WEB API RECOMMENDATION BASED ON TOPIC MODELLING AND CLUSTERING

A THESIS SUBMITTED TO AUCKLAND UNIVERSITY OF TECHNOLOGY
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF COMPUTER AND INFORMATION SCIENCES

Supervisor

Dr. Jian Yu

Dr. Sira Yongchareon

July 2019

By

Fangran Zhang

School of Engineering, Computer and Mathematical Sciences

Copyright

Declaration

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the qualification of any other degree or diploma of a university or other institution of higher learning.

Signature of candidate

Acknowledgements

I would like to thank my supervisor Dr. Jian Yu, and the cast of thousands who supported me in my journey of discovery.

Abstract

Nowadays, Recommender Systems are widely used in various web portals, while service discovery is still a great challenge for better integrating appropriate services into business scenarios. Gaining insight of the development of recommender systems is helpful for tackling the issues. This thesis proposes a recommender system framework to achieve Web APIs recommendation based on the collected data of Web API directory: $ProgrammableWeb.com$. We intend to build a comprehensive Recommender system by combining the method of collaborative filtering recommendation with topic modeling in natural language processing.

Specifically, we find that the collaborative filtering algorithms as the mainstream recommendation method is affected by the cold start problem, and the different kinds of recommendation algorithms lack systematic comparison in existing works. Therefore, we integrated topic vector and then document clustering to solved the cold start problem, and then implement the representative collaborative filtering algorithms. The various evaluation methods are used to evaluate the ranking quality and diversity of Web APIs recommendations.

We discover that the topic vector extraction of LDA algorithm with the K-means algorithm combines the Agnes algorithm is able to achieve satisfying document clustering results, and the collaborative filtering algorithm has good recommendation performance by considering invoke relationship, content similarity, and latent semantic mining.

Keywords: Recommender system, topic model, collaborative filtering, LDA, PMF

Contents

Copyright **2**

Declaration **3**

Acknowledgements **4**

Abstract **5**

1 Introduction **11**
1.1 Motivation 11
1.2 Background 13
1.2.1 Collaborative Filtering based Recommendation Algorithm . . 13
1.2.2 Topic Modelling 15
1.3 Research Questions 17
1.4 Contribution 18
1.5 Thesis Structure 19

2 Literature Review **21**
2.1 Introduction 21
2.2 Recommender System 22
2.2.1 Evolution of Recommender System 22
2.2.2 The Categories of Recommender System 23
2.3 Topic Modelling 29
2.3.1 Latent Sematic Analysis (LSA) 29
2.3.2 Probabilistic Latent Semantic Analysis (PLSA) 30
2.3.3 Latent Dirichlet Allocation (LDA) 32
2.3.4 Latent Factor Model 40
2.4 Clustering Algorithm 41
2.4.1 Partition-based Clustering Algorithm 42
2.4.2 Statistics-based Clustering Algorithm 44
2.4.3 Development Tend of Clustering Algorithm 45
2.5 Conclusion 45

3 Research Method **46**
3.1 Introduction . . . 46
3.2 Research Designing . . . 46
3.3 Data Sources and Data Collection . . . 48
3.4 Pre-processing of Data Set . . . 49
3.4.1 LDA Section . . . 51
3.4.2 TF-IDF Section . . . 52
3.5 Topic Modelling . . . 52
3.5.1 Two-Dimension Array of LDA . . . 52
3.5.2 LDA with Gibbs Sampling . . . 55
3.5.3 Two Level LDA Model . . . 57
3.5.4 TF-IDF Model . . . 58
3.6 Mashup Service Clustering . . . 60
3.6.1 Jensen-Shannon Divergence . . . 60
3.6.2 K-means Clustering Algorithm . . . 62
3.6.3 Agnes Clustering Algorithm . . . 62
3.7 Web APIs Recommendation . . . 63
3.7.1 Web APIs Recommendation based on CF Algorithm . . . 63
3.7.2 Web APIs Recommendation based on Popularity . . . 65
3.7.3 Web APIs Recommendation based on Matrix Factorization . . . 65
3.8 Evaluation Metrics . . . 67

4 Results **69**
4.1 Introduction . . . 69
4.2 Mashup Service Clustering . . . 70
4.2.1 Topic Modelling . . . 70
4.2.2 Cluster Results . . . 73
4.3 Problem Definition of Recommendation . . . 76
4.3.1 User-item Matrix . . . 76
4.3.2 Define User Preferences . . . 77
4.4 Web APIs Recommendation . . . 78
4.4.1 User-item based Recommendation Model . . . 79
4.4.2 Popularity based Recommendation Model . . . 82
4.4.3 User-item with Popularity based Recommendation Model . . . 85
4.4.4 PMF based Recommendation Model . . . 89
4.5 Conclusion . . . 92

5 Discussion **93**
5.1 Introduction . . . 93
5.2 Topic Vector Extraction . . . 94
5.3 Recommendation Model Comparison . . . 96
5.3.1 DCG Assessment . . . 96
5.3.2 HMD Assessment . . . 99
5.4 Contributions . . . 102

6 Conclusions **105**
6.1 Limitations 105
6.2 Further Research 108

References **110**

Appendices **117**

List of Tables

3.1 The Distribution of Mahsup Services in Top 20 Categories 50

4.1 The full report of LDA result 70
4.2 The topic distribution probability of words in topic 0 71
4.3 The Overall of the Topic Distribution Probability 72
4.4 The overall of JS based Cluster with LDA model 76
4.5 An Example of Matrix R 77
4.6 The DCG result of User-item based Recommendation 80
4.7 The HMD result of User-item based Recommendation 81
4.8 The DCG result of Popularity based Recommendation 83
4.9 The HMD result of Popularity based Recommendation 84
4.10 The DCG result of User-item with Popularity based Recommendation 87
4.11 The HMD result of User-item with Popularity based Recommendatio . 88
4.12 The DCG result of PMF based Recommendation 90
4.13 The HMD result of PMF based Recommendation 91

5.1 The DCG comparison results 97
5.2 The HMD comparison results 100

List of Figures

2.1 The Flow Chart of Recommender System Basic Framework 24
2.2 The Diagram of Uni-gram Model 36
2.3 The diagram of Mixture Uni-gram Model 37
2.4 The Diagram of PLSA Model 38
2.5 The Diagram of LDA Model 38

3.1 Flow Chart of Recommender System 47
3.2 An example of Mashup service 48
3.3 The flow chart of data pre-processing 50
3.4 The flow chart of Two Level LDA 58

