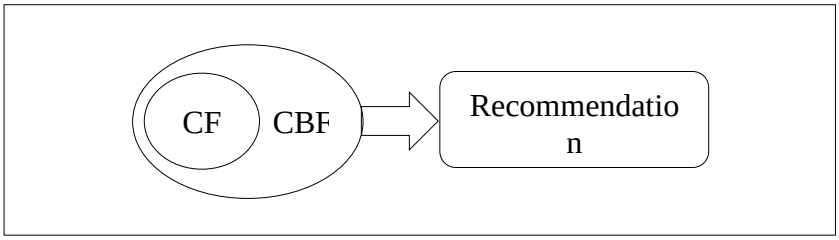
a. Implementing CBF and CF individually and combining their predictions, shown in Figure 2.5 (a),
b. Comprising some CBF characteristics into a CF, shown in Figure 2.5 (b),
c. Comprising some CF characteristics into a CBF, shown in Figure 2.5 (c), and
d. Building a general uniting model that combines both CBF and CF characteristics, shown in Figure 2.5 (d).



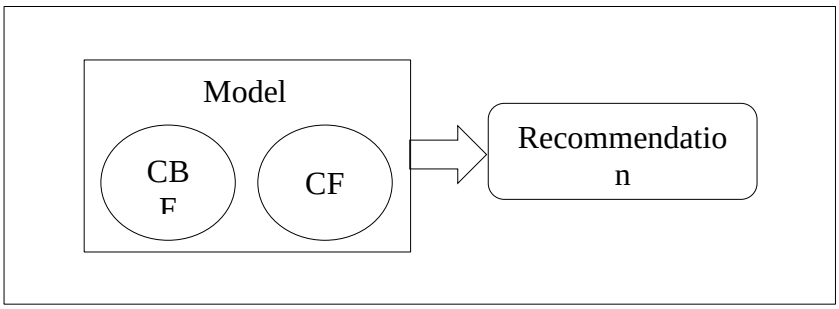
(a)



(b)



(c)



(d)

Figure 2.5 Four Aggregation Methods of Hybrid Approach

The Netflix movie recommendation system is a good example of hybrid recommendation systems [30]. In the researches of Good et al. [31] and Huang et al. [37], the CF is combined with the CBF approach to overcome the cold start problem

which is one of the major limitations of most recommendation systems. Moreover, in Ghazantar et al. [25], the demographic information was added to the hybrid recommendation system to solve the cold start and data sparsity problems. In the papers of Adomavicius et al. [1] and Burke [15], CBF and CF were integrated to build a consolidated hybrid model. To address the new user and average user problems of CBF and CF, a hybrid web recommendation algorithm is implemented in Cunningham et al. [18].

2.2 Paper Recommendation Systems

During the last decades, with the rise of publicly provided online digital libraries and many other such academic services, the research paper recommendation systems have demanded an increasingly place in our lives. Until now, the paper recommendation systems have been developed in more than 200 papers [7]. Indeed, it is still in early stage and thus need to develop more. Throughout these years, paper recommendation systems have been designed and implemented using various approaches for academic researches. Most of the earlier paper recommendation systems, such as in McNee et al. [61] and Strohman et al. [94], are based on the citations as the recommendation of citations for research papers is the popular exploited academic application.

Recently, making use of CBF approach is broadened to academy. Example of CBF approach for scientific papers is provided by Nascimento et al. [69], which is based on the scientific metadata available in public. In Oladapo [72], a CBF recommendation system was proposed for research paper recommendations. The Jaccard similarity coefficient was utilized to calculate the similarity between the user's query and the paper's attributes. Philip et al. [76] proposed a CBF paper recommendation system based on the past ratings of the active user. For representing the papers, the keyword-based vector space model is used. And for similarity calculation, the cosine similarity is used. However, the ratings of similar users are not considered in the system. Also, the same approach is employed in Philip et al. [75] to propose a research paper recommendation system based on the users' query without considering the ratings of the active user.

Likewise, the CF approach is widely applied in research paper domain. In Lee et al. [50], a personalized recommendation system is proposed to recommend academic papers to each researcher. The authors first developed a web crawler, computed similarity by using the bit vector model, and finally a recommendation system is developed using CF methods. Moreover, Madhushree [59] proposed a novel research paper recommendation system with user-based and item-based CF approaches. In the system, the preferences of similar users and the user's item visit history are used for user-based and item-based, respectively. The author does not consider the contents and features of the items or the excuse value of preferences. Furthermore, Haruna et al. [32] proposed a collaborative approach for research paper recommender system. The CF approach is used to infer the hidden associations between research papers by utilizing the publicly available metadata in order to personalize the recommendations regardless of the research field and regardless of the user's expertise.

A hybrid approach to recommend research paper is proposed in Torres et al. [104] by combining CBF and CF approaches. The authors compared different techniques of combinations of CBF and CF in the paper. Also, Gipp et al. [27] introduced a hybrid research paper recommendation system which combines both CBF and CF. The system was combined with author analysis, citation analysis, source analysis, and implicit and explicit ratings to improve the usual keyword-based search. Moreover, in Sharda et al. [88], the CBF and CF were combined with new concepts to create a personalized recommendation system. First, the user's opinions and tastes are profiled and clustered based on the area of interest. Then, the fusion hybrid approach is used for recommending of research papers to the researchers.

Apart from the three approaches described above, there are some other techniques that have been used in paper recommendation systems. The research of Hassan [33] proposed a personalized research paper recommendation using deep learning neural network techniques to improve the quality of recommendations. Sripadh et al. [92] developed a personalized research paper recommender system which proposes a new user model. The model extracts the keywords from the browser history, extracts the concepts and builds the user profile ontology in order to attain improved recommendations.

What is more, Wang et al. [106] proposed a collaborative topic regression (CTR) model which combines CF and probabilistic topic modeling. In their approach,

CF was based on latent factor models which works well for recommending known articles. LDA topic model was used to generalize unseen articles in order to provide a representation of the articles in terms of latent themes discovered from the collection. Li et al. [52] proposed a topic regression matrix factorization model to solve the problems related with recommending scientific papers. The matrix factorization was extended with probabilistic topic model to create a novel proposed model.

2.3 Entropy in Recommendation Systems

The concept of entropy originally comes from thermodynamics. Entropy is a concept in thermodynamics, statistical mechanics and information theory [108]. Information entropy, the concept of entropy used in information theory, was developed by Shannon [87] in 1948. The information entropy is used to measure the uncertainty about an event of a given probability distribution or the spreading degree of distribution. The bigger the entropy, the more scattered the distribution, and the smaller the entropy, the more concentrated the distribution.

Recently, the idea of information entropy has been used in a variety of scientific and academic fields. Mostly, the measures of entropy are useful and relevant in linguistic and computational fields to classify and compare the languages conforming to their information encoding potential and how this potential evolves over time [8]. Moreover, in the field of quantitative linguistics, the measures of entropy are used in natural languages to understand the relationship between words. By bringing entropy to the area of recommendation systems, the dispersion degree of users' preferences or the rating differences between items can be calculated.

The important practical significance of recommendation systems is to generate accurate recommendations with improved efficiency and effectiveness. The different types of recommendation approaches, such as CBF, CF and hybrid approaches have their unique characteristics and limitations. Although CF is the most successfully used approach to produce personalized recommendations, it has drawbacks including the cold start, sparsity and scalability problems.

Piao et al. [77] proposed an entropy-based CF algorithm by computing item entropy and joint entropy between items in order to overcome the cold start and sparsity problems. Chandrashekhar et al. [16] developed a personalized recommendation system

by using entropy-based CF technique that can able to estimate the selective predictability between users. Their approach can deal with the ratings sparsity problem as well as can improve the quality of recommendations.

In addition, the accuracy of CF recommendation systems depends on the similarity computation. When the ratings matrix is sparse, the conventional similarity computation methods cannot effectively measure the similarities of users or items. In such case, information entropy can be used to measure the similarity. The higher the accuracy of recommendation, the lower the value of means absolute error (MAE).

Kwon et al. [47] showed that a reduction in MAE improves the quality of recommendation by computing the weighted entropy of the differences of the rating as similarity measure. In Kwon et al. [48], the authors applied information entropy of the users' rating and incorporated into existing similarity measures to reduce MAE. Wang et al. [107] proposed a new similarity measure by using information entropy to analyse the ratings difference. The authors also proposed an average rating estimation by utilizing Manhattan distance in order to improve the recommendation performance.

However, estimating entropy based on the user rating may not be reliable when the dataset is sparse. Therefore, Lee [49] estimated entropy with respect to each item, instead of each user to deal with the sparse dataset and new user problem. Zhang et al. [115] used information entropy to figure out the cold start and unilateral problems. The authors calculated the information entropy of both users and items in the ratings matrix, and integrate the results to get the rating information entropy. Moreover, an entropy-based computational model was developed by Mehta et al. [63] to propose a personalized Web recommendation system in order to solve the scalability problem of CF.

2.4 Chapter Summary

This chapter provided three main review collections: recommendation systems, paper recommendation systems and entropy in recommendation systems. The intention of this chapter is to highlight the approaches and methods regard with the research area by making a carefully analyzed literature review. Some related works in the field of recommendation system is also covered.

Firstly, the chapter introduced the recommendation systems and explained the main phases and the general model of a typical recommendation system. Then it discussed the types of data that are inputted to recommendation systems and presented the three types of recommendation approaches. It also described the corresponding previous works of each approach. Next, the existing research paper recommendation systems are reviewed and discussed with previous studies. Finally, the information entropy is briefly described and the related researches of entropy in recommendation systems have been completely analyzed.

CHAPTER 3

THEORETICAL BACKGROUND

With the emergence of newly published digitized documents, the researchers have started to focus on analyzing large documents collection for the extraction of latent themes and summarization of this large collection. As more and more digitized information is widely disseminated across several online sources including blogs, websites and digital libraries, it is becoming crucial to collect this information and examine valuable data from this collected information to uncover the latent themes.

3.1 Topic Models

In the age of Internet's technologies, the enormous amount of the digitized data that are generated each day from several sources is merely beyond the traditional processing capability, and thus requires significant processing tools and techniques with a highly scalable fashion to obtain the desired information. Topic models, also called as probabilistic topic models, use statistical algorithms to automatically discover the latent semantic themes from an unstructured collection of text documents and to annotate the collection according to the discovered themes in order to supply an assistance to a better decision making.

In addition, topic models are different from dictionary-based keyword searching methods. It is a popular method of unsupervised learning technique to provide insights that can help in organizing, summarizing and understanding of the large documents collection. In recent years, topic models have seen a upswell of popularity in a variety of domains [54]. They can make work for the particular purpose of feature selection, clustering and information retrieval from unstructured text collections. They have been increasingly applied in many applications of academic, scientific, governmental, commercial and industrial fields. In these areas, topic models serve as more than a classification or clustering technique.

The idea behind a probabilistic topic model starts with the assumption of having a fixed number of unlabelled topics for the documents. Moreover, the topic model assumes that the topics are specified before the documents are generated. Therefore, the topics are contributed in all the documents with different distributions. Then, the

topic model takes a set of documents as input, applies a statistical model to produce a set of topics in terms of multinomial distributions over the words in the vocabulary, and identifies each document with its related topics in terms of a multinomial distribution over the different emerging topics [93]. Generally, the following concepts are used to define a typical topic model:

- A corpus D is an unordered collection of M documents, denoted by $D = \{d_1, d_2, \dots, d_M\}$.
- A document d is an ordered sequence of n words, denoted by $d = w_1, w_2, \dots, w_n$.
- A word w is a basic unit of discrete data and represented by using unit-basis vectors.

The topics produced by topic modeling techniques are the groups of topically-related words with their regarding probability distributions. Intuitively, given that a document is about a particular topic, one would expect particular words that appear in the document more or less frequently. For example, the words "dog" and "bone" will appear more often in documents about dogs, "cat" and "fish" will appear in documents about cats, and "the" and "is" will appear equally in both. A document typically concerns with multiple topics in different proportions, and therefore, in a document that is 10% about cats and 90% about dogs, there would probably be occurred about 9 times more dog-related words than cat-related words. Specifically, a topic model captures this intuition in a mathematical framework, which allows examining and discovering a collection of documents based on the statistics of the words in each, what the topics might be and what each document's balance of topics is.

Among the commonly used topic models, the Latent Dirichlet Allocation (LDA) and Correlated Topic Model (CTM) employ high-level statistical methods, that is, the outputs of the model are yielded with multinomial distributions in terms of word-topic and topic-document probability distributions. The following subsections explore the four methods of topic modeling algorithms, namely LSA, PLSA, LDA and CTM. The comparison between the four topic models based on their related characteristics and limitations are briefly described in Table 3.1. Especially, the LDA and CTM are thoroughly described with two inference methods, such as Gibbs sampling and variational inference, that are used to estimate the parameters of topic models.

Table 3.1 Comparison between Topic Models

Name	Characteristics and Limitations
LSA	Characteristics • Reduces dimensionality of tf-idf using SVD. • No robust statistical background. Limitations • Hard to obtain and determine the number of topics. • Hard to interpret loading values with probability meaning.
PLSA	Characteristics • Generates each word from a single topic, even though various words in one document may be generated from different topics. • Reduces dimensionality to topic level. • Handles polysemy. Limitations • Cannot provide probabilistic model at documents level.
LDA	Characteristics • Needs to manually remove the stopwords. • Cannot make the representation of relationships among topics. Limitations • Not allow to allocate a word to multiple topics. • Unable to model correlations among topics.
CTM	Characteristics • Reveals correlations among topics by using logistic normal distribution. • Allows the occurrences of words in other topics. • Allows the topic graphs. Limitations • Requires lots of computation. • Having lots of general words inside the topics.

Indeed, there are several scenarios when topic modeling can prove useful. Topic models have a wide range of applications such as text categorization in Wu et al. [110] and information filtering Bassiou et al. [6] in the field of information retrieval; similarity measure in the field of recommender systems, for example, to recommend news with a topic structure similar to the news the user has already read in Luostarinen et al. [57]; and uncovering latent themes of texts, for example, to detect and classify the several trends of the online publications in Kim et al. [42].

3.1.1 Latent Semantic Analysis

Latent Semantic Analysis (LSA), proposed by Deerwester et al. [21], is a mathematical technique of analyzing the relationships between a set of documents and the terms that are contained in the documents by producing a set of concepts related to the documents and terms. Moreover, LSA is an information retrieval method to capture the hidden structure of the data. The LSA method is capable of dealing with the problem of multiple terms referring to the same object. An important assumption of LSA is the distributional semantics such that the words that are close in meaning will occur together in similar pieces of text. In the context of its application to information retrieval, it is sometimes called latent semantic indexing (LSI).

The purpose of LSA is to create a vector-based representation of texts and compute the similarity between texts by using the vector representation to pick the most related words. LSA utilizes the singular value decomposition (SVD) proposed in Forsythe et al. [24] in order to provide a solution for high-dimensional large datasets by giving lower-dimensional representation of original documents and to significantly reduce the term-document matrix. Precisely, the term-document matrix is constructed from a high-dimensional dataset. It is an overly sparse matrix that describes the occurrences of terms in documents, where, the rows represent terms and the columns represent documents. Then, SVD is used to eliminate the noisy terms in order to reduce the dimension of the matrix while preserving the similarity structure among the documents.

However, the resulted lower-dimensional matrix might be difficult to interpret and match to the observed data. Because the probability model of LSA assumes that the terms and documents compose a joint Gaussian model while a Poisson distribution has

been observed. Therefore, a probabilistic latent semantic analysis model (PLSA) based on multinomial model is created to correct the statistically unstable aspects of LSA and to yield better results than LSA.

3.1.2 Probabilistic Latent Semantic Analysis

Probabilistic Latent Semantic Analysis (PLSA), proposed by Hofmann [36], is a two-level hierarchical Bayesian model where each word is generated from a single topic and each document is reduced to a probability distribution of a fixed set of topics. Following with this concept, PLSA has improvements in producing promising results over the LSA. The PLSA, also known as probabilistic latent semantic indexing (PLSI) in information retrieval, is a generative statistical model which is based on the aspect model to analyse the cooccurrence of data. The representation of two-level PLSA generative model is shown in Figure 3.1.

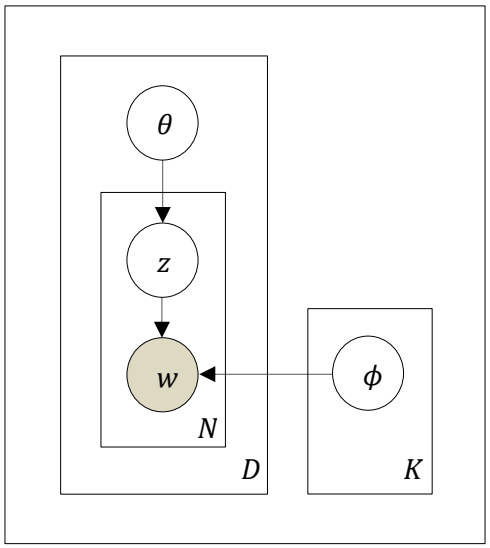


Figure 3.1 Graphical Model of PLSA

The formal definition of PLSA generative process that generates documents is described in the following procedure:

For each of the D documents d,

For each of the N_d words $w_{n,d}$,

Choose $z_{n,d} \sim Multinomial(\theta)$

Choose $w_{n,d} \sim Multinomial(\phi_{z_{n,d}})$.

The notations which are required to describe the concept of PLSA topic model are shown in Table 3.2.

Table 3.2 Notations for PLSA Topic Model

Symbol	Description
D	Number of documents in the collection
N	Number of words in the collection
K	Number of latent topics
θ	Topic distribution for the document d
ϕ	Word distribution for the topic k
$z_{n,d}$	Topic assignment of the word $w_{n,d}$
$w_{n,d}$	Word n in document d

PLSA can be viewed as a matrix factorization technique as illustrated in Figure 3.2. By training the PLSA model, the words-documents matrix is decomposed into words-topics and topics-documents matrices. PLSA assumes that, for each document d in the documents collection, each word w is generated from the latent topic z with the probability of $p(w \mid z)$. Again, each topic z is generated from the topic distribution θ with the probability of $p(z \mid \theta)$. Then, the conditionally independent multinomial distribution is used to model the probability of occurrence of a word in a document as follows,

$$p(w, d) = p(d)\, p(w \mid d) \tag{3.1}$$

where

$$p(w \mid d) = \sum_{z \in Z} p(w \mid z)\, p(z \mid \theta) \tag{3.2}$$

However, as PLSA is a two-level probabilistic model, no probabilistic generative model at a document level to generate topic proportions is not considered in PLSA. With this fact, PLSA suffers from an overfitting problem when new documents are introduced to the trained PLSA model. Therefore, latent Dirichlet allocation (LDA) model is created to overcome this problem by building a three-level hierarchical Bayesian model at a corpus level.

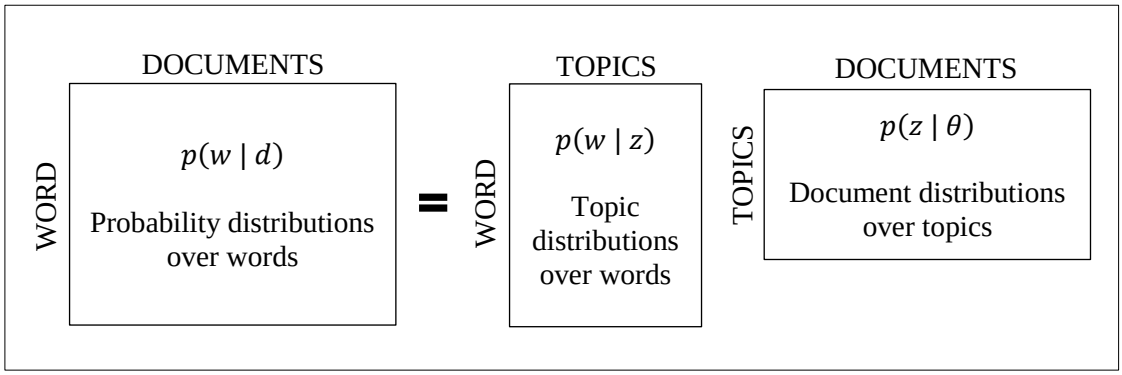


Figure 3.2 Probabilistic Latent Semantic Analysis

3.1.3 Latent Dirichlet Allocation

Latent Dirichlet allocation (LDA) proposed by Blei et al. [12], the most widely used probabilistic topic model, employs statistical methods to find the hidden themes that can be used in organizing and understanding of the large datasets. The LDA prevents the over-fitting problem that occurs when the model parameters do not grow linearly with the dataset's size. Moreover, LDA is also generalization of PLSA model which is represented by bags-of-words.

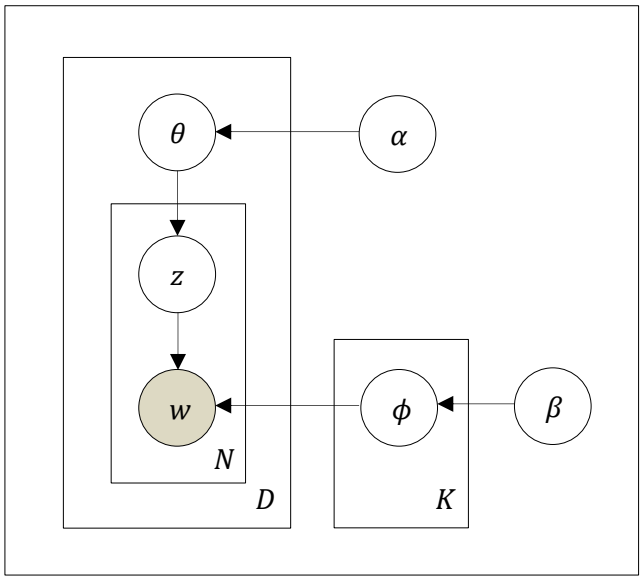


Figure 3.3 Graphical Model of LDA

In LDA, each document in a collection of documents is modeled as a multinomial distribution over a set of finite topics, where each topic is characterized as a multinomial distribution over a set of unique words. LDA is a high-level generative

model which utilizes the Dirichlet distribution to estimate the intractable hidden topic structure of the model. In other words, the Dirichlet distribution describes the prior distribution of the latent topic structure. The basic idea behind LDA is that each word of a document from an input collection is generated by first selecting a topic in the document it corresponds to and then finding out the probability of the word within that topic.

The three-level LDA is represented as a generative model in Figure 3.3. In the figure, the outer rectangular plate represents the total documents in the collection, while the inner plate represents the total words in the collection. The Table 3.3 shows the notations which are used to describe the generative model of the LDA topic model.

Table 3.3 Notations for LDA Topic Model

Symbol	Description
D	Number of documents in the collection
N	Number of words in the collection
K	Number of latent topics
V	Number of words in the vocabulary
α	Dirichlet parameter over θ
β	Dirichlet parameter over ϕ
θ	Topic distribution for the document d
ϕ	Word distribution for the topic k
$z_{n,d}$	Topic assignment of the word $w_{n,d}$
$w_{n,d}$	Word n in document d

Intuitively, the generative process of the LDA model over a collection of documents is described as follows:

1. For each of the K topics ϕ_k,
   Choose $\phi_k \sim Dirichlet(\beta)$
2. For each of the D documents d,
   a. Choose $N_d \sim Poisson(\xi)$
   b. Choose $\theta_k \sim Dirichlet(\alpha)$
   c. For each of the N_d words $w_{n,d}$,
      i. Choose $z_{n,d} \sim Multinomial(\theta)$
      ii. Choose $w_{n,d} \sim Multinomial(\phi_{z_{n,d}})$.