4.1 An Example of Mashup Service Document Latent Probability Distribution 73
4.2 An example of clustering result 74
4.3 The chart of DCG result for User-item based recommendation 80
4.4 The chart of HMD result for User-item based recommendation 81
4.5 The chart of DCG result for Popularity based recommendation 83
4.6 The chart of HMD result for Popularity based recommendation 84
4.7 The chart of DCG result for User-item with Popularity based recommendation 87
4.8 The chart of HMD result for User-item with Popularity based recommendation 88
4.9 The chart of DCG result for PMF based recommendation 90
4.10 The chart of HMD result for PMF based recommendation 91

5.1 The stacked line curve chart of the DCG comparison results 97
5.2 The chart of HMD result for PMF based recommendation 100

Chapter 1

Introduction

1.1 Motivation

With the remarkable developments in Internet technologies and rise of cloud and IoT services, a rapid increase in the number of Web services (a.k.a. Web APIs) is becoming a phenomenon. At the same time, it is more and more difficult for users to get the information they need in time. On the one hand, users are confronted with a large amount of information which is difficult to screen and often lost in a large amount of information space; on the other hand, website provider also lost contact with users, and not be able to effectively establish stable and solid cooperation with users (Ricci, Rokach & Shapira, 2015). Among Internet service, the Web service as a remote invoking technology across programming languages and operating system platforms (Kun, Xiao-Ling & Ao-Ying, 2004), has emerged explosive growth with development of the Internet due to its functional advantages. How to effectively discover suitable Web services is one of the core issue to be solved in the field of service-oriented computing. The recommender system came into being in this context, which be able to recommend objects that meet users' interests and preferences according to users' usage history information or online synchronization information. In additional, clustering similar Web services is also

an effective method of service discovery and service management (Z. Zhou, Sellami, Gaaloul, Barhamgi & Defude, 2013). Thus, how to build a comprehensive recommender system, which considers not only user preferences but also content similarity, is the core research orientation of this thesis.

With the increasing number of Web services and the categories of service functions on the Internet, it is becoming more and more difficult and time-consuming to accurately locate Web services that satisfy users' specific business requires from a large number of Web services sets with difficult functional attributes to define. Todays, most Web services exist in the form of Web API (Application Programming Interface), which are the basic building blocks of software. Developers are able to choose an Web API set online and combine them to form a new Web service, which are often called Mashup (Alonso, Casati, Kuno & Machiraju, 2004). In recent years, a large number of Mashups and Web APIs repositories have emerged, the most famous of repositories are ProgrammableWeb, myExperiment and Biocatalogue (Xia, Fan, Wu, Bai & Zhang, 2017). According to statistics, ProgrammableWeb is currently the largest and most active Web server publishing and sharing platform, on which dozens of new Web APIs are generated every day (TIAN, KOCHHAR & LO, n.d.). How to assist users to discover appropriate Web services effectively, and consider the popularity of Web services, as well as new and unpopular Web services, is the core research of this project.

This thesis makes an in-depth study on the acquisition and modeling of Web service information, the research of recommendation algorithm, and the evaluation of recommender system. Aim of not only to help users discover the Web APIs they need, but also to establish the relationship between the website and users through the recommender system so as to effectively retain the users, and ultimately to improve the clicks and loyalty of websites.

1.2 Background

Recommender systems are important methods of information retrieval on the Internet. The rise of recommender systems are closely related to the development of the Internet. Recommender systems are able to be divided into content-based recommendation algorithm and collaborative filtering based recommendation algorithm (Gharia, Desai & Gandhi, 2018). Content-based recommendation algorithms are mostly based on topic models, while collaborative filtering based recommendation algorithms are mostly based on similarity degree based on data set between user and item. However, the content-based recommendation algorithm and the collaborative filtering based recommendation algorithm have their own advantages and disadvantage, so how to combine the topic modelling with the collaborative filtering algorithm has important search significance.

In 2005, the review paper by Adomavicius et al. divided the recommender systems into three main categories: content-based recommendations, collaborative filtering based recommendations, and hybrid based recommendations, which caused significant influence in the field of recommender systems (Adomavicius & Tuzhilin, 2005). In recent years, the recommendation algorithm based on collaborative filtering has developed rapidly, while the content-based recommendation algorithm and hybrid based recommendation algorithm have developed relatively slowly. Thus, this thesis argues that combining collaborative filtering based recommendation algorithm with content-based recommendation algorithm has an important research significance.

1.2.1 Collaborative Filtering based Recommendation Algorithm

The technical idea of collaborative filtering algorithm is not complicated, while it is the most widely used technologies in recommender systems. After more than 20 years of development, collaborative fitering based recommendation algorithms have become more sophisticated. The earliest collaborative filtering technology can be traced back to

1992, when Xerox's search center first used collaborative filtering technology to solve the problem of overload of email information among employees (Goldberg, Nichols, Oki & Terry, 1992). The GroupLens research team first used collaborative filtering technology for news recommendation in 1994, which not only proposed the concept of collaborative filtering for the first time, but also established a formalized system for the recommender system (Konstan et al., 1997). The GroupLens algorithm is a milestone of the collaborative.

With the development of collaborative filtering, many different technical idea have been proposed. Among them, the users and items based collaborative filtering and matrix factorization based collaborative filtering are the most widely used collaborative filtering methods.

The users and items based collaborative filtering is the most traditional collaborative filtering method. In 1998, Amazon launched a collaborative filtering algorithm based on items and users, which was applied to serve more than ten millions of users and handle more than ten millions of items, and produced high quality recommendation results (Smith & Linden, 2017).

The milestone matrix factorization recommendation algorithm is Funk-SVD. The Funk-SVD proposed by Simon Funk in 2006, which shines brilliantly on Netfix Prize and has begun to be widely used in the field of recommender system (Simon, 2006). In 2008, Probabilistic Matrix Factorization (PMF) was proposed by Salakhutdinov et al. as a probabilistic interpretation version of FunkSVD (Mnih & Salakhutdinov, 2008). Bias SVD was proposed by Koren et al. in 2009, which is a variant version of Funk-SVD (Takimoto & Hirose, 2009). In 2010, SVD++ was proposed by Koren, which focuses on latent data mining (Jia, Zhang, Lu & Wang, 2014). In the same year, Koren et al. put forward the time-SVD algorithm, which adds interest attenuation process on the basis of SVD++ (Koren, 2009).

With the development of deep learning and multi-layer neural network, the combination of collaborative filtering recommendation algorithm and deep learning has become a hot research trend. Matrix factorization has good scalability, so it be able to be easily combined with deep learning. Kim et al. proposed Conv-MF algorithm in 2016, which comines matrix factorization with convolutiona neural network perfectly (D. Kim, Park, Oh, Lee & Yu, 2016).

However, although collaborative filtering develops rapidly and is widely used, the issues of data sparseness and cold-start still not able to solve fundamentally. Thus, the combination of external information is an effective way to solve these problem. The clustering of topic model is one of the effective ways to achieve it.

1.2.2 Topic Modelling

Topic modelling is a typical type of content-based recommendation model, which mainly consists in building user and item corpus data set, in order to calculate the similarity between users and items. Once the corpus of users and items are established, the similarity be able to be calculated directly, or they also can be put into the recommendation model of collaborative filtering. The advantages of building content-based recommendation model can be divided into two points: 1) as long as the corpus of items and users are obtained, the cold-start problem can be perfectly handled; 2) since the corpus of items and users are explicit features, so the training model has good interpret-ability (Gharia et al., 2018).

In natural language processing, a very important process is word embedding, that is the process of converting words into vectors, and topic modelling is essentially a method of converting words into vectors (Crossno, Wilson, Shead & Dunlavy, 2011). Topic modelling is used for document modeling, which converts documents into numerical vectors, and each dimension of numerical vectors corresponds to a topic. Therefore, the

topic modelling is proposed to achieve the structured corpus in essence, and structured corpus can be compared and retrieved with each other. The topic modelling is a generalized concept, but it generally default refers to the classical Latent Dirichlet Allocation (LDA) model (Sadamitsu, Mishina & Yamamoto, 2007).

As early as 1880s, researchers often adopted term frequency–inverse document frequency (TF-IDF) to retrieve documents and extract information. While TF-IDF simply believes that the more important the word with low text a low text frequency, the less useful the word with a high text frequency, which obviously not entirely correct (Ramos et al., 2003). It not be able to effectively reflect the importance of words and the distribution of topics, so the accuracy is limited. Latent semantic analysis (LSA) goes further, aiming to discover implicit topic features from the corpus, which provides a good theoretical basis for topic model and matrix factorization (Landauer, Foltz & Laham, 1998). While, LSA model is not able to deal with polysemy problem. Hofmann proposed probabilistic latent semantic analysis (PLSA) model in 1999, which is the real topic model (Hofmann, 1999). The PLSA model is based on probability theory, and the number of parameters and the corpus are linear, the increase of corpus leads to the serious problem of over-fitting. David Blei put forward the LDA model in 2003, which is the most classical topic model (Blei, Ng & Jordan, 2003). The most of later topic models are based on the improved LDA model. The LDA model extends the PLSA model by Bayesian theory and introduces a priori parameter, which greatly reduces the over-fiting problem.

After the LDA model was put forward, the field of topic models ushered in blowout development. In the next year, David Blei et al. also published Hierarchical Latent Dirichlet Allocation (HLDA) model, which establishes the hierarchical relationship between topics, and the model be able to automatically generate the number of topics (Griffiths, Jordan, Tenenbaum & Blei, 2004). Ramage et al. proposed Labeled LDA model in 2009, which combines LDA model with supervised learning (Ramage, Hall,

Nallapati & Manning, 2009). Perotte et al. proposed Hierachical Labeled Latent Dirichlet Allocation (HLLDA) model in 2011, which combines HLDA and Labeled LDA, and it is the first supervised and hierarchical topic model (Perotte, Wood, Elhadad & Bartlett, 2011). Dieng et al. put forward Topic-RNN model in 2017, which combines global information extraction of topic model and local feature extraction of Recurrent Neural Network (Dieng, Wang, Gao & Paisley, 2016).

With deep learning becoming the cutting-edge technology of machine learning, more and more algorithm are based on it. While, the topic model is based on probability, and its combination with deep learning has always been difficult to achieve very ideal effect (Crossno et al., 2011). Google proposed Word2vec model in 2013, which is a vector extraction model based on neural network (Mikolov et al., 2013). Since then, the topic model has gradually been replaced by Word2vec. However, LDA still has many advantages that word2vec cannot relace. For example, LDA model be able to train the relationship between words and topics, topics and documents, but Word2vec can only train the similarity between words. In additional, the training of deep learning requires huge data set to achieve the best training effect, and our data set are difficult to support the training of deep learning. Thus, vector extraction using LDA model is more suitable for our project.

However, despite more than ten years of development, LDA model has still been the most classical topic model. Therefore, LDA model and clustering algorithms will be adopted to calculate the similarity between Web services to solve the cold-start problem of collaborative filtering algorithms.

1.3 Research Questions

Recommender systems has been widely used in the field of data mining in recent years; topic vector extraction, clustering, and recommendation ranks are the basic processes

of recommendation models. Therefore, the research question of this thesis are:

1. How to utilize the relationship between users and integrate user profile, in order to classify users into groups with high accuracy.

   In this thesis, the data set are crawled from the online Mashups and Web APIs repositories of ProgrammableWeb. We adopt LDA model and TF-IDF model to extract topic vectors of Mashups, and use JS Divergence to calculate the similarity between Mashups, then utilize K-means and Agnes algorithm to unsupervised clustering of Mashups. Ultimately, it will be able to achieve automatic user data classification.

2. How to rank and recommend relevant and diverse Web APIs in common Mashup service clusters.

   A good recommendation result should recommend more relevant results, and also include both popular and non-popular results. We have adopted a variety of collaborative filtering algorithms, and compare the rank of recommendation result by Discounted Cumulative Gain (DCG) and Hamming Distance (HMD) indicators.

Since the core idea of this thesis is to adopt hybrid recommendation algorithm to improve the accuracy of recommendation result, so we need to compare various model and select more efficient recommendation methods.

1.4 Contribution

The contribution of this thesis is a Web APIS recomender system based on hybird model. We will extract Mashup content by topic model and clustering algorithms, and implement Web APIs recommendation by multiple collaborative filtering algorithms.

The experiments is divided into four parts: 1) crawling data set from the online Mashups and Web APIs repositories of $ProgrammableWeb$; 2) topic vector extraction by topic model; 3) clustering the Mashups by topic vector of topic distribution; 4) recommendation by various collabrative filtering algorithm.

Moreover, this thesis also discuss the related algorithms for topic model, clustering model, and recommendation model. The recommended results of different parameters and algorithms are compared and analyzed in the same data set. The research of topic model and clustering algorithm provides a reference solution for the automatic classification replace manual in related websites. The analysis and comparison of recommendation algorithms be albe to provide more guarantee for the cooperation relationship between website and users.

1.5 Thesis Structure

This thesis consists of six chapters:

In the second chapter, we mainly introduce literature review. First, we will introduce the evolution of recommender systems, and the categories of recommender systems. Then we list some clssical algorithm and model in recommder systems, and systematically introduce the working principle of collaborative filtering recommendation model and topic model. Finally, we introduce the classical algorithm of clustering algorithm, and the interrelation between topic model and clustering algorithm. Therefore, the second chapter is more about learning and understanding the results of previous studies, summarizing experience, in order to better carry out our research.

In the third chapter, we will discuss the methodology of this thesis. These include data collection method, topic model methods, clustering algorithm methods, and a variety of collaborative filtering algorithm, as well as recommender system model design methods. The design and implementation of experimental process will also be

listed.