As shown in the graphical model representation of LDA in Figure 3.3, a collection of documents consists of D documents and each document d contains a sequence of N words. The distribution of the topic ϕ over the vocabulary assumes that the topic z is fixed. The topic mixture θ at the document level is multinomially distributed over all the topics. The posterior distribution of the latent variables given a document is described in the following equation,

$$p(\theta, z \mid w, \beta, \alpha) = \frac{p(\theta, z, w \mid \beta, \alpha)}{p(w \mid \beta, \alpha)} \tag{3.3}$$

where, given the corpus-level parameters α and β, the joint distribution of a topic mixture θ, a topic z, and a word w is given by,

$$p(\theta, z, w \mid \beta, \alpha) = p(\theta \mid \alpha) \prod_{n=1}^{N} p(z_n \mid \theta)\, p(w_n \mid z_n, \beta) \tag{3.4}$$

However, a major limitation of LDA is the incapability of directly modeling and capturing the correlations between topics because LDA infers the topic proportions by using a Dirichlet distribution. The distribution assumes that the presence of one topic is not correlated with the presence of another because of its independence structure. However, in most of the realistic applications, the latent topics can have relations between them and correlate with each other. Therefore, the correlated topic model (CTM) is proposed to overcome this difficulty of LDA.

3.1.4 Correlated Topic Model

Probabilistic topic models reveal the underlying thematic structures in a collection of documents by extracting topics. With these extracted topics, the whole document collection can be categorized and summarized without human annotation effort. The Correlated Topic Model (CTM), which was first proposed by Blei et al. [11], figures out the limitation of LDA by replacing the Dirichlet distribution with the logistic normal distribution to disclose the correlations of latent topics in order to produce more coherent and reasonable results. The covariance matrix of the logistic normal distribution captures the information about the latent topics' correlations that can able to improve the predictive power of the CTM.

The CTM is also a generative probabilistic model as LDA to find the patterns of words in documents, to discover the hidden semantic themes of a collection of

documents and to describe how these themes are distributed over individual texts. CTM makes a better fit to the documents collection due to more assumptions made, and thus can provide more expressivity, such as visualization and exploration, than LDA. By means of CTM, every document can be viewed as a combination of multiple topics with different proportions. The CTM, one of the statistical topic models, has been applied in many domains, such as images in Tao et al. [96] and Xu et al. [112], web services in Aznag et al. [3], computer vision in Sang et al. [84] and text analysis in Hoang et al. [35] and Xun et al. [113].

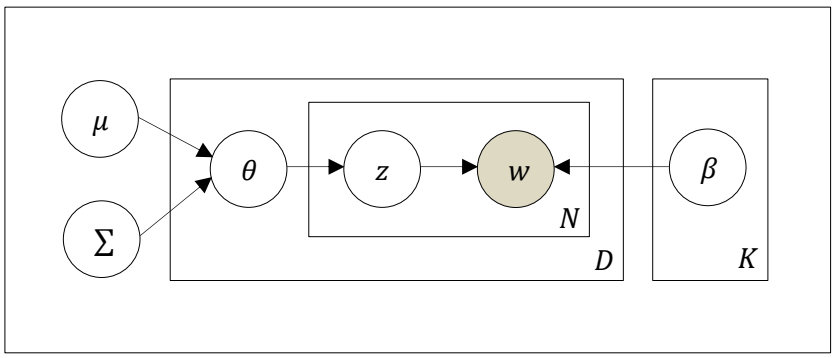


Figure 3.4 Graphical Model of CTM

Figure 3.4 illustrates the graphical model representation of CTM. A graphical model provides a framework that graphically represents the conditional dependencies or the interactions among the corresponding random variables involved in a statistical model of a probabilistic system. In the Figure 3.4, the unshaded nodes represent the latent variables of CTM and the arrows show the conditional dependencies among the variables. Also, the rectangles denote the replicated structure of the model, and only the shaded node w_n is observed.

CTM assumes that the words of each document arise from a mixture over latent topics, each of which is modeled as a distribution over the vocabulary. The core concept of CTM is the use of logistic normal distribution. CTM exhibits the pairwise correlations between latent topics through the covariance matrix of the logistic normal distribution. As a consequence, the logistic normal adds complexity to the variational inference and parameter estimation process. The perception of CTM is that a document consists of many topics with different proportions and different topics have different

distributions over the vocabulary. The important notations which are used in CTM graphical model are summarized in Table 3.4.

Table 3.4 Notations for CTM Topic Model

Symbol	Description
D	Number of documents in the collection
N	Number of words in the collection
K	Number of latent topics
V	Number of words in the vocabulary
μ, Σ	Logistic normal parameters over θ
β	Word distribution for the topic k
θ	Topic proportion for the document d
η	Topic distribution for the document d
$z_{n,d}$	Topic assignment of the word $w_{n,d}$
$w_{n,d}$	Word n in document d

Given a collection of documents D, a K-dimensional Normal distribution of mean and covariance matrix $N(\mu, \Sigma)$ and a number of topics K, CTM assumes that the documents are generated according to the following generative process:

1. For each topic k,
   Choose a distribution over the vocabulary $\beta_k \sim N(\mu, \Sigma)$
2. For each document d,
   a. Choose a distribution over the topics $\eta_d \sim N(\mu, \Sigma)$
   b. For each word n,
      i. Choose a topic assignment $z_{n,d}$ from $Mult(f(\eta_d))$
      ii. Choose a word $w_{n,d}$ from $Mult(\beta_{z_{n,d}})$.

Topic proportion θ for each document is obtained from the logistic normal transformation,

$$\theta = f(\eta) = \frac{\exp\{\eta\}}{\sum_i \exp\{\eta_i\}} \tag{3.5}$$

In order to extract the latent topics, the CTM may have a challenge in calculating the posterior distribution of topics over observed words. The word distribution per topic β_k and topic distribution per document η_d of CTM are difficult to compute directly, but

different inference algorithms have been proposed to figure out the model parameters estimation, including Gibbs Sampling and variational Expectation-Maximization (variational EM).

Gibbs sampling is a Markov Chain Monte Carlo algorithm which draws samples from probability distributions. Variational EM algorithm of Blei et al. [10] relies in computing the maximum likelihood estimates of parameters. To learn the parameters of CTM, a variational EM algorithm is used for inferencing. The two procedures in variational EM consists of posterior inference and parameter estimation.

Given an observed document w and model parameters $\{\beta, \mu, \Sigma\}$, the posterior distribution of the latent variables $p(\eta, z \mid w, \beta, \mu, \Sigma)$ is intractable to compute. Jensen's inequality is used to bound the log probability of a document,

$$\text{Log}\, p(w_N \mid \mu, \Sigma, \beta) \geq E_q[\log p(\eta \mid \mu, \Sigma)] + \sum_{n=1}^{N} E_q[\log p(z_n \mid \eta)] + \sum_{n=1}^{N} E_q[\log p(w_n \mid z_n, \beta)] + H(q) \quad (3.6)$$

where $H(q)$ denotes the entropy of variational distribution. The posterior inference adds a set of variational parameters to obtain the approximate lower-bound on likelihood of each document. The variational distribution is set to,

$$q(\eta, z \mid \lambda, \nu^2, \phi) = \prod_{i=1}^{K} q(\eta_i \mid \lambda_i, \nu_i^2) \prod_{n=1}^{N} q(z_n \mid \phi_n) \quad (3.7)$$

where (λ, ν^2) is variational mean and covariance of normal distribution, ϕ is a variational multinomial distribution.

Given a collection of documents, the parameter estimation maximizes the likelihood of the whole documents collection by using a variational EM. In the E-step, a variational inference for each document is done to maximize the bound with respect to the variational parameters $\{\lambda, \nu^2$ and $\phi\}$. In the M-step, the bound is maximized with respect to the model parameters $\{\mu, \Sigma$ and $\beta\}$. The E-step and M-step are repeated until the bound on the likelihood converges. The detailed explanations of posterior inference and parameter estimation can be found in Blei et al. [10].

3.2 Apache Hadoop

Hadoop is an open-source framework developed by the Apache Software Foundation not only for storing, processing, and analyzing of big data but also for providing reliable, scalable and distributed computing [99]. It has become a popular platform in the IT industry because of its ability to store, process, analyze and extract appropriate information from vast amounts of data. Moreover, Hadoop is a batch processing system across a cluster of nodes which is needed to deal with large unstructured datasets. Hadoop is capable of being a runtime environment for MapReduce programs and its distributed filesystem HDFS.

Apache Hadoop framework mainly consists of two functionalities:

1. Hadoop Distributed File System (HDFS): HDFS is a scalable and distributed file system written in Java for the Hadoop framework. HDFS takes care of the primary data storage part of Hadoop and provides distributed data processing capabilities to Hadoop.
2. Hadoop MapReduce: MapReduce is a computational model designed in Java programming language for the processing of large data sets in a distributed fashion across a Hadoop cluster. HDFS when coupled with MapReduce is capable of handling and processing big data.

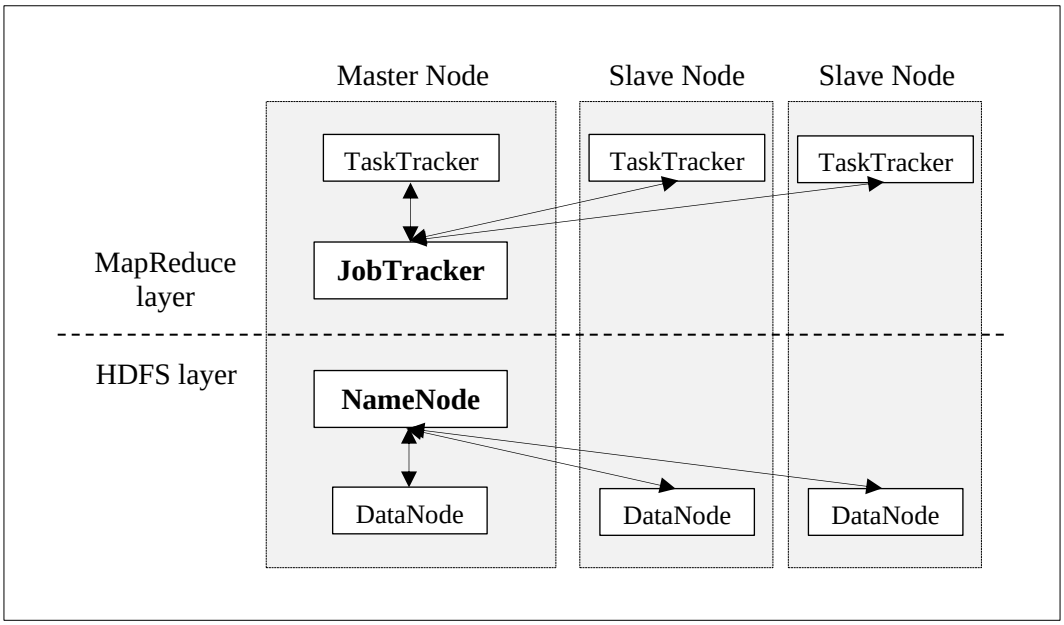


Figure 3.5 High Level Architecture of Hadoop

The Apache Hadoop follows a Master/Slave architecture designed for data processing and data storage using MapReduce and HDFS respectively. The architecture of Hadoop framework is illustrated in Figure 3.5. For the master node, the NameNode is for data storage and the JobTracker is for parallel processing of data. The main idea of Hadoop framework is to decompose a job into several identical tasks that can be executed on the DataNode for concurrent data processing. Every slave node has a DataNode and a TaskTracker that synchronizes the processes with the NameNode and JobTracker respectively [73].

3.2.1 Hadoop Distributed File System

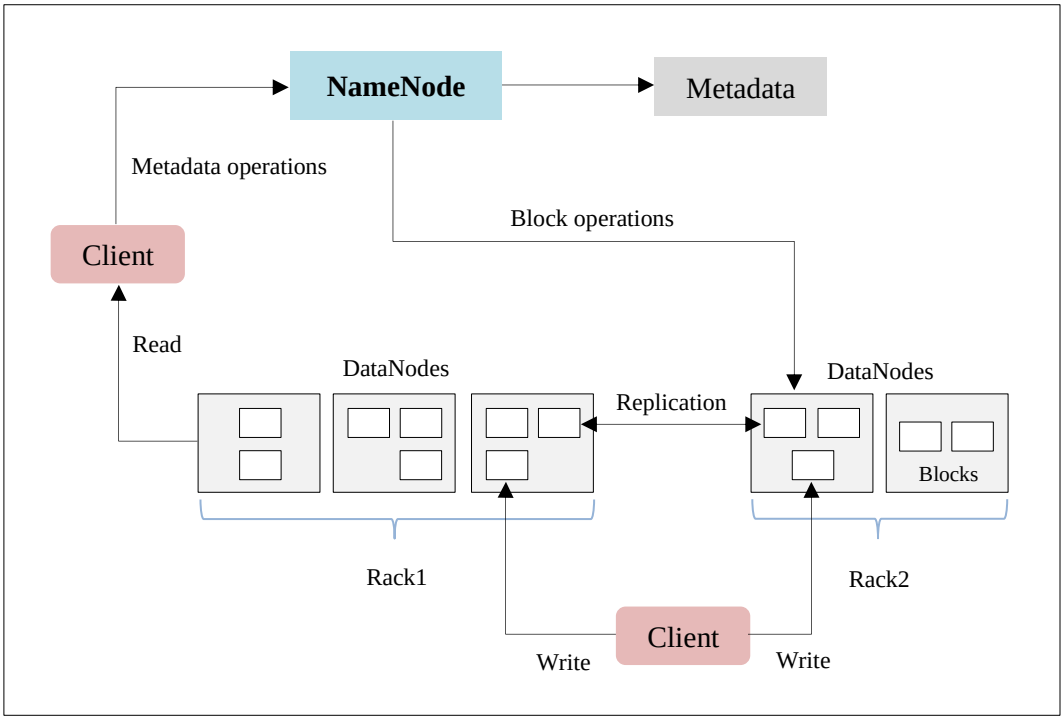


Figure 3.6 Architecture of HDFS

Hadoop Distributed File System (HDFS) is the default storage layer for Apache Hadoop as HDFS is capable of storing a large number of huge datasets. HDFS is best known for its fault tolerance and high throughput by providing data access in parallel. The HDFS architecture comprises two critical components: NameNode and DataNodes. The NameNode stores the file system metadata which includes the number of blocks, block locations, replicas and other information. The DataNode stores the

application data which is the data blocks of a file. Figure 3.6 is the explanation of the HDFS architecture [19].

In Hadoop, the NameNode is the centerpiece of HDFS and responsible for managing and assigning tasks to the DataNodes and monitoring the health status of the Slave Nodes. In addition, the NameNode performs the metadata operations such as opening, closing and renaming files and directories. The DataNode is responsible for serving the client read/write requests. The DataNode executes the creation, deletion and replication of data blocks depending on the NameNode's instructions. The DataNode periodically send heartbeats to the NameNode about its health status and the task status. If the DataNode fails to send heartbeats to the NameNode, then the NameNode considers the Slave Node to be dead and chooses the new available DataNode to reassign the task.

Since HDFS is the data storage component of Hadoop, all data stored on Hadoop is stored in a distributed manner by creating multiple replicas of data blocks depending on the replication factor. Internally, a file on HDFS is broken down into smaller units and stored on different DataNodes in the cluster. These multiple fixed-sized chunks are called blocks with a default size of 64MB in Hadoop 1.x (128MB in Hadoop 2.x). The size of a block can be configured and extended up to 256 MB based on the requirements.

3.2.2 Hadoop MapReduce

MapReduce is a programming paradigm that enables the massive data processing using dispersed and parallel algorithms on a Hadoop cluster. As the processing component, MapReduce is the heart of Hadoop which enables to perform various operations over the big data. It is a software framework for writing applications that process large datasets in parallel across hundreds or thousands of nodes on the Hadoop cluster [9].

MapReduce is a combination of two separate phases that Hadoop programs perform, namely:

- Map phase – deals with splitting and mapping of data.
- Reduce phase – deals with shuffling and reducing of data.

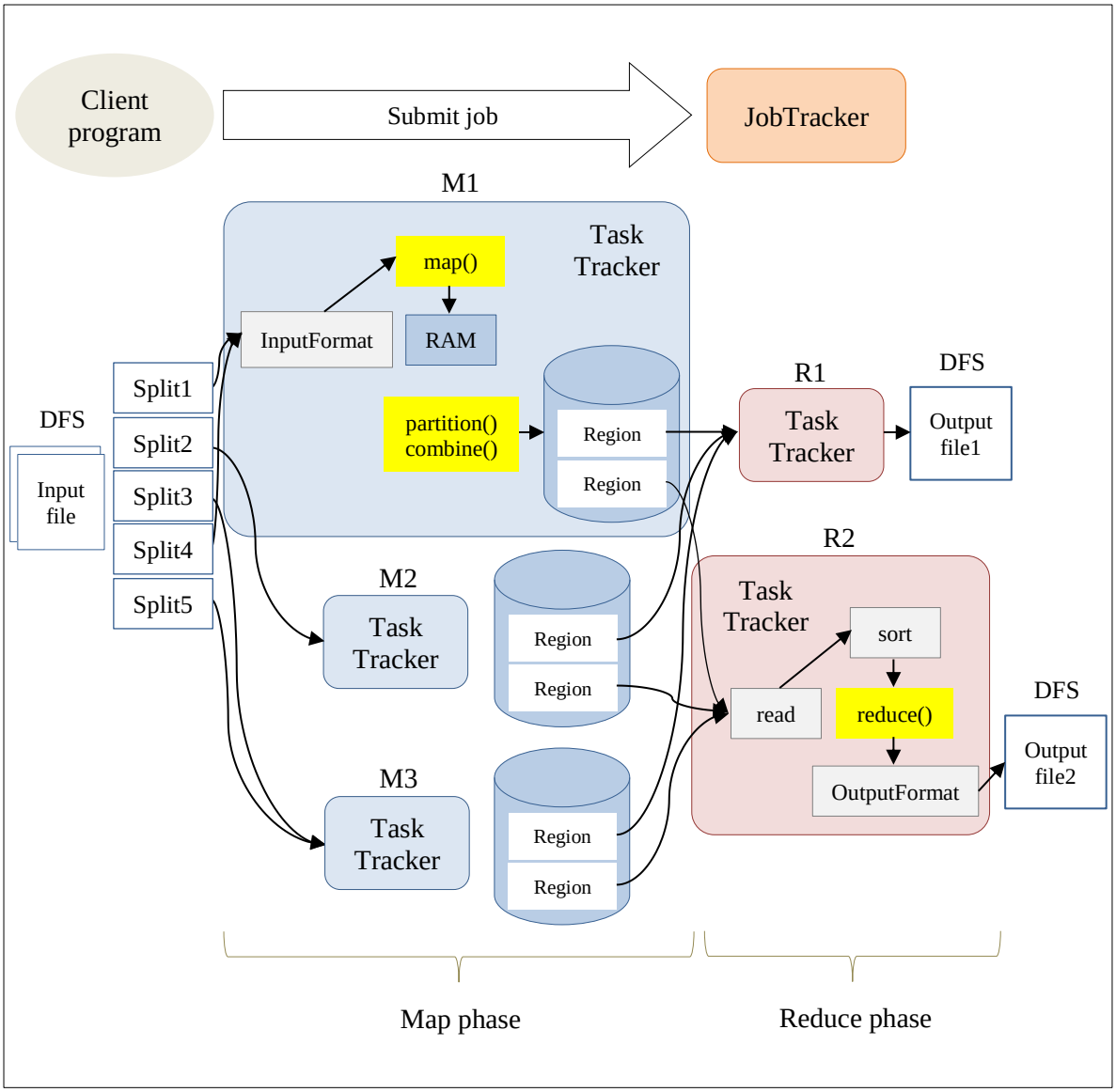


Figure 3.7 Architecture of MapReduce

The basic principle of operation behind MapReduce framework is that the Map function receives a key-value pair as input and generates intermediate key-value pairs as output. The Reduce function merges all the intermediate key-value pairs associated with the same key and then generates the final output. Semantically, the data is distributed in the Map phase and the Reduce phase actually performs the computation. Figure 3.7 depicts the execution process of MapReduce framework.

In Map phase, the MapReduce framework takes the input data, divides it into chunks and feeds each data element to the mapper. In Reduce phase, the reducer processes all the intermediate outputs from the mapper and arrives at the final output of

the framework [73]. The Reduce phase is always performed after the Map phase as the name ‘MapReduce’ implies. The underlying MapReduce library automatically provides parallelism, fault tolerance, load balancing and data distribution.

Some of the various components of MapReduce architecture consist of:

- *Client*: A program or Application Programming Interface (API) that submits jobs to the MapReduce framework.
- *Job*: The actual work that needs to be executed or processed.
- *Task*: A piece of the actual work that needs to be executed or processed. A MapReduce job comprises many small tasks that need to be executed.
- *Job Tracker*: The Job Tracker assigns the jobs to the task trackers and tracks the jobs that are assigned to all the task trackers.
- *Task Tracker*: The Task Tracker reports the status of tasks to the Job Tracker.

The complete execution process of Map and Reduce phases is controlled by the JobTracker and TaskTracker daemons. For every job submitted for execution, the JobTracker resides on NameNode and multiple TaskTrackers reside on DataNode. The following actions take place when a MapReduce function is called:

- The JobTracker first determines the number of splits from the input, and select some TaskTrackers to send the task requests.
- Each TaskTracker start the Map phase by extracting the input data from the splits.
- For each record parsed by the InputFormat, it invokes the user provided map() function, which emits a number of key-value pairs in the memory buffer.
- A periodic wakeup process will sort the memory buffer into different reducer node by invoking the combine() function.
- When the Map phase completes, the TaskTracker notifies the JobTracker.
- When all the TaskTrackers are done, the JobTracker notifies the selected TaskTrackers for the Reduce phase.
- Each TaskTracker read the region files remotely. It sorts the key-value pairs and for each key, it invokes the reduce() function, which collects the key-aggregatedValue into the output file.
- After the Map and Reduce phases complete, the JobTracker unblocks the client program.

The following Figure 3.8 illustrates a word count example of MapReduce.

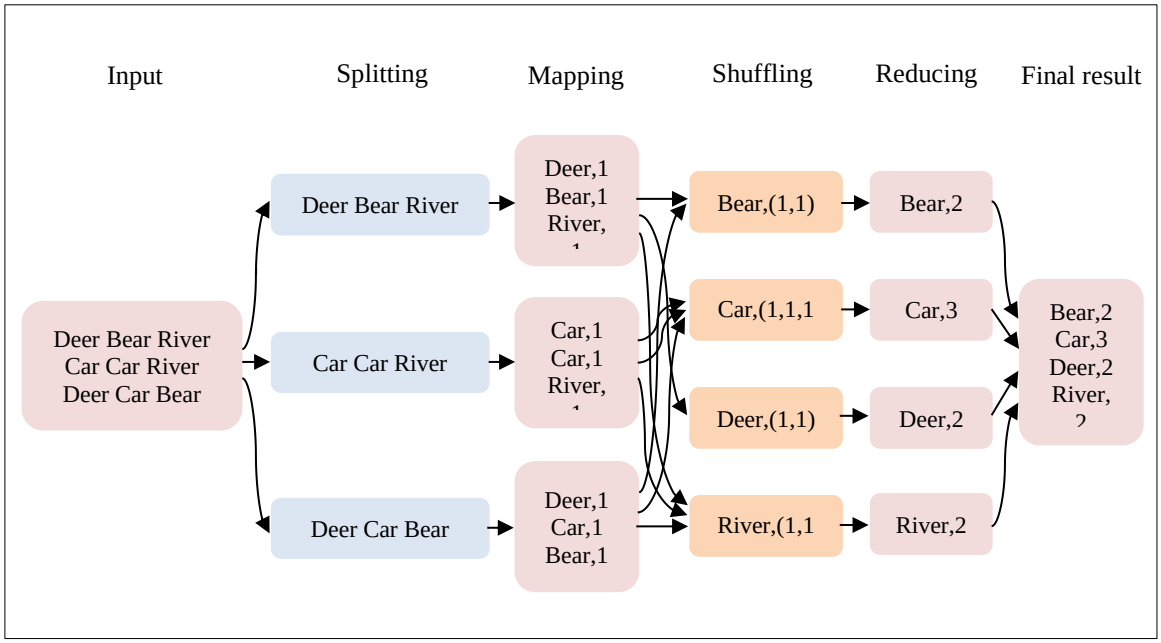


Figure 3.8 Canonical Example of a MapReduce Job

3.3 Chapter Summary

This chapter introduced the comprehensive overview of topic models for discovering a set of topics that can represent a collection of text documents, and then discussed the notations and assumptions of topic models. The aim of this chapter is to highlight the characteristics of different types of topic models regarding with the proposed methodology. The chapter also presented the background of four major paradigms of topic models, including LSA, PLSA, LDA and CTM, with their general ideas, limitations and detailed concepts and explanations. With the use of topic models, the human-interpretable topics from a collection of documents can be automatically extracted. Furthermore, an overview of the architecture of Apache Hadoop is presented to provide an understanding of Hadoop. Two main components of Hadoop ecosystem: Hadoop Distributed File System (HDFS) and Hadoop MapReduce, are addressed with illustrations to gain insights on how MapReduce can be used in practical applications.