In the fourth chapter, the experiment result are mainly introduced, including the training and testing results in different recommendation algorithm, as well as the analysis of the experimental results under different parameters. We will also show the figure, table, and chart with an intuitive explanation.

In the fifth chapter, we will analyze and discuss our experimental results and compare them with different algorithm, in order to demonstrate the advantage and disavantage of different recommendation algorithm in practical application. The conclusion and future work will be introduced in the sixth chapter.

Chapter 2

Literature Review

2.1 Introduction

The research direction of this thesis is to implement recommender system based on the framework of clustering algorithm with topic model method. This chapter starts with the evolution of recommender system to introduce the content-based recommender system, collaborative filtering based recommender system, and hybrid based recommender. At the same time, the advantages and disadvantages of different types of recommender system are explained. Then, we will elaborate on different stages of the topic model from the perspective of the evolution of the topic model algorithm, such as Latent Sematic Analysis, Probability Latent Semantic Analysis, and Latent Dirichlet Allocation. Among them, this section will specifically describe the working principle of Latent Dirihlet Allocation, based on the reason that it is the mainstream algorithm of topic modelling. Finally, we describes different types of clustering algorithms, and specifically elaborates the partition-based clustering algorithm and statistics-based clustering algorithm.

2.2 Recommender System

Recommender system is a personalized information service system, which be able to serve as a bridge between users and information resources. It is a tool to help users quickly find useful information. The rise of recommender system is closely related to the development of information technology and world wide web. With the development of information technology and world wide web, human society has entered the era of information overload from the era of information scarcity, and information is more and more difficult to display to user who may be interested in it with the amount of information increasing. The task of recommender system is to connect information with users, so as to improve the efficiency and realize its value. Recommender system can be used in any size of e-commerce websites, and provide recommendation result that users may interest.

2.2.1 Evolution of Recommender System

The early recommender system evolved from textual mining based on term frequency - inverse document frequency (TF-IDF) (Spärck Jones, 2004). The concept of recommender system originated from the use of Tapestry system by Xerox in 2012 to solve the problem of mail overload in research centers. The first personalized recommender system is GroupLens (Konstan et al., 1997), which was launched by the University of Minnesota GroupLens Research Group in 1994. The GroupLens system first proposed the idea collaborative filtering to implement recommendation and established a formal model for recommendation system which has led the development direction of recommendation system. Recommender system has been widely used and has become an important research field, since Jannach et al. first proposed the term recommender system in 1997 (Jannach, Resnick, Tuzhilin & Zanker, 2016). Adomavicius et al. divided recommender system into three main categories (Adomavicius & Tuzhilin, 2005),

which content-based recommender system, collaborative filtering recommender system, and hybrid recommender system. David et al. released latent Dirichlet allocation (LDA) in 2003 (Blei et al., 2003), this algorithm is also the core mainstream idea of contemporary topic model algorithm. In recent years, some researchers have tried to apply singular value decomposition (SVD) (Shah, Pareek, Patel & Panchal, 2013), deep learning (Ouhbi, Frikh, Zemmouri & Abbad, 2018), and reinforcement learning (Munemasa, Tomomatsu, Hayashi & Takagi, 2018) etc. techniques into recommender system, which are achieving good recommendation effect.

The generation of search engine and recommender system provides an important technical to solve the problem of information overload. In a fundamental point of view, search engine is also a kind of recommender system. Search engine matches relevant information according to the keywords input by users, and recommender system not only passive matching related information, but also actively recommend information matched with users' historical information. The basic framework of recommender system development be able to be understood as six parts, which are the relationship between items, the relationship between keywords and items, the relationship between categories and items, the relationship between clustering and items, the relationship between users and items, and the relationship between users. The flow chart of recommender system basic framework is shown in Figure 2.1.

2.2.2 The Categories of Recommender System

The recommender system mentioned above can be divided into three categories: content-based recommendation model, collaborative filtering recommendation model and hybrid model. The content-based recommendation model (Gharia et al., 2018) mainly establishes the archives of users and items, and then calculates the similarity between users

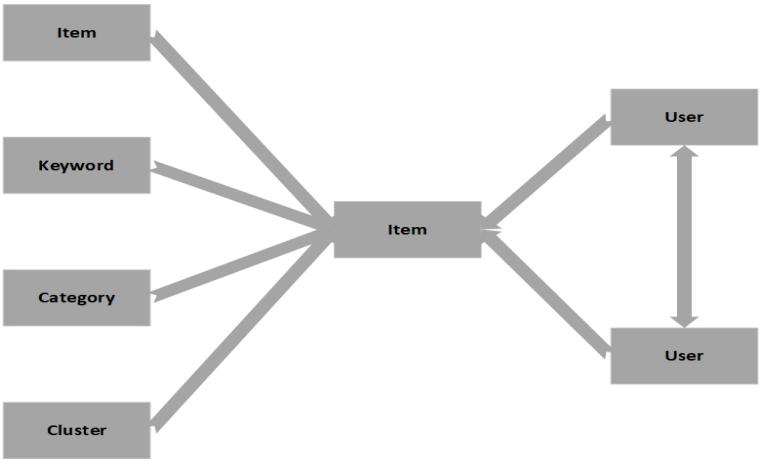

Figure 2.1: The Flow Chart of Recommender System Basic Framework

and items. The key problem of content-based recommendation is to model user interest and item profiles (Wu et al., 2018). The main methods are topic modelling, vector space model, linear classification, linear regression etc. Collaborative filtering recommendation model does not need to collect user interest and item profiles, but only gathers user history behavior records, and then discovers the potential similarity between users and users, items and items, and achieve recommendation based on the similarity of these group (L. Zhou, Tang & Dong, 2017). However, the commonly used recommender system model in industry are hybrid model (Khan et al., 2018), which be able to be combined by various ways of ensemble learning, or it can be an end-to-end model. For example, we be able to establish the recommendation model that contains multiple recommendation model and apply the voting method to determine the final recommendation result. Hybrid recommendation is not inferior to any single recommender system category in theory, but its complexity and optimization problems may affect its recommendation accuracy.

1. Content-based recommender system

   Content-based recommender systems recommend content to users, according to users' historical search. It is mainly used in information retrieval system, and the

most of technology in information retrieval and information filtering be able to be applied in content-based recommender system (Adomavicius & Tuzhilin, 2005). Content-based recommender system mainly consists of three part, which are item representation, profile learning, and recommendation generation (Pazzani & Billsus, 2007). Item representation extract the content of each item to represent the item; profile learning applies the method of machine learning to calculate user behaviors; recommendation generation recommend relevant items to users by calculating the similarity between user behaviors and dataset.

The item representation section is the raw data preprocessing section, which convert the raw data into vectors needed by the profile learning section (Pazzani & Billsus, 2007). In practical use, each item often has some attributes that can describe it. These attributes can usually be divided into two types: structured attribute and unstructured attribute. The structured attribute data are well-defined, which the vector value selection is more stable; while unstructured attribute data are not well-defined, which difficult to get vector value and invoke directly. The structured attribute data often be able to invoke in profile learning section, such as the 'tag' and 'followers' columns in our original dataset; the unstructured attribute data often need to convert it into structured attributes data by information indexing model, such as TF-IDF, and LDA etc. Examples of structured attribute data be able to refer the 'tag' and 'followers' columns in our original dataset, while examples of unstructured attribute data be able to refer the 'description' column.

The profile learning section is the process of data modeling, which training and learning the vector of historical content (Lops, De Gemmis & Semeraro, 2011). Both supervised classification or unsupervised clustering model be able to adopt in profile learning section, such as k-Nearest Neighbor, Rocchio, Linear Classifer, Decision Tree, Naïve Bayes, LDA clustering, k-meaning clustering etc. The

recommendation generation section matches user behaviors and dataset with high similarity and output recommendation contents to users, which the similarity calculation be able to adopt Pearson Correlation Coefficient, Jensen-Shannon Divergence, Kullback-Leibler Divergence etc.

In addition, attenuation mechanism is also an optional part in content-based recommender system. We can bring in an attenuation mechanism, in order to solve the problem of users' interests may change over time. Each user's keyword preference is attenuated periodically, which bring in a coefficient, and the preference of every users' keywords are multiplied by the coefficient at intervals to simulate the process of user interest migration (Blanco-Fernández, López-Nores, Gil-Solla, Ramos-Cabrer & Pazos-Arias, 2011).

The advantages of content-based recommender system can be divided into three point: user independence, transparency, and new item problem. For user independence, each user's profile is based on his own information, so the recommendation results are not disturbed by external factors; for transparency, recommendation result has high similarity with user behaviors; for new item problem, all contents are recommended in the same probability, so new items can be retrieved as well. And the disadvantages of content-based recommender system also can be divided into three point: limited content analysis, over-specialization, and new user problem. The limited content analysis means feature extraction of items is usually hard to realize; the over-specialization represents it unable to mining potential interests of users; the new user problem is cannot generate recommendations for new users (Pazzani & Billsus, 2007).

2. CF-based recommender system

   Collaborative filtering, as a classical recommendation algorithm, is widely used in industry of Internet. It has many advantages, such as strong universality of the model, no need for professional knowledge in data mining, simple engineering

implementation, and well precision of recommendation (Qu, Yao, Wang & Yin, 2018). Therefore, collaborative filtering-based recommender system is the most mainstream recommendation method at present (Kwon & Jung, 2013). Collaborative filtering algorithm can be divided into three types: user-based collaborative filtering, item-based collaborative filtering, and model-based collaborative filtering (Sharma, Gopalani & Meena, 2017) (Badaro, Hajj, El-Hajj & Nachman, 2013). User-based collaborative filtering needs to find the similarity relationship between users by online; item-based collaborative filtering be able to adopt offline data to calculate the similarity relationship between items; model-based collaborative filtering adopts data modelling by the method of machine learning. User-based collaborative filtering evaluates the similarity of users by rating items from different users, and then implements recommendation based on user similarity. Generally speaking, recommend items that users are interested in who has similar tastes and interests. The implementation principle of user-based collaborative filtering recommendation algorithm is able to be divided into two steps: 1) find a user set similar to the interest of the target user; 2) find items that the user likes in the set and the target user has not heard of and recommend them to the target user (Tan & He, 2017). User-based collaborative filtering requires synchronous data collocation by online, considering user interests migration factors. In addition, Jaccard similarity and Cosine similarity are the most common method to calculate the similarity between users.

Item-based collaborative filtering evaluates the similarity between items by the users' rating of different items and makes recommendations based on the similarity between items (Tao, Cheung, She & Lam, 2014). Generally speaking, recommend items similar to what user interested. Item-based collaborative filtering does not invoke the content attributes of items to calculates the similarity between items, it mainly calculates the similarity between items by analyzing the

user's behavior records. The implementation principle of item-based collaborative filtering recommendation algorithm is also bel to be divided into two steps: 1) calculate the similarity between items; 2) generate recommendation lists for users according to similarity of items and historical behavior of users (Abdelkhalek, Boukhris & Elouedi, 2016). Item-based collaborative filtering evaluates differentiated from user-based collaborative is embodied in the co-occurrence matrix, which tall all users' records of items and form a co-occurrence matrix that reflects the correlation degree of items (Kawasaki & Hasuike, 2017).

Model-based collaborative filtering, as the most popular type of collaborative filtering, is often adopts machine learning algorithm to predict user's rating on an item (Y. Shi, Larson & Hanjalic, 2014). It will be composed of m users and n items, among which some items are rated by users. The number of users and items are huge, so the user-item matrix generally generates a sparse matrix. However, it has only a part of users and a part of items have rating recorded, while others are black. At this time, we need to use the part of sparse data to predict the rating relationship between those blank items and users, and then seek out the highest rating items recommended to users. Mainstream method of model-based collaborative filtering algorithm be able to be divided into: association rule algorithm, such as Apriori (AlZu'bi, Hawashin, EIBes & Al-Ayyoub, 2018), FP Tree (Shuai, Song, Wang & Zhan, 2018), and PrefixSpan (Ma & Ye, 2018); clustering algorithm, such as k-mean clustering (Bagde & Tripathi, 2018), balancd iterative reducing and clustering using Hierachies (Madan & Dana, 2016), density-based spatial clustering of application with noise, and spectral clustering (Y. Zhang, Liu & Deng, 2013); classification algorithm, such as logistic regression (Wooff, 2004), and Naive Bayes (Yu, Jiang, Zhang & Wang, 2018); matrix decomposition algorithm, such as singular value decomposition (SVD) (Paterek, 2007), Funk-SVD (Jannach, Gedikli, Karakaya, Juwig et al., 2012), Bias-SVD (Fang & Guo, 2013),

and SVD++ (Kumar, Verma & Rastogi, 2014); neural network algorithm, such as Restricted Boltzmann Machine (Salakhutdinov, Mnih & Hinton, 2007); graph-based algorithm, such as SimRank (Jeh & Widom, 2002); Topic Model algorithm, such as LDA (Pennacchiotti & Gurumurthy, 2011), latent sematic analysis (LSA) (Nanopoulos, Rafailidis, Symeonidis & Manolopoulos, 2010), and probabilistic latent sematic analysis (pLSA) (Hofmann, 2003).