CHAPTER 4

THE ARCHITECTURE OF THE PROPOSED SYSTEM

The massive growth in the extent of information over the Web has demanding the efforts and skills of the users to search for the information they wished for. To come through this situation, the recommendation systems emerged as a revolutionary insight in the academic domain to provide users with the admissible and useful information. In this chapter, a research paper recommendation system for researchers is proposed in order to have scalability and gain accuracy when working with big data in the recommendation systems.

To make it work efficiently with the increased number of research papers in the context of big data challenge in recommendation system domain, the Correlated Topic Model (CTM) implemented on MapReduce framework is exploited in the proposed research paper recommendation system. The purpose is to deliver the fast and relevant search results and generate recommendations at scale. For this reason, the proposed system implements the processing of MapReduce CTM in a Hadoop cluster.

In addition, as different users may have different requirements, the proposed system accepts the specific keywords from the user. For the purpose of discovering more similarity relationship between the keywords and the documents, the proposed system applies entropy to reveal the predictability relationship before the recommendations are generated to the user.

In a nutshell, this chapter will go over presenting with multiple sections to introduce the proposed paper recommendation system. The proposed system works in four main stages. The workflow of the proposed system is illustrated in Figure 4.1.

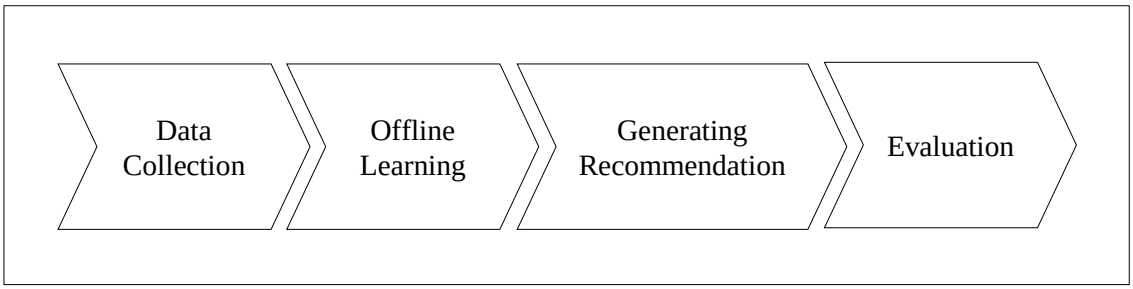


Figure 4.1 Proposed Recommendation System Workflow

- **Data Collection Stage**: The first stage is the collection of publicly available digitalized documents from the digital libraries by employing a web crawler.
- **Offline Learning Stage**: The second stage is the processing of offline learning tasks which in turn involves three phases: pre-processing of data to clean the crawled datasets, applying CTM model to extract the topics that can represent the documents and calculating similarity between the extracted topics. Primarily, training the CTM with variational EM algorithm with MapReduce implementation is the main task of this stage.
- **Generating Recommendation Stage**: The third stage is the generation and presentation of recommendations which involves three phases: keywords processing, documents extraction and predictability computation by using entropy. This stage is executed when the user provided to the searched keywords to the recommendation system.
- **Evaluation Stage**: Finally, the last stage is the evaluation of the proposed paper recommendation system based on the calculation of topics coherence measures and the calculation of precision and recall measures. The detailed evaluations are discussed in Chapter 6.

Figure 4.2 illustrates the overall architecture of the proposed paper recommendation system that mainly depicts the several phases involved in offline learning and generating recommendations stages. The MapReduce CTM is incorporated in the offline learning stage and the entropy is applied in generating recommendation stage. In the initial data collection stage, the datasets are gathered from the multidisciplinary digital libraries and are stored in Hadoop Distributed File System (HDFS) to accomplish the processing in the Hadoop cluster. After this stage, the three phases involved in the offline learning stage are initiated.

During the offline learning stage, the collected raw text documents are pre-processed and the latent topics that can represent the whole documents collection are extracted by using the MapReduce CTM model. In this phase, the model is trained in offline mode using Hadoop MapReduce framework and HDFS of the Hadoop ecosystem. Then, the similarity between the extracted topics are computed to provide the semantically related documents.

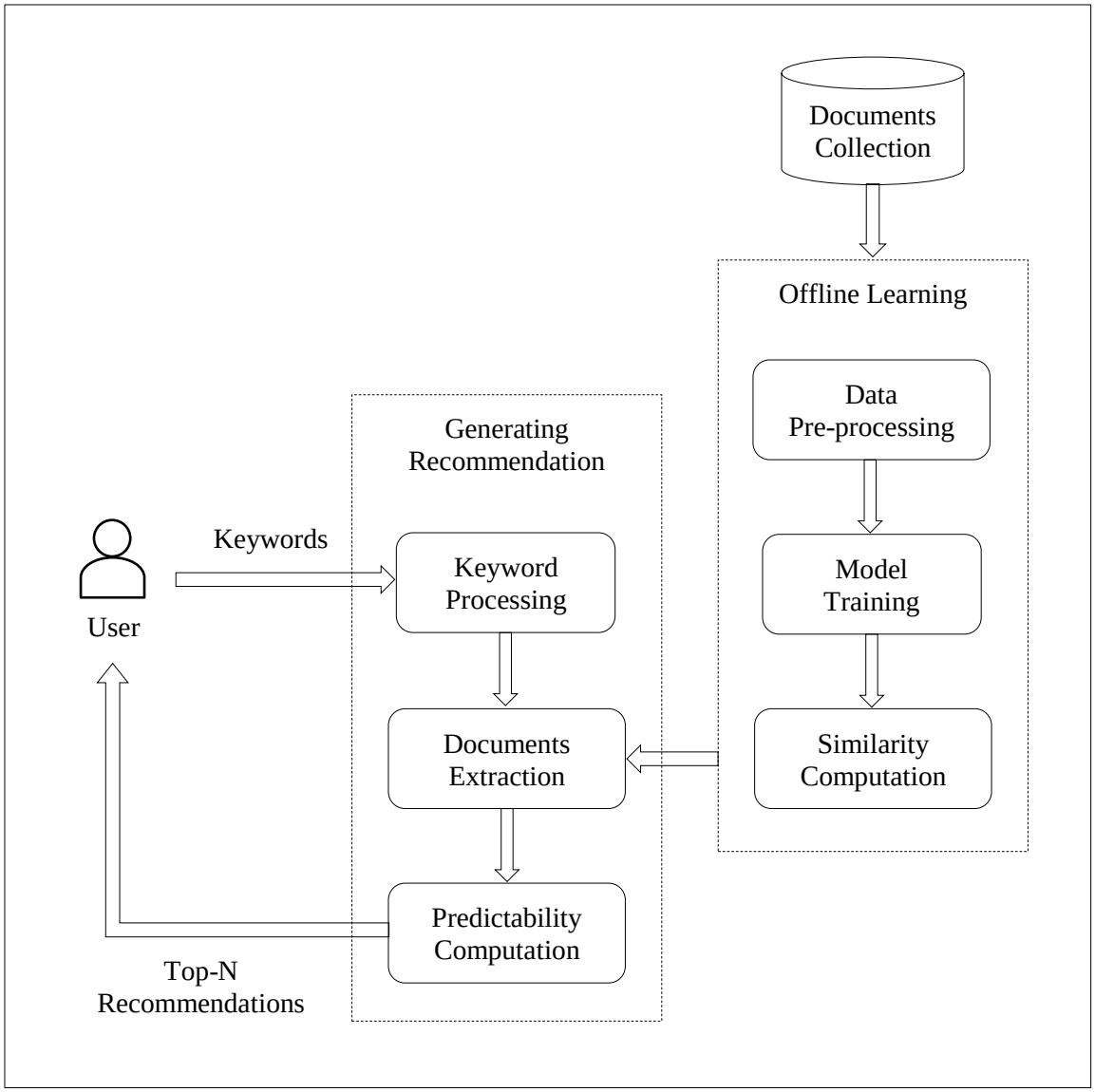


Figure 4.2 Architecture of Proposed System

After the offline learning stage is accomplished, the proposed system is partially prepared for the recommendations. When the user enters the query keywords to the paper recommendation system, the system processes the keywords to remove the unnecessary words that do not have the taste of the user. Then, the documents are extracted by matching the keywords to the top words of the topics. Again, the extracted documents are filtered to maintain the more relevant ones by computing the predictability. At last, the final recommendations are displayed to the active user. For this purpose, the interactive Graphical User Interface (GUI) is employed for helping the user to interact with the paper recommendation system.

4.1 Data Collection Stage

To gather the digital documents from two multidisciplinary digital libraries, namely, CiteSeerX and PLOS ONE, a web crawler is employed for this task. A web crawler, sometimes called a web spider, is an automated script that starts crawling from the initial seed URL, indexes the contents of the fetched webpages, extracts the targeted data and transfers that data into a structured format. In this way, documents from each digital library are filtered and assembled to build a data collection. CiteSeerX (https://citeseerx.ist.psu.edu/index) is one of the largest and widely used research paper repository which focuses primarily in computer and information science [101]. PLOS ONE (https://journals.plos.org/plosone/) covers primary researches from any discipline within science and medicine [79].

To browse and collect the published research documents from each multidisciplinary library, a web crawler is developed by using Java implementation. The crawling process starts scanning the pre-defined URL of each digital library and goes from one link to another to find and scrape the documents in Portable Document Format (PDF) ending with .pdf extension. The textual contents of each crawled PDF document are extracted and converted into plain Text files ending with .txt extension by using the Apache PDFBox library. The PDFBox [100] is an open-source Java tool that allows to create, manipulate and extract contents from PDF documents. The extracted text data are uploaded and stored in HDFS to perform further offline learning phases in the Hadoop cluster. Figure 4.3 depicts the steps of data collection stage.

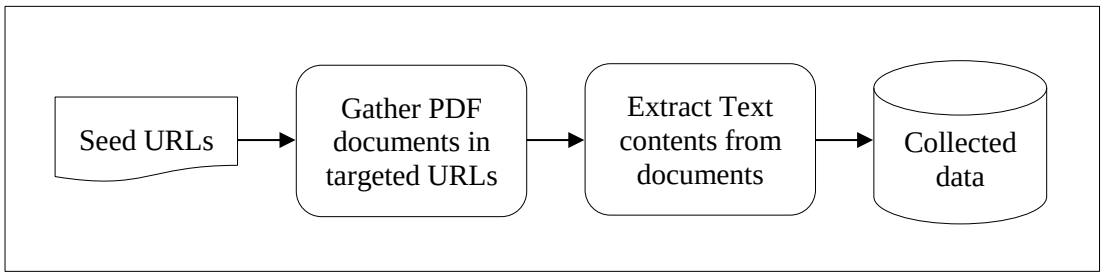


Figure 4.3 Data Collection Steps

4.2 Offline Learning Stage

The offline learning stage plays an important role of the proposed paper recommendation system. This stage is facilitated in offline prior to the generating

recommendation stage in order to deliver the recommendations to the users in a short response time.

Firstly, the collected data are pre-processed to remove unnecessary contents in order to transform the input text data into a more digestible form so that the MapReduce CTM can perform better. Then, the topics are extracted from the pre-processed datasets by using MapReduce CTM model with the variational EM algorithm to discover the semantic themes of the datasets. The purpose is to integrate the extracted topics into the recommendation generation stage in order to recommend the latent and unseen research documents to ensure a good performance of the paper recommendation system.

After the model is built, the similarity computation by using a distance measure and the predictability computation by using entropy are carried out. Here, the intention of using entropy is to improve the accuracy and quality of recommendations and at the same time to filter and rank the recommended documents. Implementing the MapReduce CTM algorithm with variational EM algorithm using the MapReduce functionality of the Hadoop processing platform is the backbone of this stage.

6.3.1 Data Pre-processing

The data pre-processing phase represents a critical role prior to the topic extraction of the model training phase. The purpose of data pre-processing is to structure the unstructured data. The pre-processing of data is performed in order to filter and clean the data which are much more useful, to extract the vocabulary, to yield better results and to speed up the rest processing phases of the offline learning stage.

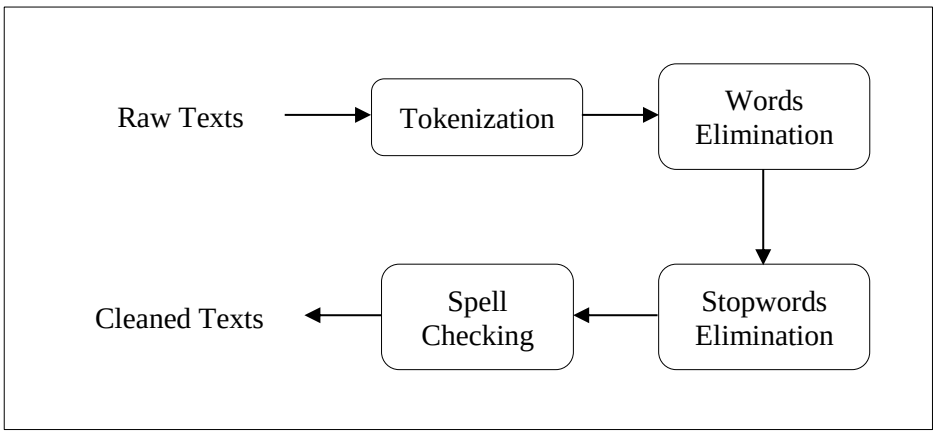


Figure 4.4 Data Pre-processing Steps

A series of steps under the data pre-processing phase are illustrated in Figure 4.4. In this data pre-processing phase, the Map and Reduce functionalities of the Hadoop are used for the four pre-processing tasks. The input to this phase is the raw converted full-text documents which are stored in HDFS.

4.2.1.1 Tokenization

From the raw text documents stored in HDFS, tokenization is the process of breaking up the large chunks of given text documents into smaller chunks. To be specific, the input text documents are split into smaller pieces called sentences, and these sentences are again separated into a list of single words, known as tokens. For example, paragraphs can be tokenized into sentences and sentences can be tokenized into words.

4.2.1.2 Words Elimination

During words elimination step, the numbers, punctuations, irrelevant words, short and long words with less than four and greater than twenty characters are eliminated in Map function and the occurrences of those words are counted in Reduce function. The procedure of words elimination process is summarized in Figure 4.5.

Procedure: Words Elimination

```
// key: key, value: document contents
method Map (LongWritable key, Text value) {
    for each word w in value
        w ← w.replaceAll("[^A-Za-z]+$", "").trim();
        if (w.length( ) < 4 || w.length( ) > 20)
            w ← w.replaceAll(w, "").replaceAll("\\s" + " ").trim();
        endif
        Emit (w, one);
    endfor
}
// key: word, values: list of counts
method Reduce (Text key, Iterator values) {
```

```
        int result ← 0;
        for each value v in values
                result += v;
        endfor
        Emit (key, result);
}
```

Figure 4.5 Procedure of Words Elimination

4.2.1.3 Stopwords Elimination

Stopwords are very common words such that they do not have any semantic meaning and do not help much in the process of topics extraction, and hence these words can be eliminated to save the time and efforts in processing of text data. In this step of elimination, the stopwords which appear redundantly in almost every document are removed. Moreover, the words which occur less than five times in the dataset are also eliminated to speed up the topics extraction process. The procedure of stopwords elimination process is described in the Figure 4.6.

Procedure: Stopwords Elimination

```
Read stopword file from DistributedCache
// key: key, value: word, count
method Map (LongWritable key, Text value) {
        for each word w in value
                c ← extractInt(w);
                if (c >= 5 && !w.matches("[0-9]+") && !w.isEmpty())
                        if (!stopWordList.contains(w))
                                Emit (w, null);
                        endif
                endif
        endfor
}
```

Figure 4.6 Procedure of Stopwords Elimination

4.2.1.4 Spell-Checking

After removing the stopwords, the spell-checking step is performed based on the dictionary file. The procedure for the spell-checking process is summarized in Figure 4.7. **T**he spell checking is performed to yield the pre-processed words as final results. For training CTM model, spell checking has a significant influence on the model's results.

Procedure: Spell-Checking

```
Read dictionary file from DistributedCache
// key: key, value: word
method Map (LongWritable key, Text value) {
        for each word w in value
                if (!dictionaryList.contains(w))
                        Emit (w, null);
                endif
        endfor
}
```

Figure 4.7 Procedure of Spell-Checking

6.3.2 Model Training

Given a pre-processed full-text dataset, the MapReduce CTM model is trained to learn the underlying hidden thematic structure of the dataset. The CTM describes each document by a distribution of topics, and each topic by a distribution of words. After building the model, the word distributions for each topic and the topic distributions for each document are computed as the model results. The sample input and outputs of the model are shown in Figure 4.8.

From the model's results, the topic representation of documents allows to summarize the whole documents collection without requiring the prior knowledge. In other words, it provides an interpretable latent structure of the documents so that it can be realized and recognized by the humans. The model training phase has two sub-tasks: topics extraction task and topics validation task.

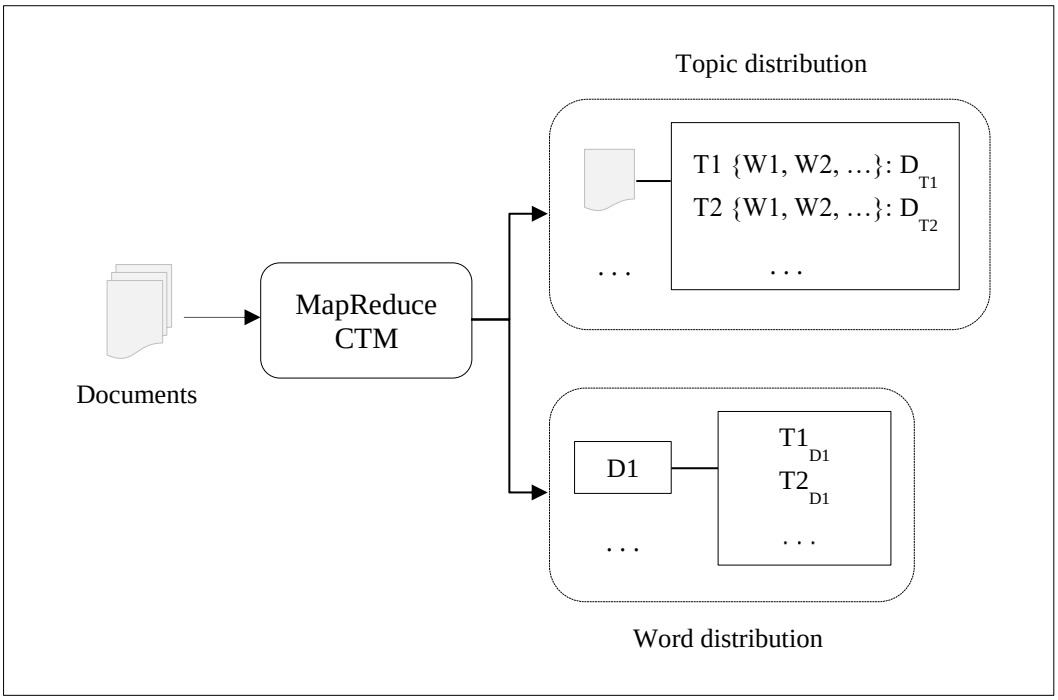


Figure 4.8 Sample Input and Outputs of MapReduce CTM

4.2.2.1 Topics Extraction

The estimation of parameters for CTM is chaotic problem and requires variational inference algorithm to solve the problem. Therefore, the variational inference of CTM is adopted to extract the topics from the full-texts collection. In this work, the variational EM algorithm of CTM implemented in MapReduce framework is proposed to solve the parameter estimation problem and to handle the volume of a collection of full-texts documents.

The entire MapReduce variational EM algorithm is divided into three classes: the Driver, the Mapper and the Reducer classes. The Driver class takes the control of the whole inference process and the responsibility of submitting the MapReduce job to the Hadoop cluster for execution. It first accepts the input dataset from HDFS and divides it into fixed-sized pieces called input splits. The Driver is also responsible for the initialization of the model parameters $\{\mu, \Sigma, \beta_K\}$ and the variational parameters $\{\lambda_i, v_i^2, \zeta, \phi_{n,i}\}$. The following Figure 4.9 provides the training procedure of the Driver class of variational EM algorithm.

Procedure: Driver Class

Input: Number of topics K, A corpus consisting of D documents,

N_d words in document d

Output: Model parameters $\{\beta_{1:K}, \mu, \Sigma\}$

Initialize: Variational parameters: $\lambda_i = 0, v_i^2 = 0, \zeta = 1, \phi_{n,i} = 1/K$

for $i \in K$ and $n \in N_d$

Model parameters: $\mu = 0, \Sigma = 1, \beta_i = 0.01 + rand()$

Procedure:

Call Mapper algorithm

Call Reducer algorithm

Figure 4.9 Procedure of Driver Class

Figure 4.10 depicts the interconnections between the Driver, Mapper and Reducer of the workflow of MapReduce CTM.

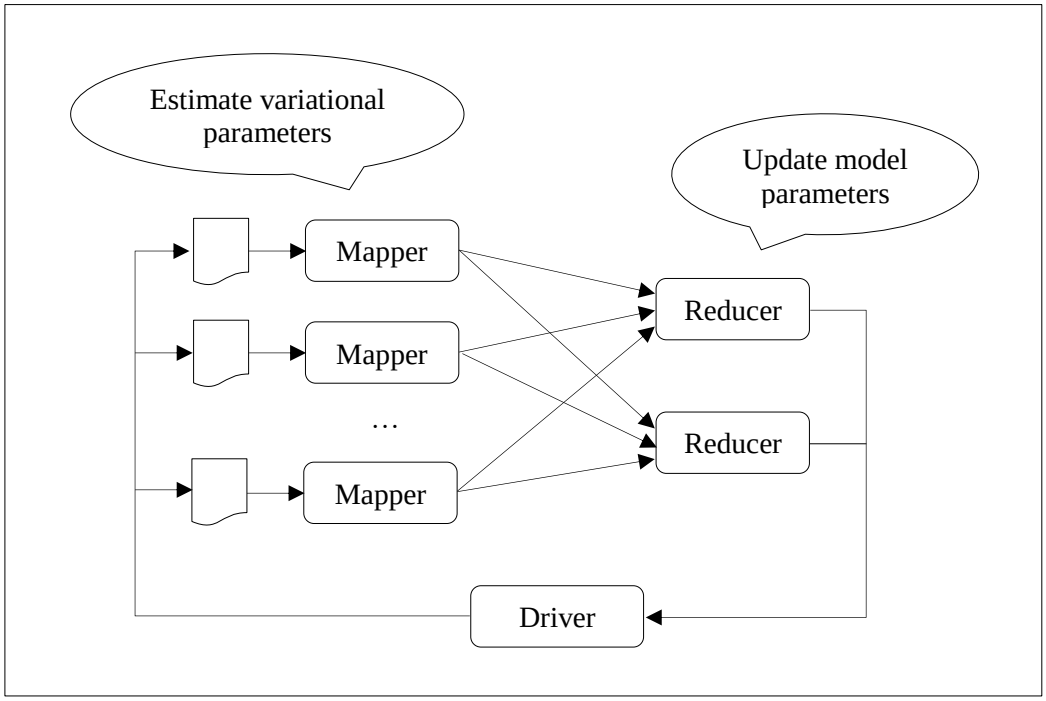


Figure 4.10 Workflow of MapReduce CTM

The number of topics K is user specified, and the corpus D is determined by the data. To find the likeliest parameter estimates for the MapReduce CTM model which

depends on latent variables (the topics), the variational EM algorithm is applied in MapReduce framework.