2.3 Topic Modelling

Topic modelling is a popular technology in the field of natural language processing, which be able to similarity comparison, keyword extraction, classification, dimensionality reduction, and elimination of noise in document (Bergamaschi & Po, 2014). The concept of topic modelling is a modelling method for implicit topics in document (Blei & Lafferty, 2009). More broadly, most of implicit semantic analysis techniques belong to the category of topic models, such as LDA model (Blei et al., 2003) and SVD model (De Lathauwer, De Moor & Vandewalle, 2000); while from the narrow point of view, topic model refers to the implicit analysis techniques based on conditional probability distribution of words, such as LDA model (Blei et al., 2003), pLSA model (Hofmann, 1999), LSA model (Landauer et al., 1998), and Hierarchical Dirichlet Process (HDP) (Teh, Jordan, Beal & Blei, 2005). Nowadays, topic models are often expressed as their specific meanings in the field of recommender system.

2.3.1 Latent Sematic Analysis (LSA)

Initially, the concept of topic modelling was derived from the improvement of TF-IDF (W. Zhang, Yoshida & Tang, 2011) and vector space model (VSM) (H. Xu et al., 2015) techniques. Two of the most important obstacles to TF-IDF and VSM are polysemy and synonymy of word in corpora (Newman & Block, 2006). The synonymy will reduce

the recall of TF-IDF's indexing result, and the polysemy will reduce the precision of TF-IDF's indexing result. The LSA model was proposed by Papadimitriou et al. in 1998, in order to overcome the shortcoming of TF-IDF and VSM method (Landauer et al., 1998). The LSA model came into being and attracted more attention, which laid a solid ideological foundation for the follow-up development of topic model (Steyvers & Griffiths, 2007). The LSA has made a direct and far-reaching theoretical basis for latent factor model (LFM) and probabilistic latent semantic analysis (pLSA) (Crossno et al., 2011).

The LSA model be able to deal with the synonymy problem that VSM and TF-IDF not be able to solve, but it cannot solve the polysemy problem (Sahlgren, 2002). Since LSA maps word into a point in the latent semantic space, which polysemy of word are the same point in the latent semantic space that not distinguished (Landauer, 1998). In addition, LSA has three other important drawbacks: the process of singular value decomposition (SVD) lacks clear physical significance; high computational complexity, which is difficult to meet the challenges brought by the big data; the basis of mathematical statistics is not solid enough, and negative numbers sometimes appear in the decomposition results (Miaskiewicz, Sumner & Kozar, 2008).

2.3.2 Probabilistic Latent Semantic Analysis (PLSA)

Hoffman proposed pLSA based on the theoretical basis of LSA in 1999, in order to solve the polysemy problem (Hofmann, 1999). The pLSA adopts probabilistic models to express the problem of LSA (Jin, Zhou & Mobasher, 2004). That is establishing likelihood function for observed variables and exposing latent variables, then using Expectation Maximization algorithm (EM algorithm) to calculate the experimental result (Hofmann, 2001). The EM algorithm is the core algorithm of pLSA, and the algorithm can be divided into two parts: E-step and M-step, which two step iteration

until convergence (Neal & Hinton, 1998). The E-step calculates the posterior probability of the implicit variable given under the parameter conditions currently estimated; the M-step maximizes the expectation of complete data logarithmic likelihood function, and then we use the posterior probability of implicit variable calculated in E-step to obtain new parameter values. The pLSA provides a solid theoretical and practical basis for the classical topic model algorithm – LDA (Chang & Hsiao, 2013).

The superiority of pLSA are mainly derived from the improvement of LSA model, while it still has a lot to improve (Jin et al., 2004). The advantages of pLSA be able to be summarized: 1) it defines the probability model, and each variable and its corresponding probability distribution and conditional probability distribution have a clear physical significance; 2) the Multi-nomial distribution hypothesis implied in pLSA is more consistent with the text characteristics, compared with the Gauss distribution hypothesis implied in LSA; 3) the computational complexity is relatively reduced, because the optimization objective of pLSA is to minimize KL-divergence rather than relying on the criterion of mean-square error (MSE); 4) various criteria of model selection and complexity control can be adopted to determine the dimension of topic. The shortcoming of pLSA can be summarized: 1) the probabilistic model is not complete enough, which has not suitable probabilistic model at the document level and it is not a complete generative model, so that the amount of documents have to be determined; 2) As the number of documents and terms increases, the pLSA model increases linearly and becomes larger and larger; 3) there is no good method to get the probability of topic when a new document has been added; 4) EM algorithm has to be repeated iterations, so it still need a high computation (G. Xu, Zhang & Zhou, 2005).

2.3.3 Latent Dirichlet Allocation (LDA)

LDA is the most classical model of topic modelling, and the most of the other modern topic models make the most of the theoretic and framework of LDA model to design and realize (Lin, Tian, Mei & Cheng, 2014), so understanding LDA model is the only way which must be passed to comprehend topic modelling. Blei at al. proposed the LDA model by extending the pLSA model in 2003 (Blei et al., 2003), which is a more complete generation probability model of discrete data set. From the point of view of the relationship between LDA and pLSA, LDA model only adds a Bayesian framework on the basis of pLSA model (Lu, Mei & Zhai, 2011). The pLSA belongs to the Frequentists, which the sample is random, and its parameters are unknown but fixed; while the LDA belongs to the Bayesians, which the sample is fixed, and its parameters are unknow and not fixed that subordinate to a certain distribution (Gallé, 2014). The frequentists take the approximate probability of frequency as its core idea, it has relatively low computational complexity, so it be able to efficiently complete the probability analysis of simple events; while the Bayesians be able to solve the inference problem that the frequentists incapable of action, and the Bayesians regards probability as a variable and independent the super-parametric of the variable (Bayarri & Berger, 2004).

The basic principle of LDA can be divided into two parts: prior distribution and text modeling (Wallach, Mimno & McCallum, 2009). The prior distribution involves massive mathematical knowledge, such as Beta-Binomial Conjugate, Gamma Function, Dirichlet Multinomial Conjugate, Dirichlet Distribution, Markov Chain Monte Carlo (MCMC), and Gibbs Sampling; the text modeling involves Unigram model, pLSA model, and LDA model (Blei et al., 2003). According to its working principle, LDA can be divided into five steps: 1) one function – Gamma function (Ayadi, Maraoui & Zrigui, 2015); 2) four distribution – Binomial distribution, Multinomial distribution,

Beta distribution, and Dirichlet distribution (Sadamitsu et al., 2007); 3) one concept and one framework: conjugate prior and Bayesian framework (Liu, Sharan, Adelson & Rosenholtz, 2010); 4) three model – Unigram model, pLSA model, and LDA model (Lu et al., 2011); 5) one sampling: Gibbs sampling (Darling, 2011). In additional, the topic distribution of words in each document and the distribution of words under each topic obeys the Multinomial distribution, and its priori selection of conjugate priors by the Dirichlet distribution in LDA (Bíró, 2009).

1. Gamma function

   Gamma function is closely related to Poisson function, and it has high consistency in mathematical form. The difference of them is that the Poisson distribution is discrete, and the Gamma distribution is continuous. The function of Gamma distribution is shown in below (Ayadi et al., 2015):

$$\Gamma(x) = \int_0^{+\infty} t^{x-1} e^{-t} dt \tag{2.1}$$

2. Binomial Distribution and Multinomial Distribution

   In probability and statistics, binomial distribution is n independent discrete probability distributions, in which the probability of each experiment is p. The single experiment of Binomial distribution also known as the Bernoulli distribution. In fact, the binomial distribution is the Bernoulli distribution when n equals 1. In additional, the ultimate limit of Binomial Distribution is Poisson. The function of Binomial Distribution shown in below (Sadamitsu et al., 2007):

$$P(K = k|n, p) = \binom{n}{k} p^k (1-p)^{n-k} \tag{2.2}$$

   Assuming that a coin is tossed n times, the probability distribution of k times

facing upward is obtained, the $\binom{n}{k}$ is the Binomial coefficient, and the p is the probability of coins facing upward. It is noteworthy that the random variables of binomial distribution are 0 or 1, so the p is fixed value 0.5. The equation of $\binom{n}{k}$ is (Sadamitsu et al., 2007):

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{2.3}$$

The multinomial distribution is the case that binomial distribution extends to multi-dimension, and the value of p is not fixed. The function of multinomial distribution shown in below (Sadamitsu et al., 2007):

$$P(x1, x2, ..., x_k; n, p1, p2, ..., p_k) = \frac{n!}{x_1!...x_k!} p_1^{x1} ... p_k^{x_k} \tag{2.4}$$

3. Beta Distribution

Beta distribution be able to be regarded as distribution over binominals and a conjugate prior of binominal distribution. Beta distribution can be seen as the distribution obtained from repeated experiments of binomial distribution exactly as the binomial distribution be able to be seen as the distribution obtained from repeated experiments of the Bernoulli distribution. In probability, Beata distribution refers to a set of continuous probability distributions defined in $(0, 1)$ interval, it has two parameters: alpha and beta, and both alpha and beta are greater than 0. The probability density function of Beta distribution is (Sadamitsu et al., 2007):

$$f(x; \alpha, \beta) = \frac{1}{B(\alpha, \beta)} x^{a-1} (1-x)^{\beta-1} \tag{2.5}$$

Among them,

$$\frac{1}{B(\alpha,\beta)} = \frac{\Gamma+\beta}{\Gamma(\alpha)\Gamma(\beta)} \tag{2.6}$$

4. Dirichlet Distribution

   Dirichlet distribution is a distribution over a multinomial distribution. In simple terms, we have obtained a set of probability distributions from a multinomial distribution, and the probability distributions that compute the same probability distribution are called Dirichlet distribution. The probability density function of Dirichlet distribution is (Sadamitsu et al., 2007):

$$f(x1,x2,...,x_k;a1,a2,...,a_k) = \frac{1}{B(\alpha)}\prod_{i=1}^{k} x_i^{a_i-1} \tag{2.7}$$

   Among them, $B(\alpha)$ is a multinomial beta distribution:

$$B(\alpha) = \frac{(\prod)_{i=1}^{k}\Gamma(a_i)}{\Gamma((\sum)_{i=1}^{k}a_i} \tag{2.8}$$

5. Conjugate prior Distribution and Bayesian Framework

   In Bayesian probability theory, if the posterior probability $p(\theta|x)$ and the prior probability $p(\theta)$ satisfy the same distribution law, then the prior distribution and the posterior distribution are called conjugate distribution. In additional, the prior distribution is called the conjugate prior distribution of likelihood function. The Beta distribution is a conjugate prior distribution of binomial distribution, while Dirichlet distribution is a conjugate prior distribution of multinomial distribution. The probability distribution function of conjugate distribution is (Liu et al., 2010):

$$p(\theta|x) = \frac{p(\theta,x)}{p(x)} \tag{2.9}$$

   In the Conjugate prior distribution, the $p(\theta)$ represents prior probability that

the probability of θ occurrence; the $p(x)$ represents the probability of dataset x occurrence; the $p(\theta|x)$ represents posterior probability that the probability of θ occurrence supported by data-set x. The Bayesian Framework adopts posterior distribution as core theoretical basis.

6. Mixture of Uni-gram model

   Uni-gram model is the basic model for both pLSA and LDA, it adopts a bag of word model, which assumes that documents are independent of each other and that vocabulary in documents are also independent of each other. The model obtains the probability distribution function of a word through training corpus, and then generates one word each time according to the probability distribution function. The generation function is (Lu et al., 2011):

$$p(w) = \prod_{n=1}^{N} p(w_n) \tag{2.10}$$

   Among then, the w represent a document, which $w = (w_1, w_2, ..., w_n)$; the N represent the number of vocabularies in a document, and N is random variable. The diagram of uni-gram model shown in Figure 2.2.

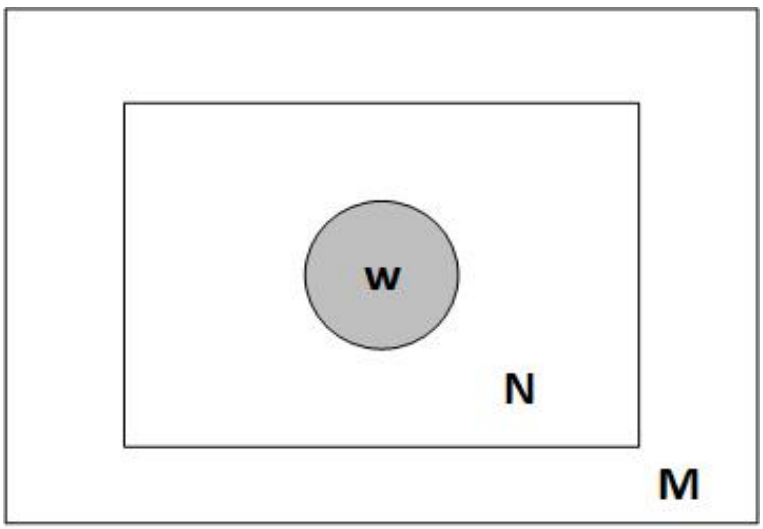

Figure 2.2: The Diagram of Uni-gram Model

The shortcoming of the uni-gram model is that the generated text has no topic and is too simple. To solve this problem, Mixture of uni-gram model bring in a single topic. In the mixture of uni-gram model, z represents a topic, $p(z)$ represents probability distribution of topic, z generated by the probability distribution of $p(z)$, $p(w|z)$ represents the distribution of w when given z. The diagram of mixture of uni-gram model is shown in Figure 2.3.