For the variational EM algorithm to be implemented, the E-step is executed in the Mapper class and the M-step is executed in the Reducer class. A pair of Map and Reduce functions comprises a single iteration of the variational EM algorithm. After each MapReduce iteration, the Driver updates the model parameters β, μ and Σ. The detailed procedures of Mapper and Reducer classes of MapReduce CTM are summarized in Figure 4.11 and Figure 4.12, respectively.

Procedure: Mapper Class

//key: documentID, value: document contents
method Map (Intwritable key, Document value) {
repeat
    for $n = 1$ to N_d
        for $i = 1$ to K
            Update ζ with $\zeta = \sum_i exp(\lambda_i + v_i^2/2)$
            Update $\phi_{n,i}$ with $\phi_{n,i} = exp(\lambda_i)\,\beta_{i,n}$
        endfor
    endfor
    Update λ_i with $dL/d\lambda = -\sum^{-1}(\lambda - \mu) + \sum_{n=1}^{N} \phi_{n,1:K} -$
    $(N/\zeta)\, exp(\lambda + v^2/2)$
    Update v_i^2 with $dL/dv_i^2 = -\sum_{ii}^{-1}/2 - (N/2\zeta)\, exp(\lambda + v_i^2/2) + 1/(2v_i^2)$
until convergence
Emit (key, variational parameters);
}

Figure 4.11 Procedure of Mapper Class

In the Mapper class, the MapReduce framework creates a new map task for each input split. Since the files are smaller than the HDFS split size, the number of map tasks is equal to the number of input splits. The objective of Mapper is to update and estimate the variational parameters for each document. The Map function reads each record of the input dataset stored in HDFS and maps input key-value pairs to intermediate key-

value pairs. Then, the Mapper executes the E-step and outputs the updated variational parameters to the Reducer.

Procedure: Reducer Class

```
//key: key, values: list of values
method Reduce (key, Iterator values) {
    for d = 1 to D
        for i = 1 to K
            Update β_i with β_i = Σ_d φ_{d,i} n_d
        endfor
        Update μ with μ = 1/D Σ_d λ_d
        Update Σ with Σ = 1/D (Σ_d diag(v_d^2) + Σ_d(λ_d − μ)(λ_d − μ)^T)
    endfor
    Emit (key, model parameters);
}
```

Figure 4.12 Procedure of Reducer Class

In the Reducer class, each reduce task receives the intermediate output produced from the Map function and performs operation on the list of values against each key. The objective of Reducer is to update the model parameters. Hence, the Reducer executes the M-step which updates the model parameters using the learned variational parameters from the E-step. The Reduce function emits the final output key-value pairs which are stored in HDFS.

4.2.2.2 Topics Validation

After applying the MapReduce CTM to extract the topics from the collection of documents, the discovered topics are validated to identify the optimal number of topics for the collection. In particular, choosing the appropriate number of topics is a crucial problem because the themes of the whole documents collection can best be described by the chosen number of topics. More specifically, too few topics will result in very broad topics and too many topics will result in uninterpretable topics.

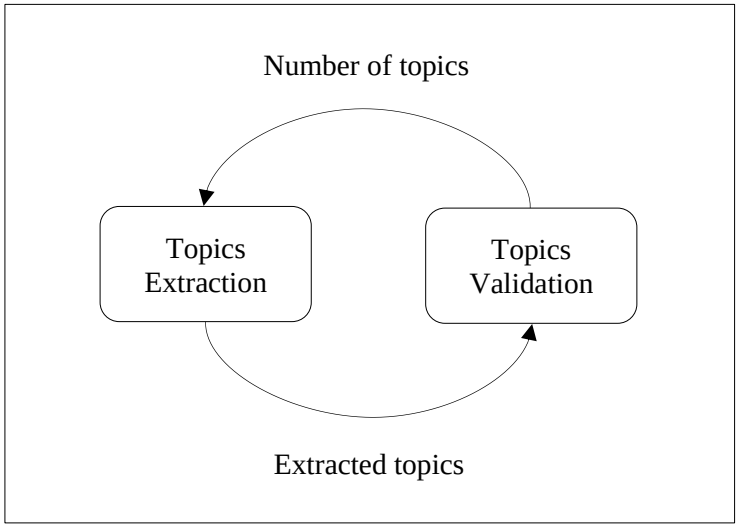


Figure 4.13 Topics Validation Process

Figure 4.13 illustrates the topics validation process of MapReduce CTM model. A number of techniques are introduced to determine the exact number of topics associated with the documents collection. In order to determine the optimal number of topics for the collection, several MapReduce CTM models are built with various values of number of topics. Then, choose the one which yields the highest mean coherence measure for the topic model. In this work, the two topic coherence measures, UCI and UMass, are employed to investigate the suitable number of topics. The one achieving the best results over the documents collection is selected as the optimal number of topics.

6.3.3 Similarity Computation

In many recommendation systems, the computation of similarity is indeed important in maintaining the quality of generated recommendations. Moreover, the measure of similarity affects the performance of recommendation system because it greatly influences the scalability and prediction accuracy of the recommendation system.

After the set of topics derived from the datasets are produced as results from the MapReduce CTM, the similarity between topics is computed based on the word distributions for each topic. It can be said that two documents are similar to the extent that they share the same topics, and two topics are similar to the extent that they have the same words appear in those topics.

One of the most commonly used similarity measures, cosine similarity is applied to measure the distributional similarity between two topics using the inferred word distributions. The cosine similarity measures the similarity between two vectors by calculating the cosine of the angle between them.

Let A and B be the two non-zero vectors of two topics, the cosine similarity $cos(\theta)$ is defined as follows:

$$similarity(A, B) = cos(\theta) = \frac{A \bullet B}{||A||\ ||B||} \quad (4.1)$$

where θ is an angle between two vectors, $A \bullet B$ is the dot product of the two vectors and $||A||\ ||B||$ is the product of the magnitude of each vector.

The value of $cos(\theta)$ is bounded between 0 and 1, and therefore, similar vectors will have a lower $cos(\theta)$ and dissimilar vectors will have a bigger $cos(\theta)$. Otherwise stated, the more similar the two topics are, the higher the cosine similarity and the smaller the angle between them is.

4.3 Generating Recommendation Stage

The primary task of this stage is the computation and presentation of the generated top-N recommendations to the active user of the proposed system. The proposed paper recommendation system assists users in retrieving the most relevant papers containing semantically related latent terms from large amounts of documents. When the keywords to be searched is entered by the active user, the processing of keywords, the extraction of documents and the computation of predictability are conducted. Once the documents to be recommended for the user are computed, the system displays the documents as the list of top-N recommendations to provide to the active user. Figure 4.14 shows the main procedure of the proposed recommendation system for generating top-N recommendations for the active user.

4.3.1 Keyword Processing

Keyword is the main descriptor to capture the essence or interest of the active user in the item, in this case the research document that the user wished for. After the user has entered the searched keywords to the proposed system, the steps of keyword

processing including tokenization and stopwords removal, are executed in order to support the cleaned keywords to the next phase.

Procedure: Recommendation

Input: Search query from the user

Output: Top-N recommendation list

1. Process the user's search query
    a. Extract keywords
    b. Remove stopwords
2. Generate the topics similarity matrix based on the topics extracted from the MapReduce CTM model
3. Retrieve the documents by matching the keywords to the top N words of each topic
4. Calculate the entropy between keywords and candidate documents to retrieve the more relevant documents
5. Rank the documents in increasing order with respect to entropy values
6. Display the documents as top-N recommendations list

Figure 4.14 Procedure of Recommendation System

4.3.2 Documents Extraction

The process of extracting documents involves matching the user's search query to the extracted topics of the model training phase. For each keyword of the query, the system matches the keyword to the top-N topic words of each topic. Then, the topic containing the keyword is taken out. If more than one topic contains the keyword, the system chooses the topic that has the highest word distribution of the keyword. As well, according to the topics similarity computation, the system selects the other topics that have similarity relationship with the chosen topic. Herein, the system considers all the matching documents that have topical relevance with the chosen topics and adds them to the candidate documents list.

4.3.3 Predictability Computation

In order to improve the accuracy of recommendations, the predictability relationship between the user's keywords and candidate documents are computed. To measure the predictability, the information theoretic measure called entropy is applied in the proposed system. Indeed, the entropy quantifies the level of uncertainty of the words. Besides, the concept of predictability opposite to that of uncertainty, and hence, a measure for predictability can be derived from a measure of entropy.

A prerequisite for predictability computation is the estimation of the probability of occurrence of the keyword in each candidate document. The probability of a word $p(w)$ in a document can be estimated as:

$$p(w) = \frac{frequency_w}{\sum_{i=1}^{W} frequency_i} \tag{4.2}$$

where $frequency_w$ is the frequency of the keyword in the document and W is the total number of words in the document. Given these probabilities, the entropy of the keyword in the document can be calculated as follows:

$$Entropy = -(p(w)\ log\ p(w)) \tag{4.3}$$

The entropy value of the word is in the range between 0 and 1. Finally, the candidate documents are sorted based on the increasing entropy measures of the query keyword. The lower entropy indicates the higher predictability. Hence, the candidate document with the lowest entropy is ranked first and the ranked order of the documents determines the ranking of the top-N recommendations list.

4.4 Chapter Summary

This chapter presented the architecture of the proposed recommendation system for the recommendation of relevant and related research papers. To recommend the unseen and latent items that are semantically related with the searched query, the MapReduce CTM model with variational EM algorithm is introduced and utilized in the proposed system in order to extract the latent topics. The MapReduce CTM model is trained using the MapReduce implementation in a Hadoop cluster to increase the

scalability of the paper recommendation system. In order to find the optimal number of topics, the CTM is trained with different number of topics and evaluated.

The system takes the search query in terms of keywords from the user and generates a top-N recommendation list that satisfy the needs and tastes of the users. To improve the accuracy of the recommendation results, the full-texts of research papers are analysed and the similarity between latent topics are computed. Then, to address the limitation of similarity prediction problem, the predictability between items is calculated by using entropy. In the above sections, the detailed explanations are described with the applied methods and algorithms. The implementation of proposed paper recommendation system is also described in Chapter 5. The experimental results and performance evaluations over the proposed paper recommendation system using the MapReduce CTM model are fully discussed in Chapter 6.

CHAPTER 5

IMPLEMENTATION OF THE PROPOSED SYSTEM

In this chapter, the implementation of the proposed paper recommendation system based on Correlated Topic Model in MapReduce framework is developed to recommend the unseen and relevant research papers by producing a list of recommendations to the user and therefore save the user's time and effort.

According to the proposed system pipeline discussed in Chapter 4, the implementation of offline learning stage and generating recommendation stage are described in this chapter. At first, a Hadoop cluster is configured to create a distributed computing environment on a single machine in order to implement the offline learning process of the proposed system. Then, the implementation procedures for the generating recommendation process is presented with system demonstrations by offering a user-friendly GUI to provide a clear visual explanation.

5.1 Hadoop Cluster Configuration

Hadoop cluster is a special type of cluster for storing and analyzing large amounts of unstructured data in a distributed environment. In similar term, it is a computational cluster to distribute the data analysis workload across multiple nodes of the cluster in order to perform parallel processing of the data. The architecture of Hadoop cluster is a Master/Slave topology which is composed of a single Master and a group of Slaves running on either physical or virtual machines. The Master node consists of two components: NameNode and ResourceManager. Each Slave node has two components: DataNode and NodeManager. The main features of Hadoop cluster are high scalability and high resiliency to failure.

Before starting the configuration for the Hadoop cluster, the meaning of the different components of a Hadoop cluster are covered as follows to understand the Hadoop architecture.

- NameNode (NN): The NameNode is the centrepiece of HDFS that tracks the files, manages the distributed file system and knows the location of the stored

data blocks inside the Hadoop cluster. When performing MapReduce operations, the NameNode consults with DataNodes.

- DataNode (DN): The DataNode is the slave node of HDFS that manages the actual data physically stored on the node.
- SecondaryNamenode (SN): The SecondaryNamenode works as a helper daemon for the NameNode that performs the checkpoints of the file system in HDFS. The SecondaryNamenode regularly connects with the NameNode and builds snapshots of the directory information of the NameNode.
- ResourceManager (RM): The ResourceManager runs on master node that manages the YARN jobs and takes care of scheduling and executing processes on slave nodes.
- NodeManager (NM): The NodeManager runs on slave node that takes instructions from the YARN scheduler and manages execution of tasks on the node.
- JobHistoryServer (JHS): The JobHistoryServer logs all the information of the retired MapReduce jobs from birth to death.

5.1.1 Overview of Hadoop Cluster

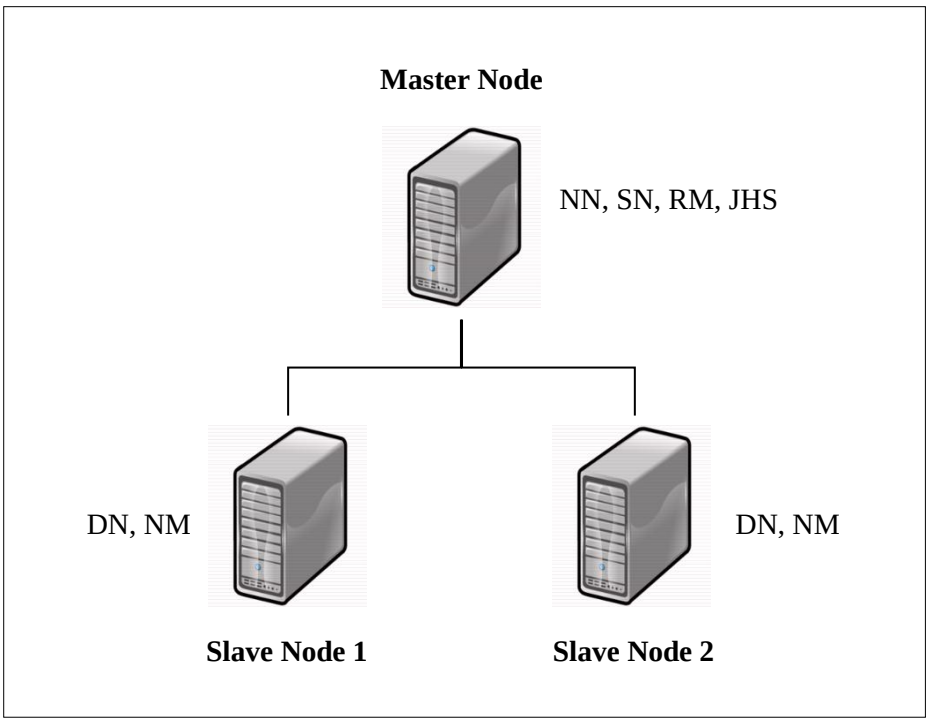


Figure 5.1 An Overview of the Hadoop Cluster

Figure 5.1 shows an overview of the Hadoop cluster consisting of 3 nodes (1 Master and 2 Slaves) running on 3 virtual machines and the Hadoop daemons running on each node. The Master node has four daemons running (NN, SN, RM, JHS) while each Slave node hosts two daemons (DN, NM).

5.1.2 Prerequisites for Hadoop Cluster

The prerequisites (platform and software requirements) for the configuration of a 3-node Hadoop cluster (1 Master and 2 Slaves) running on Ubuntu in a single machine are as follows:

i. VMware Workstation: The VMware® Workstation version 12 or later is downloaded and installed to create the 3 virtual machines (VMs).
ii. Operating System (OS): The Linux-based operating system (Ubuntu 16.04 or later) is downloaded and installed on each VM.
iii. Java: The Java 7 package or later is installed on each OS.
iv. Hadoop: The Hadoop 2.7.1 package or later is required to install Hadoop on each OS for setting up the Hadoop cluster.

All prerequisites must be applied on all the machines in the Hadoop cluster to create a distributed computing environment.

5.1.3 Installation Steps for Hadoop Cluster

The step-by-step installation instructions for setting up a Hadoop cluster on the nodes (Master and Slaves) are described in the section. Table 5.1 explains the Hadoop 3-node cluster with the private IP addresses of the three nodes. The server1 will act as NameNode and the server2 and server3 are DataNodes. On each node, the `/etc/hosts` file is edited to add the IPs in order to communicate with each other.

Table 5.1 Hadoop Cluster Description

Node	IP	Host Name
Master Node	192.168.227.10	server1
Slave Node 1	192.168.227.20	server2
Slave Node 2	192.168.227.30	server3

Step 1: Installing and configuring SSH

- The Master node uses an SSH connection to connect to Slave nodes and to manage the cluster.

Step 2: Installing Java

- Java must have installed prior to installing Hadoop.
- Download and decompress the Java package.
- Execute the `java -version` command to verify the Java has been installed on the system.

Step 3: Installing Hadoop

- Download and decompress the Hadoop 2.7.1 package.
- Modify `~/.bashrc` file for setting up the Hadoop environment variables.
- Modify `hadoop-env.sh` file for HDFS configuration.
- Check the Hadoop has been properly installed on the system by executing the `hadoop version` command.

Step 4: Configuring Hadoop on Master node

- Modify the Hadoop configuration files. Edit the `core-site.xml`, `hdfs-site.xml`, `mapred-site.xml` and `yarn-site.xml` files.
- Format the HDFS using the `hdfs namenode -format` command.

Step 5: Configuring Hadoop on Slave nodes

- Modify the Hadoop configuration files. Edit the `core-site.xml`, `hdfs-site.xml` and `mapred-site.xml` files.

Step 6: Starting the Hadoop cluster

- Start the HDFS, YARN and JobHistoryServer daemons by running the following scripts:

```
start-dfs.sh
start-yarn.sh
mr-jobhistory-daemon.sh start historyserver
```

- Check whether the Hadoop daemons are running or not by using the `jps` command. If the Hadoop cluster has started successfully, then the NameNode, SecondaryNameNode, ResourceManager and JobhistoryServer are running on the Master node whereas the DataNode and NodeManager are running on all Slave nodes.

Step 7: Accessing Hadoop on the browser

- Check the NameNode Web UI using the URL: `http://server1:50070/`.

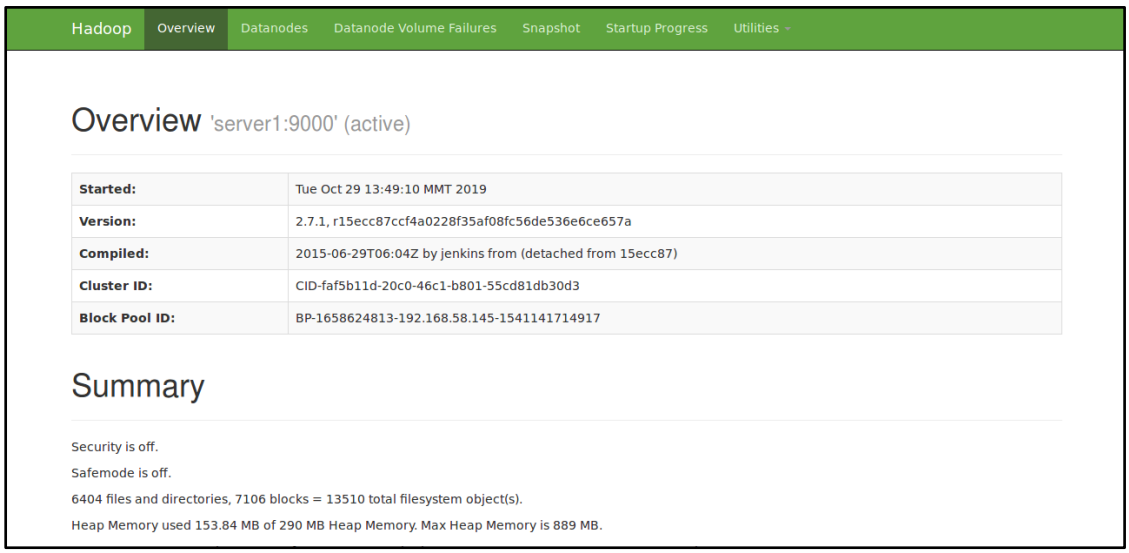


Figure 5.2 Web User Interface of the NameNode

Step 8: Stopping the Hadoop cluster

- Stop all the Hadoop daemons by running the following scripts:

```
stop-dfs.sh
stop-yarn.sh
mr-jobhistory-daemon.sh stop historyserver
```

5.2 Implementation of MapReduce CTM

The MapReduce CTM is proposed in the offline learning stage to alleviate the problem of LDA that is the incapability of modeling correlations among latent topics. In order to produce the latent topics with semantically related words, the proposed MapReduce CTM with variational EM algorithm is contributed to generate relevant

and reasonable results in the generating recommendation stage. The implementation of MapReduce CTM is described in Figure 5.3.

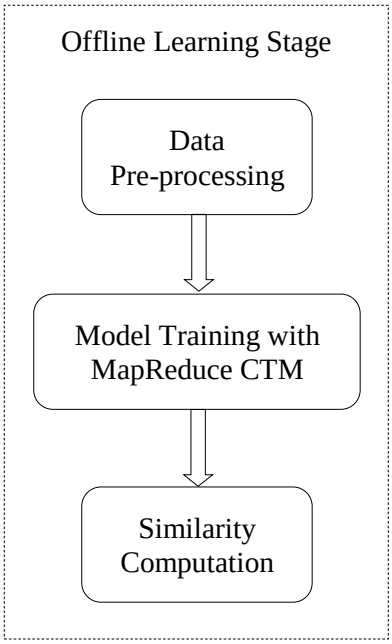


Figure 5.3 Implementation with MapReduce CTM

5.3 Implementation of Paper Recommendation System

In this section, the implementation of the paper recommendation system is carried out with the application of graphical user interface (GUI). All the implementations have been done using various tools and the programming has been done in Java using Eclipse IDE. The system is implemented as a Web-based system using Java Swing application to create a GUI which is acted as an interface between paper recommendation system and active user of the system.

The proposed paper recommendation system accepts the user's search query (keywords) from the user and performs the cleaning of keywords. Then, the topics extracted and validated from the MapReduce CTM which performed well on the recommendation task, were analysed. Rather than finding the whole dataset for the query keywords, the recommendation task is emphasized on the generated topics. Subsequently, the proposed system extracts the candidate documents and selects the most relevant documents for the user. Then, the generated recommendations are displayed to the active user through the GUI.

The user interface of paper recommendation system is composed with Topics, Similarity Matrix, Recommendation and Evaluation panels. The Topics panel displays the topics extracted by the MapReduce CTM, where the topics with top topic words are displayed. The generated topics similarity matrix is shown in the Similarity Matrix panel. The Recommendation panel contains a text box to enter the search query. Clicking the Search button displays the topics that contains the user's search keywords. The system also exhibits the top-N recommendations list which is ranked by the use of entropy values. Figure 5.4 shows the recommendation UI of recommendation system. Finally, the Evaluation panel presents the results obtained using precision and recall metrics.

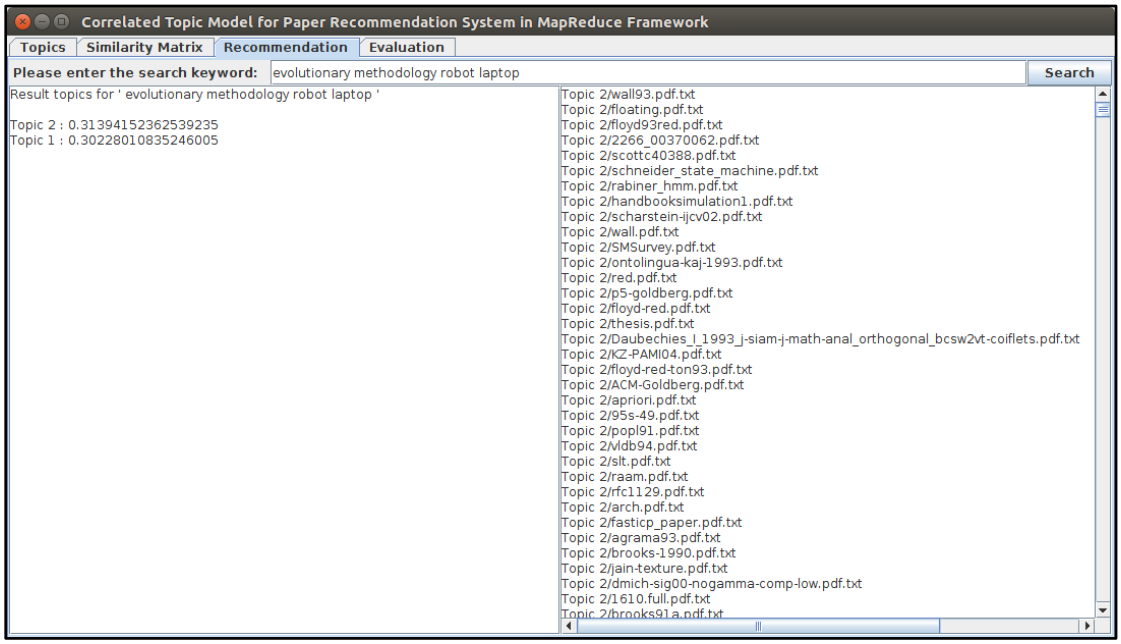


Figure 5.4 Recommendation UI of Paper Recommendation System

5.4 Chapter Summary

This chapter describes the implementation of the proposed MapReduce CTM model by utilizing the MapReduce framework in a Hadoop cluster. Also, the user interface of paper recommendation system is illustrated. Moreover, the configuration of the Hadoop cluster with step-by-step explanation is incorporated in this chapter. The empirical experiments and evaluation results of the proposed system are discussed in Chapter 6.

CHAPTER 6

EXPERIMENTAL RESULTS AND EVALUATIONS

In this chapter, the experimental study is presented and discussed on the performance evaluation of the proposed paper recommendation system along with the comparison of proposed MapReduce CTM and LDA. The recommendation results are accomplished by the required steps of pre-processing, model training, similarity computation, documents extraction and predictability computation. The aim of this chapter is to evaluate the accuracy and retrieval performance of the proposed paper recommendation system.

The experimental study of this chapter is divided into four main sections. The first section is the technical specification details of the host and virtual machines. The second section presents the two datasets on which the experiments and evaluations are conducted. The third section is the extracted latent topics evaluation of the proposed MapReduce CTM algorithm. The MapReduce CTM algorithm is applied in the paper recommendation system in order to generate the recommendations. Section 6.3 explains the thorough experiments and evaluation of this work. Finally, the fourth section is the performance evaluation of the proposed paper recommendation system based on the learned topics obtained from the MapReduce CTM. The experiments focus on this section are described in Section 6.4.

6.1 Experimental Setup

The detailed technical specifications of the host and virtual machines are described in Table 6.1.

Table 6.1 Specifications of Machines

Machine Type	Specifications
Host Machine	Intel® Core™ i7-7500U CPU @ 2.70GHz, 16GB RAM, 1TB Hard Disk
Virtual Machines	6GB RAM, 150GB Hard Disk (Master) 3GB RAM, 120GB Hard Disk (Slave1) 3GB RAM, 120GB Hard Disk (Slave2)

The environmental setting was executed on a host computer running Microsoft Windows 10 Operating System. The experiments of the MapReduce CTM are run in a Hadoop cluster consisting of 3 computing nodes (Virtual Machines) running Ubuntu 16.04 Operating System. All the experiments are done using Java implementation.

6.2 Datasets

The ongoing experiments are conducted based on the two datasets crawled from the CiteSeerX and PLOS ONE digital libraries. For each dataset, the data are collected in a reasonably short time frame of 5 hours from each library. Then, the crawled datasets are uploaded and stored in HDFS. The characteristics of the collected datasets in a 5-hour crawl are provided in Table 6.2. For each dataset, the total number of documents, the total number of sentences and the total number of words before pre-processing are reported.

Table 6.2 Characteristics of Datasets (Before Pre-processing)

Dataset	CiteSeerX	PLOS ONE
Number of Documents	266	148
Number of Sentences	569,636	407,309
Number of Words	3,691,405	2,696,316

Table 6.3 Characteristics of Datasets (After Pre-processing)

Dataset	CiteSeerX	PLOS ONE
Number of Documents	266	148
Number of Sentences	213,188	164,266
Number of Words	98,238	62,279
Size of Vocabulary	5,328	3,729

Table 6.3 describes the characteristics of the two datasets after the datasets are pre-processed to only contain useful data. The total number of sentences and the total number of words are decreased significantly. During pre-processing, the numbers, the punctuations and the words less than 4 and greater than 20 characters are eliminated. The last row of Table 6.3 reports the total number of words in vocabulary. For each

dataset, the vocabulary is learned from each dataset. For the extraction of vocabulary, all stopwords and all infrequent and misspelling words are eliminated. The CiteSeerX dataset has the larger vocabulary size while the PLOS ONE has the smaller size of vocabulary.

Table 6.4 Comparison of File Sizes

Dataset	File Size (in MB)	
	Before Pre-processing	**After Pre-processing**
CiteSeerX	20.7	4.9
PLOS ONE	15.7	3.9

Table 6.4 compares the dataset sizes before and after the pre-processing of datasets. Incidentally, the quality of the datasets has influences on the performance of the recommendation system. During the pre-processing of datasets, the small text files of each dataset are merged into a single large file and therefore, solving the small files problem of HDFS. Because Hadoop works better with a small number of large files and not with large number of small files. After the pre-processing, the sizes of pre-processed datasets are significantly reduced without decreasing the quality of the datasets. The cleaned datasets are stored in HDFS to perform the experiments.

6.3 Experimentation for MapReduce CTM

In this section, a series of experiments and evaluations designed to demonstrate the proposed MapReduce CTM model are discussed. The goals of this section are to verify: (1) what are the latent topics that are extracted from the dataset? and (2) what are the optimal number of latent topics that can best represent the whole dataset?

In topic modeling, there is no right answer for determining how many topics should be learned that is fixed for the given dataset. Thus, different values of the number of topics (varying from 2 to 10 topics) are trained for the determination of the number of topics that can represent and summarize the contents of the given dataset. In this way, the MapReduce CTM expresses all aspects of a training dataset by extracting the topics and validating the optimal number of topics for the dataset.

6.3.1 Topic Model Results

For the experiments, the topics generated from the two datasets using the proposed MapReduce CTM with variational EM algorithm are presented. Two different datasets, CiteSeerX and PLOS ONE datasets, are used to analyse and evaluate the efficiency of the MapReduce CTM. In the following, the MapReduce CTM model is trained to extract the semantically-related latent topics from two datasets.

For the parameter initialization of the MapReduce CTM model, the same initialization as described in Figure 4.9 is used, but considering a prefixed number of iterations. For the results, the variational EM algorithm was configured to run for 30 iterations. At the moment, the number of topics is arbitrarily set to 10 before investigating the optimal number of topics for each dataset. The model is built with 10 different topics for each dataset where each topic is a combination of words and each word contributes a certain weightage to the topic. The results for the two datasets are shown in Table 6.5 and Table 6.6, where the number of topics and a set of top-10 most probable words representing each topic are presented.

Table 6.5 Extracted 10 topics for CiteSeerX

Topics	Top-10 topic words
Topic 1	[data, algorithm, term, learning, problem, network, method, research, error, minimum]
Topic 2	[algorithm, data, network, problem, learning, system, probability, method, knowledge, minimum]
Topic 3	[data, problem, system, algorithm, probability, matrix, research, term, training, error]
Topic 4	[algorithm, data, network, research, problem, system, term, performance, probability, node]
Topic 5	[data, algorithm, problem, network, system, query, node, gateway, latency, probability]
Topic 6	[system, algorithm, problem, learning, probability, node, theory, network, query, data]
Topic 7	[data, node, performance, system, algorithm, problem, method, learning, probability, research]
Topic 8	[data, problem, algorithm, node, traffic, error, system, performance, research, network]

Topic 9	[data, problem, network, algorithm, system, node, query, term, cost, computer]
Topic 10	[algorithm, problem, data, method, node, latency, computer, gateway, structure, performance]

Table 6.5 shows the resulting 10 topics with the top-10 most likely words for CiteSeerX dataset. From Table 6.5, it can be seen that the words of each topic contain significant words relating to 'computer algorithm' and 'computer network'. Although all topics are easy to interpret, even so there are many duplicated words in all the topics. For instance, the words 'data', 'algorithm' and 'problem' are found in all topics and the words 'node', 'network' and 'system' are found in 7 topics.

Table 6.6 Extracted 10 topics for PLOS ONE

Topics	**Top-10 topic words**
Topic 1	[data, health, research, care, migration, primary, safety, area, surveillance, chapter]
Topic 2	[data, health, type, research, dose, security, global, population, mobile, children]
Topic 3	[data, health, migration, population, global, research, type, seismic, people, security]
Topic 4	[data, health, research, climate, seismic, access, object, people, population, migration]
Topic 5	[data, health, research, level, climate, type, global, safety, care, future]
Topic 6	[data, health, food, seismic, global, level, research, access, care, treatment]
Topic 7	[data, migration, research, level, type, food, population, climate, health, people]
Topic 8	[data, research, care, object, health, climate, food, seismic, area, treatment]
Topic 9	[data, health, global, research, population, food, care, access, area, assessment]
Topic 10	[research, migration, health, population, data, area, system, people, care, privacy]

Table 6.6 presents the 10 topics with top-10 words extracted from the PLOS ONE dataset. From the topics in Table 6.6, the top words of all topics talk roughly about

the concepts, such as ‘research in health care’, ‘migration of people’ and ‘climate change and seismic events’. Most topics combine words that are difficult and complex to interpret. Moreover, the words ‘data’, ‘health’ and ‘research’ can be found in all topics and so on. Therefore, by analysing the Table 6.5 and Table 6.6, many words are repeated in multiple topics showing that the number of topics set to 10 is too large for the CiteSeerX and PLOS ONE datasets.

Since the initial extracted results did not bring comprehensibility to the identification of the optimal number of topics, the training of MapReduce CTM model is intended to repeat with different topic numbers. In the next sections, topic coherence metrics are introduced and topic evaluations are performed to investigate and identify the optimal number of topics because the CTM model itself cannot verify the optimal number of topics. The hardest part of topic modeling is choosing the number of topics. Moreover, choosing the optimal number of topics depends on the nature of dataset. When too many topics are inferred from the model, it may get over fitted which is not expected at all. On the other hand, extracting too few topics does not make sense too. Therefore, it is important to identify the number of topics for the corpus by utilizing topic coherence metrics when training a topic model.

6.3.2 Topic Coherence Metrics

Since topics are not assured to be well interpretable to the coherence judgements of the humans, the topic coherence metrics are applied to reveal the semantic relatedness of the topics in order to measure the effectiveness of topic model. For the evaluations of the extracted topics from the proposed MapReduce CTM model, the two topic coherence measures, UCI and UMass, are used to investigate the optimal number of topics discovered by the MapReduce CTM. Also, the two measures can be used to distinguish between topics that are semantically interpretable topics and topics that are artifacts of statistical inference.

The topic coherence metrics measure the semantic quality of individual topics and correlate with human judgement of the topics [77]. The coherence of a single topic is scored by measuring the degree of semantic similarity between its high scoring words. Thus, by taking a mean of the coherence score per topic for all topics in order to get the topic coherence score for the topic model. In other words, all the per topic

level scores are needed to aggregate into one to get the topic coherence measure for the entire topic model.

The UCI coherence measure [115] is defined from the derivation of Pointwise Mutual Information (PMI). The PMI measures the correlation of two words by comparing the probability of observing the two words together to the probability of observing them independently. In UCI coherence measure, every single word is paired with every other single word. The UCI measure for a topic is described as follows:

$$C_{UCI} = \frac{2}{N \,.\, (N-1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} PMI(w_i, w_j) \tag{6.1}$$

where

$$PMI(w_i, w_j) = log \frac{P(w_i, w_j) + \varepsilon}{P(w_i) \,.\, P(w_j)} \tag{6.2}$$

where the word probabilities are computed by counting the word co-occurrence frequencies over an external corpus, the Wikipedia.

The UMass coherence measure [116] compares a word only to the preceding word and succeeding word, and thus need an ordered word set. The UMass measure for a topic is defined as follows:

$$C_{UMass} = \frac{2}{N \,.\, (N-1)} \sum_{i=2}^{N} \sum_{j=1}^{i-1} log \frac{P(w_i, w_j) + \varepsilon}{P(w_j)} \tag{6.3}$$

where the probabilities are derived by counting the number of documents containing words.

To estimate the probabilities, the external Wikipedia dataset is applied to reveal that the learned topics are coherent based on the external references. The Wikipedia covers a variety of topics related to a variety of domains. According to the experiments of Newman et al. [67], the authors suggested that applying the Wikipedia as external reference produces the best results.

The smoothing parameter ε is used to avoid taking the logarithm of zero for the words that are never cooccurred. With ε set to 1.0E-12, the scores of topic coherences are significantly decreased towards the higher negative values. Large negative values indicate words that don't cooccur often. Then, setting the smoothing parameter ε =

1.0E-6 instead of ε = 1.0E-12 gives the higher scores of UCI and UMass indicating that the generated topics have the better topic coherences.

6.3.3 Topic Coherence Evaluations

For the evaluations of the optimal number of topics, several MapReduce CTM models are built with different number of topics (ranging from 2 to 10) and the one that gives the highest coherence value is chosen as the optimal number of topics. Using different values of number of topics allows for the representations of the dataset at different granularity levels. For instance, number of topics 2 offers an overview of very general topics and number of topics 10 gives a specific set of topics.

In all experiments, the UCI and UMass coherence scores of the top-10 topic words from each topic are first computed, and then the global UCI and UMass coherence scores for the topic model is calculated by taking the arithmetic mean of the coherence scores of all topics. The global coherence score provides a good interpretation of the quality of topic model. Table 6.7 shows the results obtained for the UCI and UMass topic coherence measures based on the varying number of topics for CiteSeerX dataset.

Table 6.7 Mean Coherence Scores of MapReduce CTM for CiteSeerX

Number of Topics	CiteSeerX	
	UCI	**UMass**
2	**1.6951**	-2.4048
3	1.6332	-2.2687
4	1.611	-2.4751
5	1.333	-2.3335
6	1.6028	-2.3335
7	1.6201	-2.427
8	1.2835	**-2.164**
9	1.4137	-2.4177
10	1.637	-2.4466

From the above Table 6.7, the score of UCI measure is the highest at number of topics 2 and that of UMass measure is the highest at number of topics 8 (in this case,

numbers closer to zero indicates higher coherence). For a topic model, a higher coherence score means that it involves more reasonable topics containing the most probable words that are frequently co-occur together. Moreover, there is a significant difference between the mean coherence scores with regard to the number of topics, number of topics 2 for UCI and number of topics 8 for UMass. Table 6.8 and Table 6.9 present the extracted topics discovered by the MapReduce CTM ordered by UCI and UMass coherences, respectively. In the tables, each line is a topic composed of top-10 words that are semantically related with different degrees of relatedness.

Table 6.8 Extracted 2 topics ordered by UCI for CiteSeerX

Topics	Top-10 topic words	UCI
Topic 1	[data, algorithm, system, node, network, learning, probability, latency, performance, structure]	1.977
Topic 2	[problem, data, algorithm, method, term, query, theory, research, error, network]	1.4132

Table 6.9 Extracted 8 topics ordered by UMass for CiteSeerX

Topics	Top-10 topic words	UMass
Topic 4	[data, algorithm, problem, gateway, node, method, network, result, research, system]	-1.7161
Topic 6	[data, probability, problem, system, node, method, algorithm, network, term, performance]	-1.9693
Topic 3	[problem, algorithm, research, term, result, computer, error, memory, performance, data]	-2.0215
Topic 5	[data, system, algorithm, network, probability, problem, research, method, cost, path]	-2.068
Topic 7	[data, system, problem, network, algorithm, term, cost, tree, performance, method]	-2.0815
Topic 8	[data, algorithm, learning, node, performance, system, theory, computer, gateway, problem]	-2.36
Topic 2	[data, algorithm, network, term, problem, machine, fact, performance, latency, structure]	-2.4607
Topic 1	[problem, learning, algorithm, intelligence, query, energy, system, minimum, network, probability]	-2.6354

From the results in Table 6.8 and Table 6.9, it clearly shows that they both describe a general theme. As can be seen, the topics are easier to understand because the top words of each topic represent about the terms relating to ‘computer networking’. However, many words of each topic are being duplicated in many topics in most cases, for instance, the words ‘data’, ‘problem’ and ‘algorithm’ are found in almost every topic. It is probably an indication that the number of topics chosen is too large for the model because of containing many repetitive topic words. Table 6.10 gives the total counts of words for number of topics 2 and 8 (the highest coherence scores of UCI and UMass) for CiteSeerX dataset.

Table 6.10 Word Counts for Number of Topics 2 and 8 for CiteSeerX

Number of Topics	Count of Words
2	{structure=1, error=1, query=1, data=2, theory=1, problem=1, learning=1, research=1, network=2, algorithm=2, system=1, node=1, latency=1, term=1, performance=1, probability=1, method=1}
8	{computer=2, result=2, gateway=2, minimum=1, query=1, data=7, theory=1, network=6, algorithm=8, latency=1, tree=1, performance=5, path=1, probability=3, intelligence=1, structure=1, error=1, problem=8, learning=2, machine=1, cost=2, research=3, memory=1, system=6, node=3, fact=1, term=4, method=4, energy=1}

Table 6.11 presents the results obtained for the UCI and UMass topic coherence measures based on the varying number of topics for PLOS ONE dataset. The score of UCI measure is the highest at number of topics 5 and that of UMass measure is the highest at number of topics 4. For PLOS ONE dataset, the mean coherence score of UCI is adjacent to that of UMass as highly similar topic words are observed. Table 6.12 and Table 6.13 show the discovered topics for PLOS ONE dataset ordered by UCI and UMass coherence scores, respectively.

Table 6.11 Mean Coherence Scores of MapReduce CTM for PLOS ONE

Number of Topics	PLOS ONE	
	UCI	**UMass**
2	1.1926	-2.7862
3	0.9607	-2.4576
4	0.898	**-2.2495**
5	**1.2088**	-2.8253
6	1.0468	-2.4116
7	0.9919	-2.2975
8	0.9495	-2.3227
9	1.0789	-2.369
10	0.9398	-2.2709

After manually evaluating Table 6.12 and Table 6.13, the top words of all topics express about 'research data regarding health care'. It can be clearly seen that most topics in general are easily understood and interpretable. Although the extracted topics appear to be coherent, there are many redundant words that are found together in multiple topics, such as the words 'data' and 'health', which can affect the quality of the topic model. The total counts of words for number of topics 5 and 4 (the highest coherence scores of UCI and UMass) for PLOS ONE dataset are presented in Table 6.14.

Table 6.12 Extracted 5 topics ordered by UCI for PLOS ONE

Topics	Top-10 topic words	UCI
Topic 2	[data, health, research, global, migration, exposure, access, seismic, dietary, vessel]	1.758
Topic 5	[health, data, food, research, area, economic, migration, object, care, syphilis]	1.2491
Topic 4	[data, research, seismic, global, level, health, population, effort, climate, care]	1.165
Topic 1	[data, health, research, type, care, population, climate, seismic, scenario, chapter]	1.0645
Topic 3	[data, health, research, object, type, people, level, migration, treatment, patient]	0.8074

Table 6.13 Extracted 4 topics ordered by UMass for PLOS ONE

Topics	Top-10 topic words	UMass
Topic 1	[data, health, research, area, migration, level, climate, care, people, global]	-2.0021
Topic 2	[research, data, health, migration, seismic, safety, treatment, primary, climate, people]	-2.2765
Topic 4	[data, health, food, research, care, population, safety, patient, system, coastal]	-2.2853
Topic 3	[data, health, type, population, object, access, global, care, security, children]	-2.4342

Table 6.14 Word Counts for Number of Topics 5 and 4 for PLOS ONE

Number of Topics	Count of Words
5	{exposure=1, treatment=1, data=5, object=2, type=2, people=1, chapter=1, level=2, patient=1, area=1, care=3, syphilis=1, climate=2, population=2, migration=3, vessel=1, global=2, effort=1, health=5, access=1, research=5, economic=1, dietary=1, scenario=1, food=1, seismic=3}
4	{migration=2, treatment=1, global=2, data=4, children=1, security=1, object=1, access=1, type=1, health=4, coastal=1, research=3, people=2, system=1, patient=1, area=1, level=1, safety=2, food=1, primary=1, care=3, seismic=1, climate=2, population=2}

By careful analysis of the extracted topics together with the topic coherence measures, the extracted topics contain many duplicated words although all topics describe a general theme belonging a single concept. Words duplication in a topic model can cause an overestimation of the expressiveness of the model. In addition, comprising many duplicated words in the topics may overwhelm the most probable words in many of the topics and thus reducing the quality of topic models to interpret these topics.

In the previous experiments, when MapReduce CTM is trained with 30 maximum number of iterations, the resulting topics contained duplicated words and did not show distinguishable themes of the datasets. To gain a better interpretability of the

discovered topics, the inference process of MapReduce CTM is trained for a second time with 20 additional variational EM iterations. In the next experiments, the same hyperparameters initialization as described in Figure 4.9 is used and the variational EM algorithm of MapReduce CTM is configured to run for 50 maximum number of iterations. The mean coherence scores of the MapReduce CTM for number of topics 2 to 10 on datasets of CiteSeerX and PLOS ONE are computed.

Table 6.15 Mean Coherence Scores of MapReduce CTM for CiteSeerX

Number of Topics	CiteSeerX	
	UCI	**UMass**
2	2.97	-3.8635
3	2.4293	-4.9371
4	1.9878	-3.5997
5	**4.1404**	-4.4622
6	1.8912	**-2.8479**
7	2.5509	-3.5613
8	2.1087	-3.4191
9	2.453	-3.0853
10	2.0645	-3.5657

Table 6.15 shows the results obtained for the UCI and UMass mean topic coherence measures based on the varying number of topics for CiteSeerX dataset. The score of UCI measure is the highest at number of topics 5 and that of UMass measure is the highest at number of topics 6. The MapReduce CTM with 5 topics ordered by UCI coherence are shown in Table 6.16. The 6 topics discovered using the MapReduce CTM ordered by UMass coherence score are presented in Table 6.17.

Table 6.16 Extracted 5 topics ordered by UCI for CiteSeerX

Topics	Top-10 topic words	UCI
Topic 5	[substitution, dependency, awareness, machinery, lookup, turnover, dimensionality, stance, duality, abnormal]	5.3515
Topic 3	[dimensionality, cancellation, telnet, proximity, trapdoor, taxonomy, triangle, ford, chart, espresso]	5.163

Topic 4	[proximity, salesman, green, infinity, dimensionality, hart, automation, unlimited, velocity, yacc]	4.9156
Topic 1	[opinion, privacy, agenda, taxonomy, picture, espresso, substitution, stein, theoretic, locality]	4.1388
Topic 2	[chart, cipher, machinery, priming, lattice, residence, instant, attacher, minority, popped]	1.1329

Table 6.17 Extracted 6 topics ordered by UMass for CiteSeerX

Topics	Top-10 topic words	UMass
Topic 3	[computer, precision, multiple, error, operation, range, paths, barrier, candidate, language]	-1.8803
Topic 6	[error, motivation, comparison, cluster, message, intrinsic, preliminary, normal, servers, output]	-2.386
Topic 2	[candidate, language, computer, period, output, future, object, range, error, finding]	-2.6953
Topic 4	[computer, error, drop, maximum, topic, component, volatility, constraint, output, basis]	-3.0256
Topic 1	[error, lemma, operation, computer, global, range, segmentation, strategy, cluster, attitude]	-3.2135
Topic 5	[congestion, psychology, threshold, instruction, stock, semantic, cluster, economics, multiple, error]	-3.8866

By analyzing the Table 6.16, it can identify that all the 5 topics are in general understandable and involving the different aspects. The first topic, Topic 5, seems to describe a topic relating to ‘substitution of machinery’, and the next topics, Topic 3 and Topic 4, appear to represent a topic ‘dimensionality of proximity’. The last two topics, Topic 1 and Topic 2, seem to talk about ‘privacy of cipher machine’ with the exception of some words.

For the topics in Table 6.17, the topics on the top are easier to understand because the top words of these topics represent about the terms relating to ‘computer operation’. The less interpretable topics or mixed up topics can be found at the bottom of the Table 6.17. For instance, Topic 5 combines words about ‘computer network’ and ‘psychology and economics’. Moreover, only the word ‘error’ is found in all topics.

Table 6.18 presents the word counts for number of topics 5 and 6 (the highest UCI and UMass coherence scores) for CiteSeerX dataset. After manually evaluating

the topics, the MapReduce CTM with 5 topics is chosen as the best optimal number of topics for CiteSeerX dataset because of the more interpretability and diversity of topics with such number of topics used.

Table 6.18 Word Counts for Number of Topics 5 and 6 for CiteSeerX

Number of Topics	Count of Words
5	{machinery=2, lattice=1, minority=1, awareness=1, unlimited=1, substitution=2, hart=1, lookup=1, infinity=1, theoretic=1, dependency=1, stein=1, turnover=1, velocity=1, dimensionality=3, opinion=1, taxonomy=2, ford=1, stance=1, instant=1, popped=1, agenda=1, abnormal=1, espresso=2, green=1, cancellation=1, automation=1, privacy=1, duality=1, trapdoor=1, telnet=1, attacher=1, salesman=1, picture=1, proximity=2, triangle=1, residence=1, priming=1, yacc=1, chart=2, locality=1, cipher=1}
6	{computer=4, basis=1, range=3, drop=1, normal=1, motivation=1, congestion=1, constraint=1, candidate=2, segmentation=1, object=1, volatility=1, attitude=1, operation=2, barrier=1, stock=1, paths=1, component=1, servers=1, future=1, psychology=1, finding=1, instruction=1, intrinsic=1, multiple=2, topic=1, lemma=1, precision=1, error=6, global=1, semantic=1, strategy=1, preliminary=1, period=1, comparison=1, threshold=1, message=1, maximum=1, cluster=3, economics=1, language=2, output=3}

Table 6.19 describes the UCI and UMass mean topic coherence measures of MapReduce CTM with varying number of topics (from 2 to 10) for PLOS ONE dataset. Also, the maximum number of iterations for MapReduce CTM is set to be 50 iterations in this experiment. As seen from Table 6.19, the score of UCI measure is the highest at number of topics 5 and that of UMass measure is the highest at number of topics 10. Table 6.20 and Table 6.21 report the UCI and UMass topic coherence scores for the PLOS ONE dataset.

Table 6.19 Mean Coherence Scores of MapReduce CTM for PLOS ONE

Number of Topics	PLOS ONE	
	UCI	**UMass**
2	2.4463	-3.3494
3	2.8175	-4.3265
4	2.4856	-3.6837
5	**3.5993**	-3.994
6	1.8481	-3.4385
7	1.6605	-3.4759
8	2.3035	-3.4861
9	1.6792	-3.044
10	1.5913	**-2.8284**

Table 6.20 Extracted 5 topics ordered by UCI for PLOS ONE

Topics	Top-10 topic words	UCI
Topic 3	[anchor, appendices, breast, supplemental, registry, temp, transaction, ozone, authority, gestation]	4.8769
Topic 2	[signature, outlook, cent, breast, morbidity, reproductive, specification, procedure, shelf, protein]	3.4004
Topic 5	[republic, overlap, frame, addendum, registry, oxford, spice, reproductive, veterinary, shipping]	3.3783
Topic 1	[filename, reserved, directory, procedure, welfare, stem, discovery, reflect, origin, race]	3.2697
Topic 4	[injection, shelf, peak, prospective, registry, organ, radii, authority, greenhouse, loop]	3.0714

Table 6.21 Extracted 10 topics ordered by UMass for PLOS ONE

Topics	Top-10 topic words	UMass
Topic 6	[percent, effort, method, item, range, incident, protection, description, code, economic]	-2.2101
Topic 9	[dose, future, plot, percent, economic, confidentiality, months, range, meeting, description]	-2.2323
Topic 2	[dose, mobile, range, confidentiality, environment, protection, percent, blood, software, method]	-2.4871
Topic 10	[privacy, syphilis, legislation, mobile, department, extent, economic, effort, taxonomy, method]	-2.6528

Topic 7	[range, blood, method, protection, effort, code, description, plot, economic, arctic]	-2.7864
Topic 1	[privacy, arctic, density, months, effort, planning, range, dose, protection, mobile]	-2.8052
Topic 4	[syphilis, range, method, economic, strategy, environment, months, arctic, raster, plot]	-2.8139
Topic 8	[syphilis, arctic, training, confidentiality, future, mobile, percent, detection, acid, range]	-3.0173
Topic 5	[future, raster, extent, plot, incident, method, description, protection, syphilis, economic]	-3.1697
Topic 3	[mobile, future, floodplain, method, economic, protection, percent, integer, syphilis, node]	-4.1094

By inspecting the Table 6.20, it can be seen that all topics seem to talk about ‘file system’, ‘anatomy and physiology’, ‘environmental science’ and ‘social welfare’. To be more specific, each topic concerns more than one particular concept, however making it easy to distinguish the meanings of the topics. For instance, Topic 3 combines words related to ‘anatomy and physiology’ and words describing ‘file system’ and contains some irrelevant words.

For the topics shown in Table 6.21, one can observe that many words are duplicated in multiple topics. Furthermore, it seems that the top words of the topics provide a general summary of ‘economy’ (Topic 6) and ‘medical’ (Topic 9), for example. For Topic 2, the topic contains words relating to ‘medical’ and ‘technology’, but also bring the incoherent word ‘environment’. Although some topics seem coherent, unfortunately, many duplicated words and thus less informative topics are found with the highest UMass coherence score.

Table 6.22 shows the word occurrences of the topic words for number of topics 5 and 10 (the highest coherence scores of UCI and UMass). By analysing the results of the PLOS ONE dataset, the MapReduce CTM model with 5 topics is selected as the best optimal number of topics for PLOS ONE dataset due to the more understandable results with small number of topics used.

Table 6.22 Word Counts for Number of Topics 5 and 10 for PLOS ONE

Number of Topics	Count of Words
5	{greenhouse=1, peak=1, transaction=1, spice=1, race=1, anchor=1, reproductive=2, shipping=1, authority=2, ozone=1, protein=1, oxford=1, loop=1, shelf=2, morbidity=1, signature=1, procedure=2, republic=1, cent=1, prospective=1, overlap=1, gestation=1, outlook=1, addendum=1, injection=1, origin=1, welfare=1, appendices=1, reserved=1, breast=2, frame=1, organ=1, veterinary=1, directory=1, radii=1, specification=1, supplemental=1, filename=1, temp=1, stem=1, registry=3, discovery=1, reflect=1}
10	{percent=5, range=7, density=1, department=1, dose=3, plot=4, training=1, software=1, extent=2, description=4, future=4, taxonomy=1, arctic=4, syphilis=5, protection=6, integer=1, legislation=1, privacy=2, effort=4, strategy=1, code=2, blood=2, economic=7, raster=2, months=3, node=1, detection=1, environment=2, acid=1, planning=1, confidentiality=3, meeting=1, item=1, method=7, floodplain=1, incident=2, mobile=5}

As an additional check of MapReduce CTM model, the UCI and UMass coherence scores achieved with different number of iterations (30 and 50) are compared over various number of topics (2 to 10) for CiteSeerX and PLOS ONE datasets and the comparison results are illustrated in Figure 6.1 and Figure 6.2, respectively.

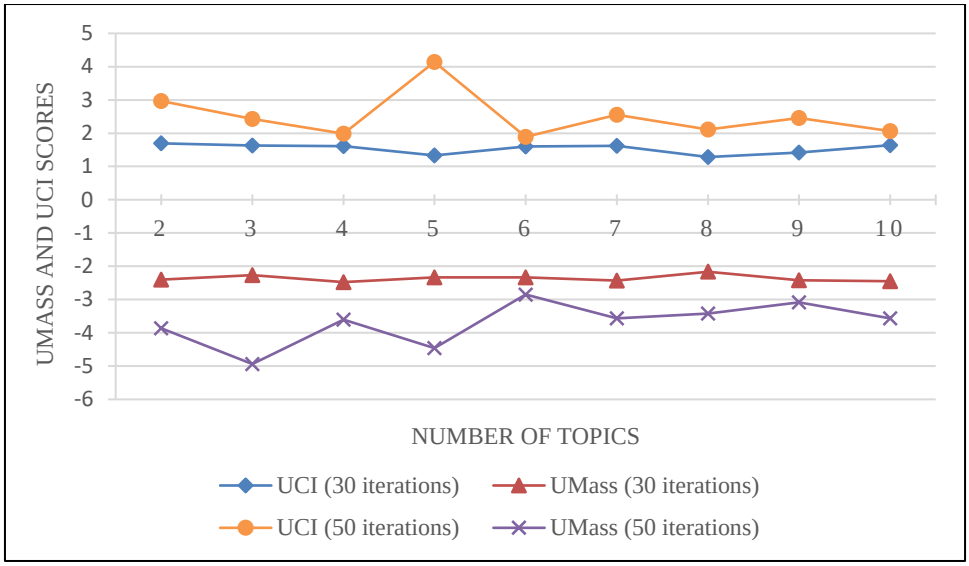


Figure 6.1 Comparison of Mean Coherence Scores for CiteSeerX

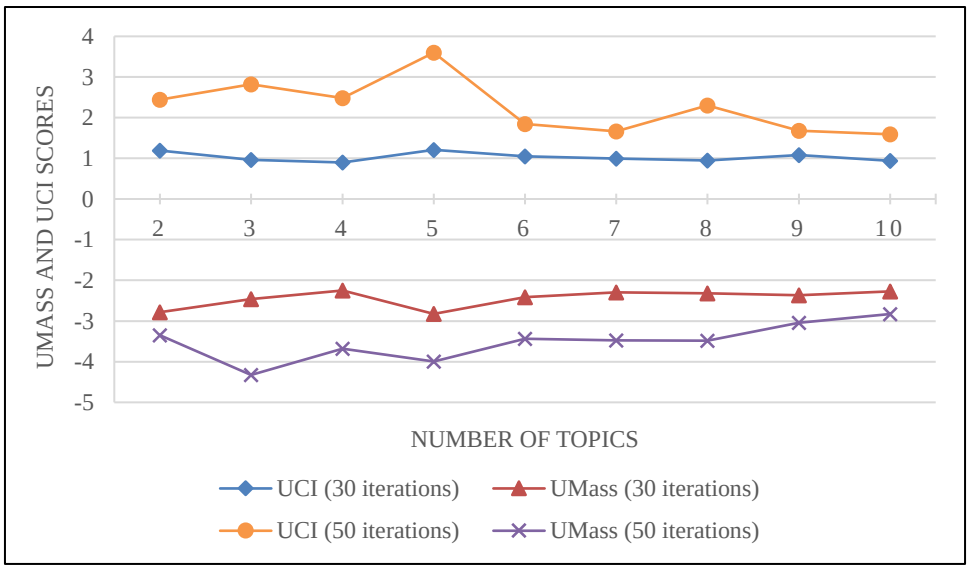


Figure 6.2 Comparison of Mean Coherence Scores for PLOS ONE

In Figure 6.1 and Figure 6.2, for both datasets learned by MapReduce CTM with 30 iterations, it shows that the UCI and UMass coherences remain steadily although the number of topics is increased. The reason is that many duplicated words are discovered in multiple topics. In general, the more duplicated the topic words, the less the diversity of the generated topics.

When the MapReduce CTM is trained with 50 variational EM iterations for the CiteSeerX dataset in Figure 6.1 (results from Table 6.7 and Table 6.15), the UCI

coherence gains better results as more semantically related topic words are learned, and therefore it reaches its best score at number of topics 5. However, the UMass coherence scores show no significant results compared with the UMass with 30 iterations. Hence, according to the results of Figure 6.1, the number of topics 5 was determined as the best optimal number of topics for CiteSeerX dataset based on UCI coherence score.

Again, for the PLOS ONE dataset in Figure 6.2 (results from Table 6.11 and Table 6.19), the UCI coherence gradually increases with the number of topics before it reaches its peak at number of topics 5. However, the increase is not significant as the number of topics gets higher (from number of topics 6 to 10) because of inducing less coherent topic words. Therefore, the number of topics 5 was selected as the most appropriate optimal number of topics for PLOS ONE dataset based on UCI coherence score.

It is reasonable to expect both UCI and UMass coherence scores to improve as semantically coherent topic words should be expected in the MapReduce CTM topic model. To be more specific, MapReduce CTM with 50 iterations achieves considerably higher UCI scores and yields better results than with 30 iterations. From the results, it can conclude that the proposed MapReduce CTM model not only provides a good interpretation and summarization of the collected datasets but also improves the topic coherence of the extracted topics as evaluated by the UCI and UMass coherence measures. Overall, the UCI and UMass coherence measures give a good picture so that one can take better decision in choosing the optimal number of topics.

6.3.4 Comparison between MapReduce CTM and Mr. LDA

In this section, the performance of MapReduce CTM is compared with Mr. LDA for CiteSeerX and PLOS ONE datasets. It is intended to have a measurable result in comparing the results of MapReduce CTM and Mr. LDA as a baseline. Therefore, the Mr. LDA topic model is adopted and the coherences of its topics are assessed. The Mr. LDA [108] is a distributed large-scale topic modeling algorithm implemented on Hadoop platform using Java.

Table 6.23 Mean Coherence Scores of Mr. LDA for CiteSeerX and PLOS ONE

Number of Topics	CiteSeerX		PLOS ONE	
	UCI	UMass	UCI	UMass
2	0.7576	-2.4834	0.2952	-2.0539
3	0.7852	-2.6996	-0.3463	-2.6333
4	0.3396	-2.7661	-0.2776	-2.2931
5	0.1602	-2.6239	1.0977	-2.5146
6	0.0305	-2.7861	1.3115	-2.9674
7	0.4697	-2.7333	1.2507	-2.9256
8	0.2333	-2.6714	1.3367	-3.0309
9	0.2803	-2.4873	1.2385	-2.7681
10	0.0103	-2.7389	1.5684	-2.9733

Table 6.23 describes the UCI and UMass coherence measures of different numbers of topics (ranging from 2 to 10) computed by the Mr. LDA with a comparable number of iterations for the CiteSeerX and PLOS ONE datasets, respectively. The results of Mr. LDA presented in Table 6.23 are compared with those of MapReduce CTM presented in Table 6.15 and Table 6.19 to evaluate the quality of both topic models, which are shown in Figure 6.3 and Figure 6.4.

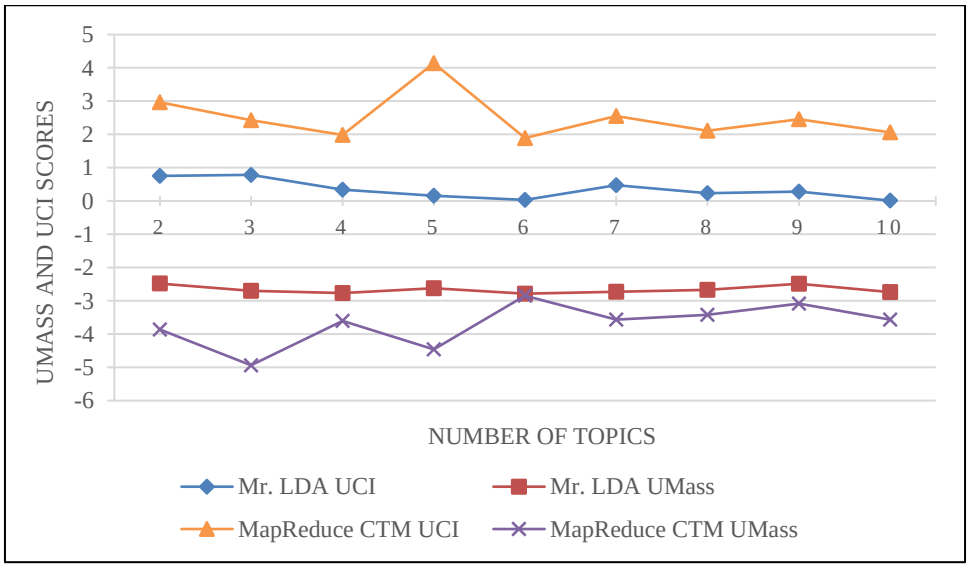


Figure 6.3 Comparison of Coherence Scores between MapReduce CTM and Mr. LDA for CiteSeerX

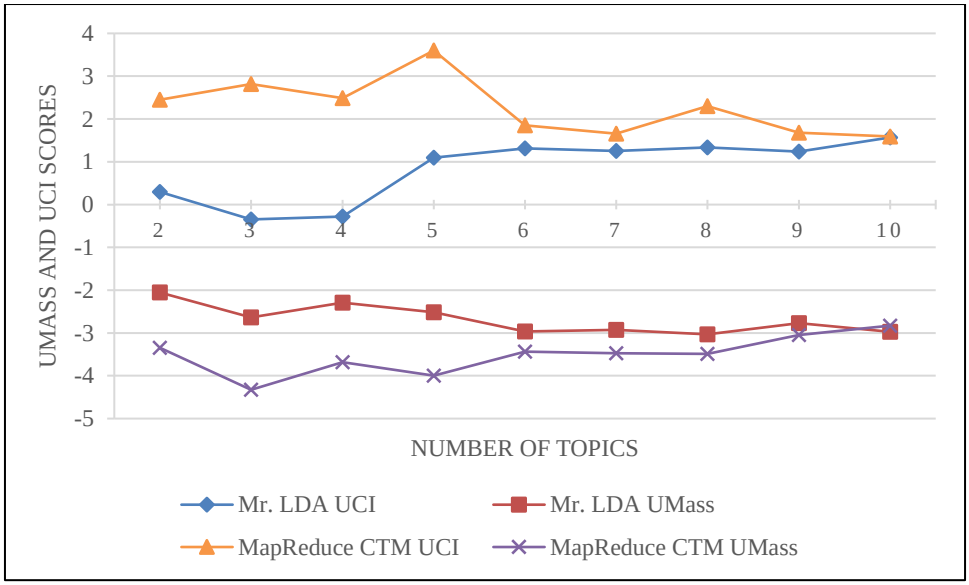


Figure 6.4 Comparison of Coherence Scores between MapReduce CTM and Mr. LDA for PLOS ONE

By analysing the Figure 6.3, it is interesting to note that the UCI scores of MapReduce CTM significantly higher than those of Mr. LDA along with the number of topics. At number of topics 5, the UCI score of MapReduce CTM reaches its highest point. In this case, the MapReduce CTM produces more reasonable topics containing more semantically related words than Mr. LDA. After this point, a substantial drop is occurred and starts flatten out from number of topics 6. In all cases measured by UCI coherence, MapReduce CTM learns better topics compared to those learned by Mr. LDA. On the other hand, the UMass scores of Mr. LDA have remained stable although the number of topics gets higher. It can be seen that the UMass scores of Mr. LDA are clearly higher than those of MapReduce CTM and only overlap with MapReduce CTM at number of topics 6. At which point, both the set of topics learned from MapReduce CTM and Mr. LDA seem to describe about the same topic, for example ‘computer networking’, but have different topic words.

According to the results in Figure 6.4, the UCI scores of MapReduce CTM substantially higher than those of Mr. LDA before reaching at number of topics 6. Specifically, it is suggested that one needs to pick the model that held the highest coherence score before flattening out. Due to this conception, the number of topics 5 is chosen for PLOS ONE dataset as described in Section 6.3.3. For the number of topics 6 to 9, the UCI and UMass scores of MapReduce CTM are slightly higher and lower

than those of Mr. LDA, respectively. Additionally, both measures remain steadily even the topics grow since the topic words are frequent and typically contain many senses.

According to the above experimental results, although the MapReduce CTM does not have relatively better performance when extracting topics for a particular dataset, it has a comparable performance as a topic modeling method. Experiments also showed that Mr. LDA appears to learn more general topics while MapReduce CTM appears to learn more coherent and specific topics. Moreover, compared to Mr. LDA, it can be seen that MapReduce CTM performs comparatively better on UCI coherence scores. Overall, the topic coherences of MapReduce CTM model slightly perform better than Mr. LDA in most cases measured with UCI score and perform marginally worse than Mr. LDA in some cases measured with UMass score.

6.3.5 Training Time Comparison

The comparative studies of training time between MapReduce CTM with 30 and 50 iterations using different values for number of topics from 2 to 10 for CiteSeerX and PLOS ONE datasets are shown in Figure 6.5 and Figure 6.6, respectively.

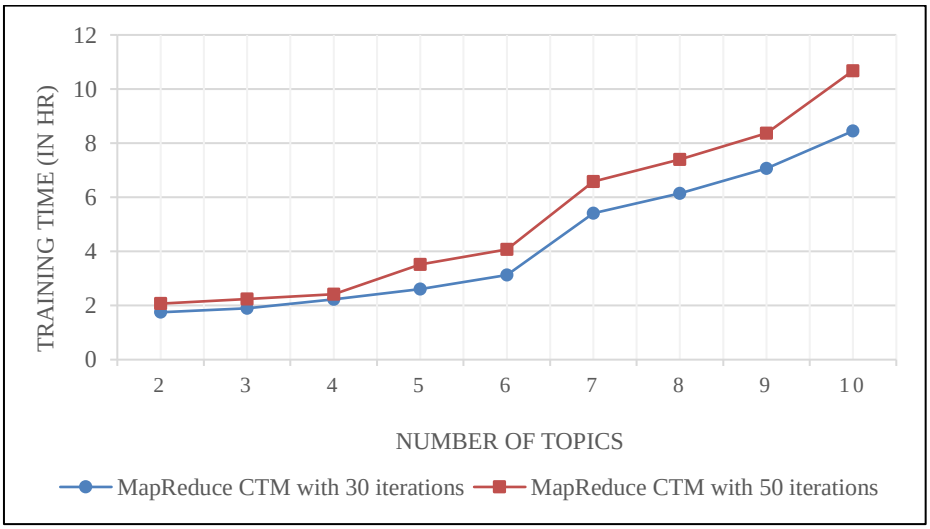


Figure 6.5 Training Time of MapReduce CTM for CiteSeerX

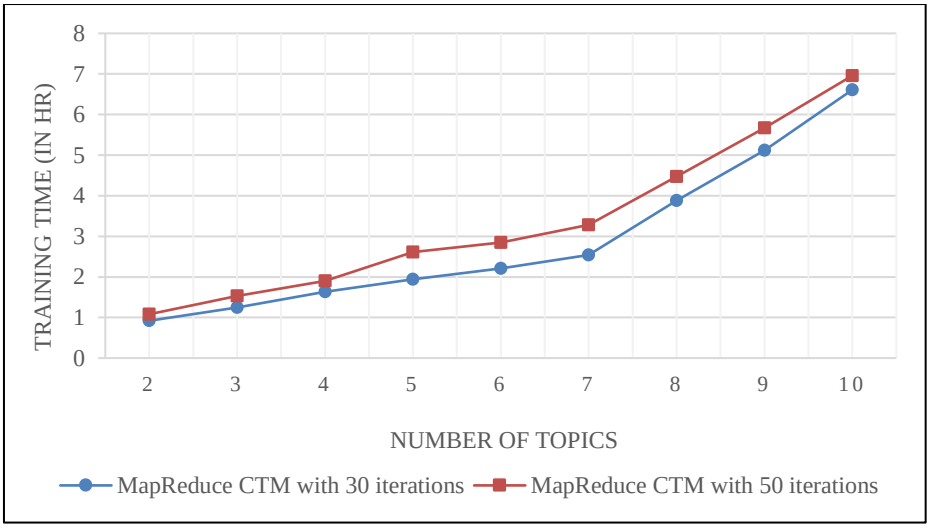


Figure 6.6 Training Time of MapReduce CTM for PLOS ONE

For the training of the two datasets by the MapReduce CTM, it was observed that the training time increases when the number of topics increases as a result of MapReduce CTM generally require more training time to converge with increasing values of topics. Moreover, the difference between the two gets larger as the number of topics gets higher. Since the training time of MapReduce CTM is dependent on the number of iterations, its per iteration training time is also considerably long. In addition, the training time of CiteSeerX is longer than that of PLOS ONE because of having the larger dataset size.

The MapReduce CTM is compared with Mr. LDA based on the varying number of topics. Figure 6.7 and Figure 6.8 demonstrate the comparison between the time taken for training of the CiteSeerX and PLOS ONE datasets using MapReduce CTM and Mr. LDA. By analyzing the results, the training time of MapReduce CTM is significantly higher than that of Mr. LDA. This confirms the fact that MapReduce CTM contains more variational parameters and requires more computations for the correlations between topics.

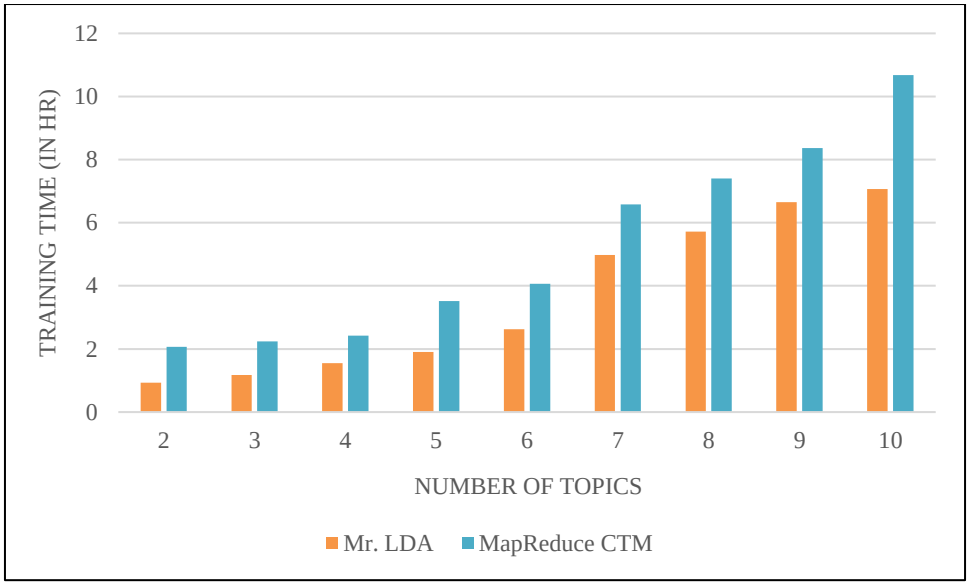


Figure 6.7 Training Time of MapReduce CTM and Mr. LDA for CiteSeerX

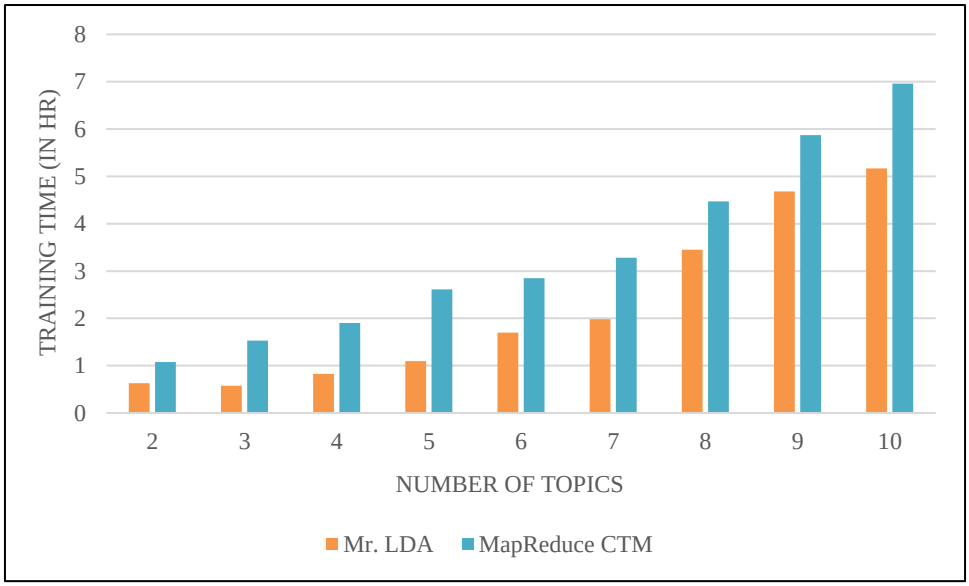


Figure 6.8 Training Time of MapReduce CTM and Mr. LDA for PLOS ONE

6.4 Experimentation for Paper Recommendation System

In this section, the offline experiments are conducted to evaluate the performance and effectiveness of the paper recommendation system. The ongoing experiments are performed after the top-N recommendations are provided to the user. The purpose of this section is to provide offline evaluations on the top-N

recommendations of the proposed paper recommendation system so that to generate relevant and related documents for the user.

6.4.1 Evaluation Metrics

Evaluation metrics are essential in judging the quality and performance of the recommendation systems. In order to evaluate the top-N recommendations that the proposed paper recommendation system provides, the most frequently used classical evaluation metrics, precision and recall [75], are considered in the experiments. Precision measures the exactness, whereas recall measures the completeness of the recommendations. Specifically, these metrics are applied to evaluate the retrieval performance of the paper recommendation system so that to estimate the relevancy of a set of top-N recommendations.

Precision is measured by the number of relevant documents containing in the recommendation list out of the total number of retrieved documents. In other words, it assesses the capability of the recommendation system to improve as much relevant documents as possible in reacting the user query. The calculation of Precision is stated in Equation (6.4):

$$Precision = \frac{Number\ of\ relevant\ documents\ retrieved}{Total\ number\ of\ retrieved\ documents} \tag{6.4}$$

where the larger value of precision indicates that the recommendation system generates the more accurate recommendation results.

Recall is measured by the number of relevant documents containing in the recommendation list out of the total number of relevant documents. On the other way, it assesses the capability of the recommendation system to improve as few irrelevant documents as possible in reacting the user query. Equation (6.5) defines the Recall calculation:

$$Recall = \frac{Number\ of\ relevant\ documents\ retrieved}{Total\ number\ of\ relevant\ documents} \tag{6.5}$$

where the larger value of recall indicates that the recommendation system sets the most relevant documents in the top place of the recommendation list.

6.4.2 Evaluation Results

Herein, the offline evaluations are performed using the evaluation methodologies presented in the previous section on two datasets, CiteSeerX and PLOS ONE. All the results have been calculated with different cut-offs based on the varying number of topics using the proposed recommendation algorithm.

Since the list of recommendations is influenced by the entropy values and the precision and recall do not seem to consider about ordering, the performance of the proposed recommendation system is only considered from rank *1* through rank *k*. Here, *k* is the number of evaluated recommendations of the top results of the generated recommendations list. Thus, instead of evaluating all recommendations from the top-N list, the precision and recall are reconstructed to precision and recall at cut-off *k* (Precision@*k* and Recall@*k*) because the recommended papers are more relevant when they appear earlier in the top-N recommendation list.

For the evaluations, several experiments are carried out for each dataset. Each experiment is repeated 10 times and the average precision and recall results are shown. Firstly, the precision and recall at cut-off 50 calculated based on the top-100 topic words on CiteSeerX dataset are presented in Figure 6.9. After that, Figure 6.10 shows the precision and recall at cut-off 100 calculated based on the top-100 topic words for CiteSeerX dataset. All the results are considered for each number of topics varying from 2 to 10.

From Figure 6.9 and Figure 6.10, it can be observed that the precision and recall values increase accordingly as the number of topics increases. As can be seen, the recall values are quite higher than the precision values and thus the paper recommendation system is capable of generating all relevant documents at the top of the recommendations list with the use of entropy. Moreover, raising the cut-off value *k* from 50 to 100 makes the precision and recall values to be lessened.

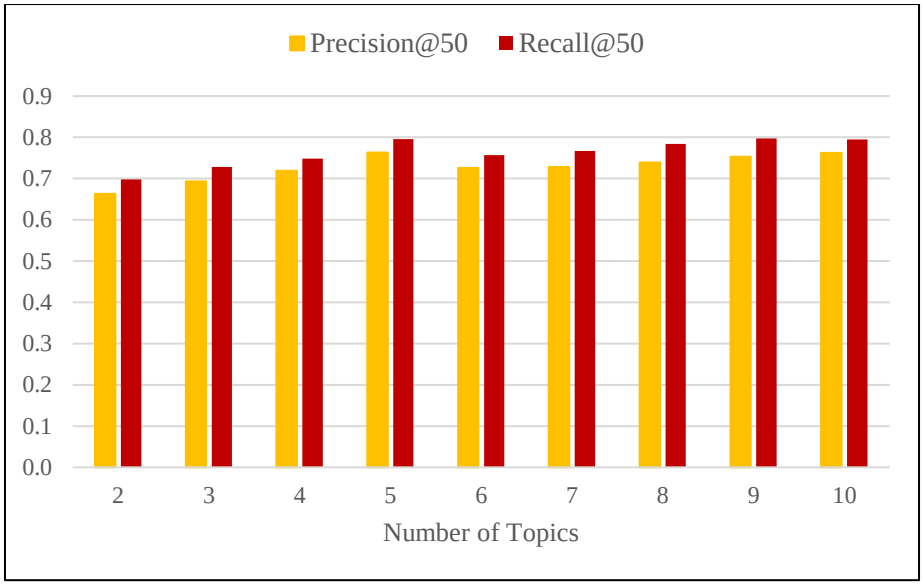


Figure 6.9 Precision and Recall with top-100 topic words at cut-off 50 on CiteSeerX

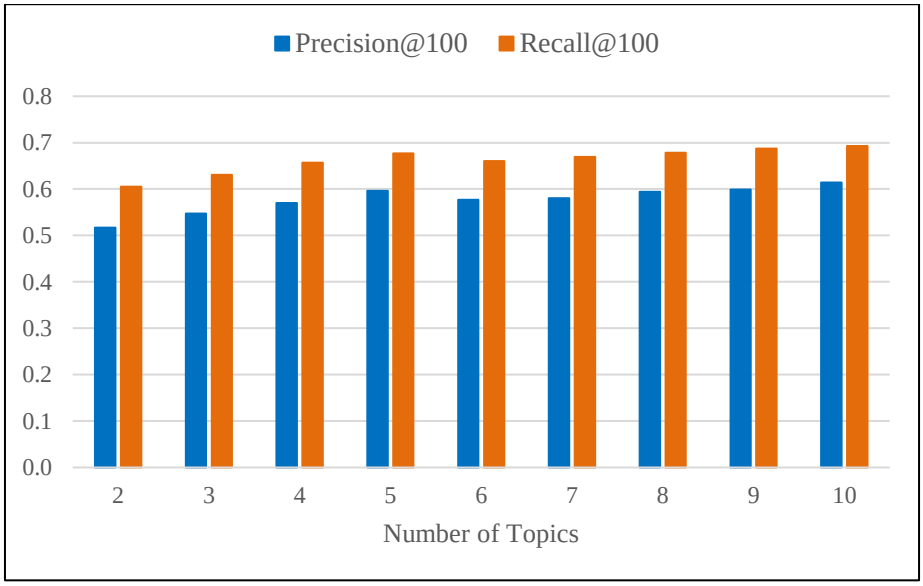


Figure 6.10 Precision and Recall with top-100 topic words at cut-off 100 on CiteSeerX

Figure 6.11 presents the precision and recall results for PLOS ONE dataset. Since the number of extracted documents for each topic is not exceeding 40, the cut-off is set as 25 for PLOS ONE. From the Figure 6.11, it can be analyzed that the performance of proposed paper recommendation system improves as a result of having the higher recall values. This is because the recall is the proportion of the number of

recommendations that are relevant to the number of all possible relevant documents. High recall is particularly important in paper recommendation systems where the most relevant documents are never missed and recommended to the user based on the search query.

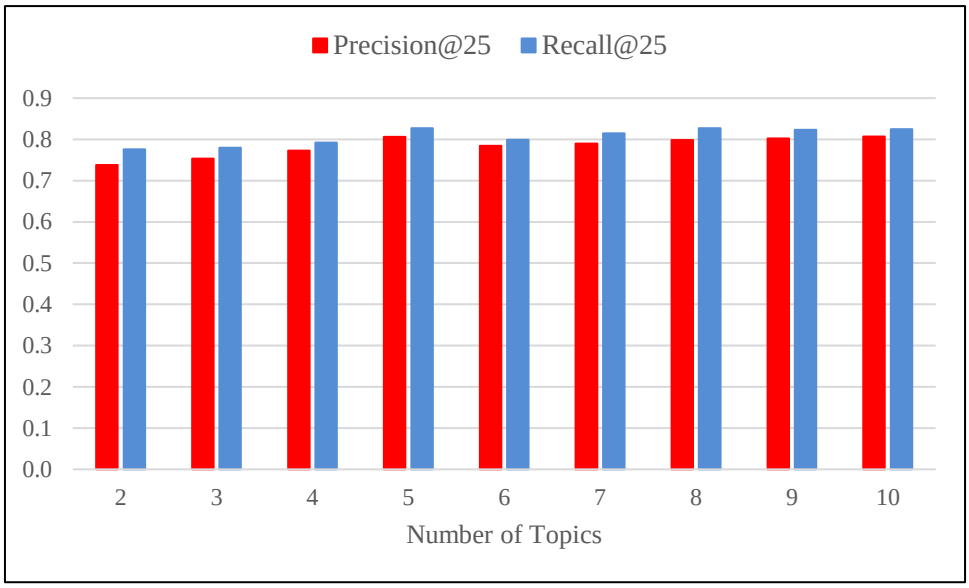


Figure 6.11 Precision and Recall with top-100 topic words at cut-off 25 on PLOS ONE

The proposed paper recommendation system intends to optimize the search results by generating the more relevant and coherent papers on the top of the displayed recommendation list. To achieve this, the proposed approach trains the dataset to extract the latent topics that can summarize the reflected theme of the dataset. To catch the semantic relatedness between the topics, the topic similarity matrix is generated. The topic words of extracted topics are then matched with the user search query to get the topics that has the highest word distribution of the keyword. To generate the more relevant documents, the entropy is used to determine the predictability relationship between the query keywords and candidate documents and to reduce the time that the user looking for the documents from the recommendations list.

From the above experiments, it can observe that the growth in the number of extracted topics not only enhances the performance metrics but also increases the recommendation response time. Moreover, the length of user query keywords also increases the search time so that the keywords are matched with the top-100 topic words

of the topics. In addition, the precision and recall values depend on the cut-off value and the top topic words of each number of topics. The precision and recall results with top-10 topic words generally have lower values than with top-100 topic words. Finally, for the paper recommendation system, a very high recall means that the system recommends the items that the users would have normally favoured without the recommendations. Overall, the system could reach 76% and 79% of the best performance when Precision@50 and Recall@50 for CiteSeerX, and reach 80% and 82% when Precision@25 and Recall@25 for PLOS ONE.

Moreover, it is worth to note that there are two certain limitations on precision and recall metrics. In the first case that the recommendation system does not have any relevant items to be recommended, i.e. the total number of retrieved items is zero, so the precision value should be set to 1. Also, in the latter case of not having the relevant items in the top-N results, i.e. the total number of relevant items is zero, thus the recall value should be set to 1. Therefore, it is important to optimize both precision and recall metrics in order to improve the performance of the recommendation system.

6.5 Chapter Summary

This chapter focused on the experiments and evaluations of the proposed MapReduce CTM model to provide an effective approach in recommending research papers in the proposed recommendation system. Normally, the latent topics can be inferred by using some topic modeling tools such as the MALLET package. However, it may require difficult computations for inferencing and parameter estimation. Implementing the proposed model in Hadoop MapReduce framework gives comparable performance when the dataset contains unstructured data. Moreover, the performance evaluations of the proposed model are carried out to analyze the ability of the proposed MapReduce CTM model and to compare the performance with another topic model. And finally, evaluations through the discussion of the proposed paper recommendation system are made with illustrations. According to the experiments, the proposed system yields better results which are semantically relevant and coherent to the user query at the top of the recommendations list in a reasonable amount of time. Especially, it increases the generation of relevant, reliable and useful recommendations with the incorporation of MapReduce CTM model.

CHAPTER 7

CONCLUSION AND FUTURE WORKS

As more and more digitized documents are spreading and scattering across many sources, it requires more efforts to summarize and analyze the whole digitized collection, and thus becoming more difficult to find the relevant information. In addition, the tremendous increase in the amount of electronically available scientific documents impels researchers to introduce an efficient and effective recommendation system in order to gather those documents and examine valuable data from these gathered documents. For this purpose, a research paper recommendation system is introduced that employs the Correlated Topic Model (CTM) with variational Expectation-Maximization (variational EM) algorithm in MapReduce framework is proposed to make meaningful recommendations of research papers.

The various empirical experiments on the crawled document collections indicate that inferring the hidden topics provides not only a better understanding of topics representation of the datasets but also a good explanation of the recommendations. From these experiments, the integration of the MapReduce CTM model in the paper recommendation system seems promising in recommending coherent and relevant papers. Moreover, the quality of inferred topics is improved by analyzing the full-texts of the documents.

Last but not least, the performance of the proposed MapReduce CTM model is evaluated in terms of UCI and UMass coherence measures of the extracted topics. The preliminary results show that the proposed model is proved to be comparable with the baseline topic model, the LDA. The comparison with the LDA model demonstrates the topic discovery capability of the MapReduce CTM model. Furthermore, it is demonstrated that the proposed paper recommendation system with predictability computation achieves at least good results in terms of accuracy and retrieval performance over all of the crawled document datasets.

7.1 Advantages and Limitations of the Proposed System

The proposed paper recommendation system gains several advantages over the quality, accuracy and efficiency of the generated recommendation outcomes that are

presented to the active users of the system. What is more, the performance of the system is effectively reflected by employing the CTM model with MapReduce implementation. With the extracted topics of the CTM model, the whole document collection can be summarized and categorized without human annotation effort. Besides, the proposed system can also be used for educational, commercial and governmental purposes with making more developments.

Mainly, the proposed system has provided several advantages. Firstly, the scalability problem of recommendation system is figured out because the offline computation for the recommendations is implemented in MapReduce framework that can perform parallel processing of large amounts of data in a distributed manner. Secondly, the utilization of trained MapReduce CTM model in the proposed paper recommendation system can be able to reduce the recommendation response time because it does not need to process the whole dataset every time when generating recommendations. Finally, the MapReduce CTM with variational EM algorithm increases the recommendation accuracy because it can be able to recommend unseen and latent articles to the user.

On the other hand, there are some limitations and constraints in the proposed system. The proposed approach requires proper memory requirements while the MapReduce CTM with variational EM algorithm is trained to learn the latent topics. In addition, the MapReduce CTM may encounter a challenge in calculating the posterior distribution of the latent variables and learning the model parameters while extracting the latent topics over the observed words. Besides, the model training time of the proposed system could be misleading since it depends heavily on the memory capacity and the computation algorithm itself.

7.2 Summary

The vast amount of digitized data available has led to the development of efficient research paper recommendation systems. Most of the proposed systems have applied Latent Dirichlet Allocation (LDA) and analyzed the titles and abstracts of the documents when generating recommendations for the active user. However, extracting and learning the hidden topics from the full-texts documents has become an important task for many topic model applications. Moreover, the conventional topic modeling

approaches suffer from the scalability problem when the size of the document collection increases.

In this proposed system, the MapReduce CTM model with variational EM is implemented in a Hadoop cluster leveraging MapReduce framework to extract the semantic correlations represented by latent topics and to recommend the most relevant documents that can satisfy the needs of users. The MapReduce framework and the HDFS of the Hadoop ecosystem have been utilized as an implementation platform for the proposed system to improve the performance of the proposed system.

First of all, the datasets for the system are crawled from the publicly available digital libraries. After the collection of datasets, the proposed system starts the offline processing component. The MapReduce CTM is leveraged to learn the semantic relationships between documents. Then, the results are fed into the paper recommendation system to calculate the similarity between the user query and the documents. The similarity calculation is performed by comparing the latent distributions of the two topics. After that, the predictability relationship is computed by applying entropy with the objective of improving not only the accuracy of similarity calculation but also the quality of top-N recommendations list.

When the offline processing finishes afterwards, the online computation component is performed to deliver recommendations by the application of GUI. Finally, the evaluations for both offline and online components are carried out to demonstrate the efficiency and effectiveness of the proposed system. The evaluation results established promising effects to exploit the MapReduce CTM model so that to yield better outcomes during recommendations generation. To conclude, applying the CTM model in MapReduce framework for the paper recommendation system can lead to a useful implementation to answer the needs and demands of the users.

7.3 Future Works

Through the empirical experiments that are conducted in the proposed approach, there are still many further works that are required to be done. In the future, the works will take into account the following research directions in order to improve the perceived quality of the paper recommendation system.

Firstly, the works will be emphasizing on increasing the size of documents collection and improving the performance of proposed paper recommendation system. Our previous work is only concerned in the investigation of recommendation accuracy for the unstructured digitized datasets. Hence, more insights from the larger dataset will be revealed with more parameter settings and more optimized Hadoop cluster configurations. Further technical works will be conducted in this area.

Secondly, the applicability of the proposed MapReduce CTM model to a larger distributed computing environment is an important issue to be settled. Further improvement is also needed to provide the guarantees for the scalability and stability of the topic model in achieving the optimal results. Also, for the usability of the topic model, further work is required to provide visual representation of the topics.

Thirdly, the proposed paper recommendation system is evaluated by using the PMI-score of the keywords, which may not effectively reflect the accuracy of paper recommendation system. Therefore, other additional accuracy evaluation approaches will be taken into account in order to conveniently match with the decisions and choices of the users.

Finally, the application of the MapReduce CTM model will be extended to other recommendation domains, such as Music or Movies, and will perhaps provide the reliable and incredible recommendation results. Additionally, further extensive comparisons are required to confirm the performance of the proposed system with other baseline recommendation approaches.

AUTHOR'S PUBLICATIONS

[P1] Mi Khine Oo, Myo Kay Khaing, "Science-related Articles Recommendation System from Big Data", in 14th International Conference on Computer Applications (ICCA 2016), Yangon, MYANMAR, pp. 184-189, February 2016.

[P2] Mi Khine Oo, May Aye Khine, "Correlated Topic Modeling for Big Data with MapReduce", in IEEE 7th Global Conference on Consumer Electronics (IEEE GCCE 2018), Nara, JAPAN, pp. 380-381, October 2018. ISBN: 978-1-5386-6310-3, ISSN: 2378-8143, DOI: 10.1109/GCCE.2018.8574696, #IEEE, SCI Conference.

[P3] Mi Khine Oo, May Aye Khine, "Topic Extraction of Crawled Documents Collection using Correlated Topic Model in MapReduce Framework", in International Journal on Natural Language Computing (IJNLC), Volume 8, Number 6, pp. 11-23, December 2019.

BIBLIOGRAPHY

[1] G. Adomavicius and A. Tuzhilin, "Toward the Next Generation of Recommender System: A Survey of the State-of-the-art and possible Extensions," IEEE Transactions on Knowledge and Data Engineering, 17(6): 734–749, 2005.

[2] Anshudeep, "Introduction to Hadoop Framework," [Online]. Available: https://netjs.blogspot.com/2018/02/introduction-to-hadoop-framework.html

[3] M. Aznag, M. Quafafou and Z. Jarir, "Correlated Topic Model for Web Services Ranking," International Journal of Advanced Computer Science and Applications, Vol. 4, No. 6, 2013.

[4] A. Bakshi, "Overview of Hadoop 2.0 Cluster Architecture Federation," May 2019, [Online]. Available: https://www.edureka.co/blog/overview-of-hadoop-2-0-cluster-architecture-federation/

[5] M. Barnwal, "Types of data in recommender systems," September 27, 2018, [Online]. Available: http://manishbarnwal.com/blog/2018/09/27/types_data_recommender_system/

[6] N. Bassiou and C. Kotropoulos, "RPLSA: A Novel Updating Scheme for Probabilistic Latent Semantic Analysis," Department of Informatics, Aristotle University of Thessaloniki, Box 451 Thessaloniki 541 24, Greece, Received 14 April 2010.

[7] J. Beel, B. Gipp, S. Langer and C. Breitinger, "Research-paper Recommender Systems: a Literature Survey," International Journal on Digital Libraries, pp. 1–34, 2015.

[8] C. Bentz, D. Alikaniotis, M. Cysouw and R. Ferrer-i-Cancho, "The Entropy of Words – Learnability and Expressivity across More than 1000 Languages," Entropy 2017, 19(6), 275, June 2017.

[9] Big Data Zone, "How Hadoop Map/Reduce Works," [Online]. Available: https://dzone.com/articles/how-hadoop-mapreduce-works

[10] D. Blei and J. D. Lafferty, “A Correlated Topic Model of Science,” The Annals of Applied Statistics, Vol. 1, No. 1, pp. 17–35, 2007.

[11] D. M. Blei and J. D. Lafferty, “Correlated Topic Models,” Advances in Neural Information Processing Systems 18 (Y.Weiss, B. Scholkopf and J. Platt, eds.), MIT Press, Cambridge, MA, 2006.

[12] D. M. Blei, A. Y. NG and M. I. Jordan, “Latent Dirichlet Allocation,” Journal of machine Learning research, 3, pp. 993–1022, 2003.

[13] D. K. Bokde, S. Girase and D. Mukhopadhyay, “Role of Matrix Factorization Model in Collaborative Filtering Algorithm: A Survey,” International Journal of Advance Foundation and Research in Computer (IJAFRC), Volume 1, Issue 6, May 2014.

[14] J. S. Breese, D. Heckerman and C. Kadie, “Empirical Analysis of Predictive Algorithms for Collaborative Filtering,” Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, UAI’98, Madison, Wisconsin: Morgan Kaufmann Publishers Inc., pp. 43–52, 1998.

[15] R. Burke, “Hybrid Web Recommender Systems,” The adaptive web, LNCS 4321, Berlin Heidelberg, Germany, Springer, pp. 377–408, 2007.

[16] H. Chandrashekhar and B. Bhasker, “Personalized Recommender System using Entropy based Collaborative Filtering Technique,” Journal of Electronic Commerce Research, Vol. 12, No. 3, 2011.

[17] S. Chen and Y. Peng, “Matrix Factorization for Recommendation with Explicit and Implicit Feedback,” Knowledge-based Systems, 2018.

[18] P. Cunningham, R. Bergmann, S. Schmitt, R. Traphoner, S. Breen and B. Smyth, “WebSell: Intelligent Sales Assistants for the World Wide Web,” Proceedings CBR in ECommerce, Vancouver BC, pp. 104–9, 2001.

[19] DataFlair, “Hadoop HDFS Architecture Explanation and Assumptions,” [Online].
Available: https://data-flair.training/blogs/hadoop-hdfs-architecture/

[20] P. Dave, “Big Data – What is Big Data – 3 Vs of Big Data – Volume, Velocity and Variety,” October 2, 2013, [Online].
Available: https://blog.sqlauthority.com/2013/10/02/big-data-what-is-big-data-3-vs-of-big-data-volume-velocity-and-variety-day-2-of-21/

[21] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer and R. A. Harshman, “Indexing by Latent Semantic Analysis,” Journal of the Society for Information Science, 41(6), pp. 391–407, 1990.

[22] E. Dumbill, “What is Big data?” January 2012, [Online]. Available: http://strata.oreilly.com/2012/01/what-is-big-data.html

[23] W. Fan and A. Bifet, “Mining Big Data: Current Status, and Forecast to the Future,” SIGKDD Explorations, Volume 14, Issue 2, 2013.

[24] G. E. Forsythe, M. A. Malcolm and C. B. Moler, “Computer Methods for Mathematical Computations,” Chapter 9: Least Squares and the Singular Value Decomposition, Englewood Cliffs, NJ: Prentice Hall, 1977.

[25] M. A. Ghazantar and A. Pragel-Benett, “A Scalable Accurate Hybrid Recommender System,” The 3rd International conference on knowledge discovery and data mining (WKDD 2010), IEEE Computer Society, Washington, DC, USA, 2010.

[26] M. R. Ghazia and D. Gangodkar, “Hadoop, MapReduce and HDFS: a Developers Perspective,” International Conference on Intelligent Computing, Communication & Convergence, 2015.

[27] B. Gipp, J. Beel and C. Hentschel, “Scienstein: A Research Paper Recommender System,” Proceedings of the International Conference on Emerging Trends in Computing (ICETiC’09), Virudhunagar, India, 2009.

[28] D. Goldberg, D. Nichols, B. M. Oki and D. Terry, “Using Collaborative Filtering to Weave an Information Tapestry,” Commun. ACM, 35(12):61–70, December 1992.

[29] K. Goldberg, T. Roeder, D. Gupta and C. Perkins, “Eigentaste: A Constant Time Collaborative Filtering Algorithm,” Information Retrieval J., vol. 4, no. 2, pp. 133–151, July 2001.

[30] C. A. Gomez-Uribe and N. Hunt, "The Netflix Recommender System: Algorithms, Business Value, and Innovation," ACM Transactions on Management Information Systems, 6(4): 1–19, December 2015.

[31] N. Good, J. Schafer, et al, “Combining Collaborative Filtering with Personal Agents for Better Recommendations,” Proceedings of the 16th National Conference on Artificial Intelligence, 439–446, 1999.

[32] K. Haruna, M. A. Ismail, D. Damiasih, J. Sutopo and T. Herawan, “A Collaborative Approach for Research Paper Recommender System,” October 5, 2017.

[33] H. A. M. Hassan, “Personalized Research Paper Recommendation using Deep Learning,” UMAP 2017 Doctoral Consortium, Bratislava, Slovakia, July 2017.

[34] W. Hill, L. Stead, M. Rosenstein and G. Furnas, “Recommending and Evaluating Choices in a Virtual Community of Use,” Proceedings of ACM CHI ’95, 194–201. Denver, CO.:ACM, 1995.

[35] T. T. Hoang and P. T. Nguyen, “Word Sense Induction using Correlated Topic Model,” International Conference on Asian Language Processing, 2012.

[36] T. Hofmann, “Probabilistic Latent Semantic Analysis,” Uncertainty in Artificial Intelligence (UAI'99), Stockholm, pp. 289–296, 1999.

[37] Z. Huang, W. Chung, et al, “A Graph-based Recommender System for Digital Library,” Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries (Portland, Ore.), ACM, New York, 65–73, 2002.

[38] F. O. Isinkaye, Y. O. Folajimi and B. A. Ojokoh, “Recommendation Systems: Principles, Methods and Evaluation,” Egyptian Informatics Journal, 2015.

[39] D. Jannach, L. Lerche and M. Zanker, “Recommending based on Implicit Feedback.”

[40] H. Jelodar, Y. Wang, C. Yuan and X. Feng, “Latent Dirichlet Allocation (LDA) and Topic Modeling: Models, Applications, a Survey.”

[41] S. Khusro, Z. Ali and I. Ullah, “Recommender Systems: Issues, Challenges, and Research Opportunities,” Information Science and Applications (ICISA), Lecture Notes in Electrical Engineering 376, 2016.

[42] S. W. Kim and J. M. Gil, “Research Paper Classification Systems based on TF-IDF and LDA Schemes,” Human-centric Computing and Information Sciences, 2019.

[43] J. H. Kim, U. G. Kang and J. H. Lee, “Content-based Filtering for Music Recommendation based on Ubiquitous Computing,” IFIP International Federation for Information Processing, vol 228. Springer, Boston, MA, IIP 2006.

[44] J. A. Konstan, B. N. Miller, et al, “GroupLens: Applying Collaborative Filtering to Usenet News,” Comm ACM, 40(3):77–87, 1997.

[45] Y. Koren, R. Bell and C. Volinsky, “Matrix Factorization Techniques for Recommender Systems,” Computer 42, 8(2009), 2009.

[46] B. Kumar and N. Sharma, “Approaches, Issues and Challenges in Recommender Systems: A Systematic Review,” Indian Journal of Science and Technology, Vol 9(47), December 2016.

[47] H. J. Kwon, T. H. Lee and K. S. Hong, “Improving Memory-based Collaborative Filtering using Entropy-based Similarity Measures,” Proceedings of International Symposium on Web Information Systems and Applications (WISA’09), 2009.

[48] H. J. Kwon, T. H. Lee, J. H. Kim and K. S. Hong, “Improving Prediction Accuracy using Entropy Weighting in Collaborative Filtering,” Symposia and Workshops on Ubiquitous, Automatic and Trusted Computing, 2009.

[49] S. Lee, “Entropy-weighted Similarity Measures for Collaborative Recommender Systems,” AIP Conference Proceedings 1982, pp. 020011-1–020011-6, 2018.

[50] J. Lee, K. Lee and J. G. Kim, “Personalized Academic Research Paper Recommendation System,” arXiv:1304.5457v1 [cs.IR], April 2013.

[51] G. Li and Q. Chen, “Exploiting Explicit and Implicit Feedback for Personalized Ranking,” Mathematical Problems in Engineering, Volume 2016.

[52] Y. Li, M. Yang and Z. M. Zhang, “Scientific Articles Recommendation,” Proceedings of the 22nd ACM International Conference on Information & Knowledge Management, ACM, pp. 1147–1156, 2013.

[53] G. Linden, B. Smith and J. York, “Amazon.com Recommendations: Item-to-Item Collaborative Filtering,” IEEE Internet Computing, 7(1):76–80, 2003.

[54] L. Liu, L. Tang, W. Dong, S. Yao and W. Zhou, “An Overview of Topic Modeling and its Current Applications in Bioinformatics,” SpringerPlus, 5(1), 1608, September 20, 2016.

[55] S. Liu, X. Tu and R. Li, “Unifying Explicit and Implicit Feedback for Top-N Recommendation,” 2017 IEEE 2nd International Conference on Big Data Analysis, 2017.

[56] P. Lops, M. d. Gemmis and G. Semeraro, “Content-based Recommender Systems: State of the Art and Trends,” Recommender Systems Handbook, LLC, 2011.

[57] T. Luostarinen and O. Kohonen, “Using Topic Models in Content-Based News Recommender Systems,” Proceedings of the 19th Nordic Conference of Computational Linguistics (NODALIDA 2013), Linköping Electronic Conference Proceedings, pp. 239–251, 2013.

[58] K. Ma, “Content-based Recommender System for Movie Website,” Master’s Thesis at VionLabs, 2016.

[59] B. Madhushree, “A Novel Research Paper Recommendation System,” International Journal of Advanced Research in Engineering and Technology (IJARET), Volume 7, Issue 1, pp. 07–16, 2016.

[60] B. Marr, “Big Data: the 5 Vs Everyone must know,” March 2014, [Online]. Available: https://www.linkedin.com/pulse/20140306073407-64875646-big-data-the-5-vs-everyone-must-know.html

[61] S. M. McNee, I. Albert, D. Cosley, P. Gopalkrishnan, S. K. Lam, A. M. Rashid, J. A. Konstan and J. Riedl, “On the Recommending of Citations for Research Papers,” Proceedings of the 2002 ACM conference on Computer Supported Cooperative Work, CSCW’02, pp. 116–125, New York, USA, 2002.

[62] E. McNulty, “Understanding Big Data: the Seven V’s,” May 22, 2014, [Online].
Available: https://dataconomy.com/2014/05/seven-vs-big-data/

[63] H. Mehta, S. K. Bhatia, P. Bedi and V. S. Dixit, “Collaborative Personalized Web Recommender System using Entropy based Similarity Measure,” IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 6, No 3, November 2011.

[64] R. Meteren and M. Someren, “Using Content-based Filtering for Recommendation,” Proceedings of ECML 2000 Workshop: Machine Learning in Information Age, pp. 47–56, 2000.

[65] D. Mimno, H. Wallach, E. Talley, M. Leenders and A. McCallum, “Optimizing semantic coherence in topic models,” Proceedings of the 2011 Conference on Emperical Methods in Natural Language Processing, pp. 262–272, Edinburgh, Scotland, UK, 2011.

[66] M. Montaner, B. Lopez and J. L. D. L. Rosa, “A Taxonomy of Recommender Agents on the Internet,” Artificial Intelligence Review, 19(4), pp. 285–330, 2003.

[67] R. J. Mooney and L. Roy, “Content-based Book Recommending using Learning for Text Categorization,” Proceedings of the SIGIR-99 Workshop on Recommender Systems: Algorithms and Evaluation, Berkeley, CA, August 1999.

[68] N. K. Nagwani, “Summarizing Large Text Collection using Topic Modeling and Clustering based on MapReduce Framework,” Journal of Big Data 2, 6, 2015.

[69] C. Nascimento, A. H. Laender, A. S. da Silva and M. A. Gonçalves, “A Source Independent Framework for Research Paper Recommendation,” Proceedings of the 11th Annual International ACM/IEEE Joint Conference on Digital Libraries, pp. 297–306. ACM, 2011.

[70] D. Newman, J. H. Lau, K. Grieser and T. Baldwin, “Automatic Evaluation of Topic Coherence,” Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, HLT ’10, pp. 100–108, Stroudsburg, PA, USA, Association for Computational Linguistics, 2010.

[71] D. Newman, Y. Noh, E. Talley, S. Karimi and T. Baldwin, “Evaluating topic models for digital libraries,” Proceedings of the 10th annual joint conference on Digital libraries, JCDL ’10, pp. 215–224, New York, USA, 2010.

[72] B. S. Oladapo, “A Research Paper Recommender System,” 2013 unpublished.

[73] R. P. Padhy, “Big Data Processing with Hadoop-MapReduce in Cloud Systems,” International Journal of Cloud Computing and Services Science, Vol.2, No.1, pp. 16–27, February 2013.

[74] M. J. Pazzani and D. Billsus, “Content-based Recommendation Systems,” The Adaptive Web, 2007.

[75] S. Philip, P. B. Shola and A. O. John, “Application of Content-based Approach in a Research Paper Recommendation System for a Digital Library,” International Journal of Advanced Computer Science and Application, Volume 5, Issue 10, 2014.

[76] S. Philip, P. B. Shola and E. P. Musa, “A Paper Recommender System based on the Past Ratings of a User,” International Journal of Advanced Computer Technology (IJACT), 2014.

[77] C. Piao, J. Zhao and J. Feng, “Research on Entropy-based Collaborative Filtering Algorithm,” IEEE International Conference on e-Business Engineering, 2007.

[78] C. Pinela, “Recommender Systems – User-based and Item-based Collaborative Filtering,” November 6, 2017, [Online].
Available: https://medium.com/@cfpinela/recommender-systems-user-based-and-item-based-collaborative-filtering-5d5f375a127f

[79] Public Library of Science, “PLOS ONE,” [Online].
Available: https://journals.plos.org/plosone/

[80] F. Ricci, L. Rokach, B. Shapira and P. B. Kantor, “Recommender Systems Handbook,” Springer, Berlin, 2011.

[81] B. Rocca, “Introduction to recommender systems: Overview of some major recommendation algorithms,” June 3, 2019, [Online].
Available: https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada

[82] M. Roder, A. Both and A. Hinneburg, “Exploring the Space of Topic Coherence Measures,” WSDM’15, Shanghai, China, February 2–6, 2015.

[83] I. Rose, R. Murty, P. R. Pietzuch, J. Ledlie, M. Roussopoulos and M. Welsh, “Cobra: Content-based Filtering and Aggregation of Blogs and RSS Feeds,” NSDI, 2007.

[84] R. Sang and K. P. Chan, “A Correlated Topic Modeling Approach for Facial Expression Recognition,” CIT/IUCC/DASC/PICom, 2015.

[85] B. Sarwar, G. Karypis, et al, “Application of Dimensionality Reduction in Recommender Systems: A Case Study,” Proceedings of the WebKDD Workshop at the ACM SIGKKD, ACM, New York, 2000.

[86] B. Sarwar, G. Karypis, et al, “Item-based Collaborative Filtering Recommendation Algorithms,” Proceedings of the 10th International World Wide Web Conference, 285–295, 2001.

[87] C. E. Shannon, “A Mathematical Theory of Communication,” Bell System Technical Journal, 27(3), pp. 379–423, 1948.

[88] D. Sharda and P. Dawgotra, “Design of Research Buddy: Personalized Research Paper Recommendation System,” International Journal of Advance Research in Science and Engineering, Volume 6, Issue 9, September 2017.

[89] U. Shardanand and P. Maes, “Social Information Filtering: Algorithms for Automating “Word of Mouth,” Proceedings of ACM CHI ’95, Denver, CO.: ACM, 1995.

[90] N. Singh, N. Garg and V. Mittal, “Big Data – Insights, Motivation and Challenges”, International Journal of Scientific & Engineering Research, Volume 4, Issue 12, December 2013.

[91] B. Srebrenik, “Introduction to Music Recommendation and Machine Learning,” December 2018, [Online].
Available: https://medium.com/@briansrebrenik/introduction-to-music-recommendation-and-machine-learning-310c4841b01d

[92] T. Sripadh and G. Ramesh, “Personalized Research Paper Recommender System,” D. J. Hemanth and S. Smys (eds.), Computational Vision and Bio Inspired Computing, Lecture Notes in Computational Vision and Biomechanics 28, 2018.

[93] M. Steyvers, “Probabilistic Topic Models,” T. Landauer, D McNamara, S. Dennis, and W. Kintsch (eds), Latent Semantic Analysis: A Road to Meaning, Laurence Erlbaum.

[94] T. Strohman, W. B. Croft and D. Jensen, “Recommending citations for Academic Papers,” Proceedings of the 30th annual international ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR’07, pp. 705–706, New York, USA, 2007.

[95] X. Su and T. M. Khoshgoftaar, “A Survey of Collaborative Filtering Techniques,” Advances in Artificial Intelligence, 2009.

[96] J. W. Tao and P. F. Ding, “CTMIR: A Novel Correlated Topic Model for Image Retrieval,” Second International Workshop on Knowledge Discovery and Data Mining, pp. 948–951, Jan 2009.

[97] TechAmerica Foundation’s Federal Big Data Commission, “Demystifying Big data: A practical guide to transforming the business of Government,” 2012.

[98] L. Terveen, W. Hill, B. Amento, D. McDonald and J. Creter, “PHOAKS: A System for Sharing Recommendations,” Comm. ACM, vol. 40, no. 3, pp. 59–62, 1997.

[99] The Apache Software Foundation, “Apache Hadoop,” [Online].
Available: https://hadoop.apache.org/

[100] The Apache Software Foundation, “Apache PDFBox,” [Online]. Available: https://pdfbox.apache.org/

[101] The College of Information Sciences and Technology, “CiteSeerX,” [Online]. Available: https://citeseerx.ist.psu.edu/index

[102] The dblp computer science bibliography, “Statistics - Publications Per Year,” September 19, 2019, [Online]. Available: https://dblp.uni-trier.de/statistics/publicationsperyear

[103] P. B. Thorat, R. M. Goudar and S. Barve, “Survey on Collaborative Filtering, Content-based Filtering and Hybrid Recommendation System,” International Journal of Computer Applications, Volume 110 – No. 4, January 2015.

[104] R. Torres, S. M. McNee, M. Abel, J. A. Konstan and J. Riedl, “Enhancing Digital Libraries with TechLens,” Joint ACM/IEEE Conference on Digital Libraries (JCDL), pp. 228–236, 2004.

[105] S. Vinodhini, V. Rajalakshmi and B. Govindarajalu, “Building Personalised Recommendation System with Big Data and Hadoop Mapreduce,” International Journal of Engineering Research and Technology (IJERT), Volume 3, Issue 4, April 2014.

[106] C. Wang and D. M. Blei, “Collaborative Topic Modeling for Recommending Scientific Articles,” KDD’11, San Diego, California, USA, August 2011.

[107] W. Wang, G. Zhang and J. Lu, “Collaborative Filtering with Entropy-Driven User Similarity in Recommender Systems,” International Journal of Intelligent Systems, Vol. 30, pp. 854–870, 2015.

[108] Wikipedia, the free encyclopedia, “Information Entropy,” January 2007, [Online]. Available: http://en.wikipedia.org/wiki/Information_entropy

[109] C. Wu, R. Buyya and K. Ramamohanarao, “Big Data: Principles and Paradigms,” Morgan Kauffman, 2016.

[110] H. Wu, Y. Wang and X. Cheng, “Incremental Probabilistic Latent Semantic Analysis for Automatic Question Recommendation,” ACM New York, NY, USA, pp. 99–106, 2008.

[111] Y. Xia, G. D. Fabbrizio, S. Vaibhav and A. Datta, “A Content-based Recommender System for E-commerce Offers and Coupons,” Proceedings of SIGIR eCom 2017, Tokyo, Japan, August 2017.

[112] X. Xu, A. Shimada and R.Taniguchi, “Correlated Topic Model for Image Annotation,” The 19th Korea-Japan Joint Workshop on Frontiers of Computer Vision, 2013.

[113] Guangxu Xun, Yaliang Li, Wayne Xin Zhao, Jing Gao and Aidong Zhang, “A Correlated Topic Model using Word Embeddings,” Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17), pp. 4207–4213, 2017.

[114] K. Zhai, J.L. Boyd-Graber, N. Asadi and M.L. Alkhouja, “Mr. LDA: A Flexible Large Scale Topic Modeling Package using Variational Inference in MapReduce,” Proceedings of the 21st Int. Conf. on World Wide Web, pp. 879–888, 2012.

[115] F. Zhang, H. Liu and Y. Cui, “Rating Information Entropy for Cold-start Recommendation,” Journal of Information & Computational Science 8: 1 (2011), pp. 16–22, 2011.

[116] P. Zikopoulos and C. Eaton, “Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data,” McGraw-Hill Osborne Media, 2011.

ACRONYMS

CBF	Content-based Filtering
CF	Collaborative Filtering
Cobra	Content-based RSS Aggregator
CTM	Correlated Topic Model
DN	DataNode
EM	Expectation-Maximization
GUI	Graphical User Interface
HDFS	Hadoop Distributed File System
JHS	JobHistoryServer
LDA	Latent Dirichlet Allocation
LSA	Latent Semantic Analysis
MAE	Means Absolute Error
NM	NodeManager
NN	NameNode
NS	Namespace
PLSA	Probabilistic Latent Semantic Analysis
RM	ResourceManager
SN	SecondaryNameNode
YARN	Yet Another Resource Negotiator