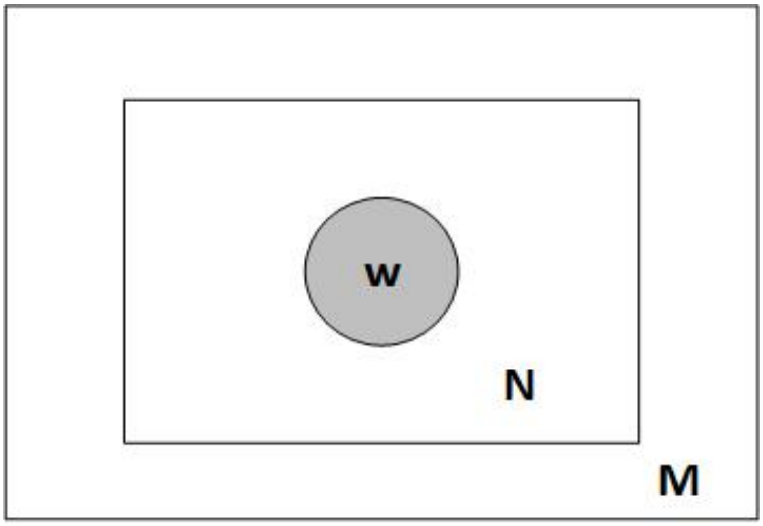

Figure 2.3: The diagram of Mixture Uni-gram Model

7. PLSA model

   The disadvantage of mixture of unigram model is that only one topic is allowed a document. This assumption is basically not valid in actual application, and most documents have multiple topics. PLSA model bring in multiple topic, and the topic of the document is also obtained according to a certain probability (Lu et al., 2011). The PLSA model bring in a parameter d, it represents document in corpus. The diagram of PLSA model is shown in Figure 2.4.

8. LDA model

   The shortcoming of PLSA model is that only analyses documents from the perspective of probability but ignores the implicit semantics of document. The LDA

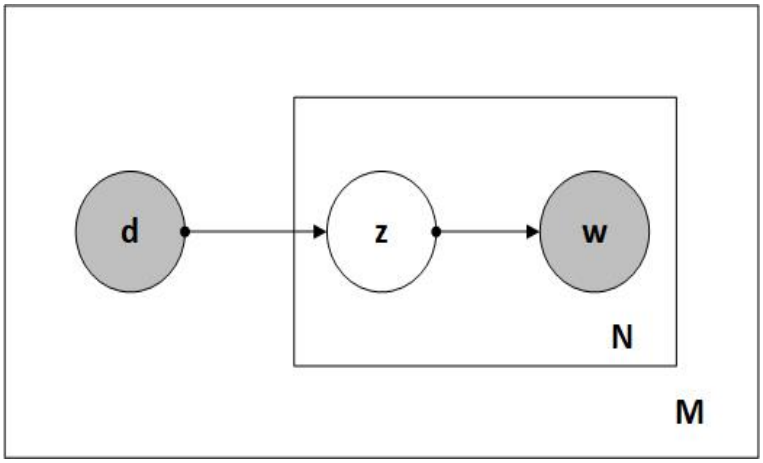

Figure 2.4: The Diagram of PLSA Model

model layered Bayesian framework based on PLSA model. In LDA model, the probability distribution of each topic in document and the probability distribution of each word in a topic are random variables, which correspond to two Dirichlet priori parameters respectively. In the diagram of LDA model, the α represents proportions parameter, the θ_d represents pre-document topic proportions, $Z_(d,n)$ represents pre-word topic assignment, $W_(d,n)$ represents observed word, β_k represents topics, η represents topic parameter. The diagram of LDA model is shown in Figure 2.5.

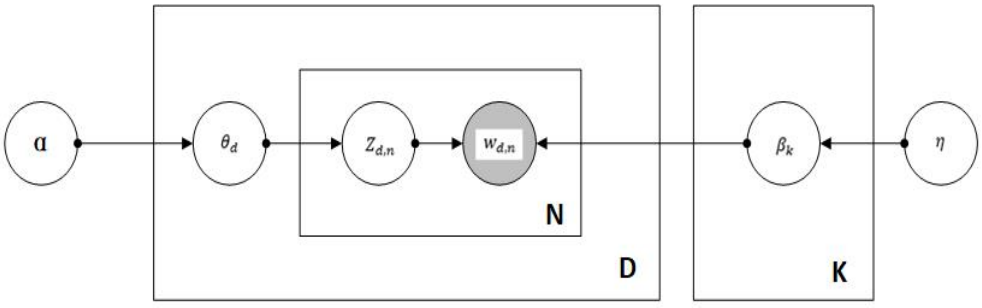

Figure 2.5: The Diagram of LDA Model

Among them, the function of pre-document topic proportions is (Lu et al., 2011):

$$\theta_d = Dirichlet(\vec{\alpha}) \tag{2.11}$$

The function of word distribution of arbitrarily a topic k is:

$$\beta_k = Dirichlet(\overrightarrow{\alpha}) \tag{2.12}$$

The function of pre-word topic assignment is:

$$Z_{d,n} = multi(\theta_d) \tag{2.13}$$

The function of observed word is:

$$w_{d,n} = multi(\beta_{z_{d,n}}) \tag{2.14}$$

In additional, α , θ_d, and z_d be able to constitute Dirichlet-multinomial conjugate. Utilizing Dirichlet-multinomial conjugate, the proportions of $theta_d$ is $Dirichlet(\theta_d|\overrightarrow{\alpha} + \overrightarrow{n_d})$, and the proportions of β_k is $Dirichlet(\beta_k|\overrightarrow{\eta} + \overrightarrow{n_k})$.

9. Gibbs Sampling

   Gibbs sampling is an algorithm adopts MCMC theory to obtain a series of observation samples that approximately equal to the specified multidimensional probability distribution. Similar to PLSA, the original thesis of LDA used EM algorithm to estimate unknown parameters, and the Gibbs Sampling algorithm has effective on LDA result was found in later experiment (Darling, 2011). Therefore, the difference between LDA and PLSA are Bayesian framework and Gibbs Sampling.

   The working principle of MCMC theory is that given the probability distribution $P_0(x)$ of the initial random variable x_0, after using MCMC method to make state transition i times, and then the matrix of the ith power of the state transition matrix will be the same for each column, so the probability distribution $P_i(x)$ of the random variable x_i will not change. Therefore, the core of MCMC method

is to construct a Markov chain whose transition matrix p, so that the smooth distribution of the Markov chain happens to be the probability distribution $p(x)$ of the sample. Then we start from any initial sample x_0, and the Markov chain converges after n steps. At this time, the $n+1$ step is the sample of the probability distribution $p(x)$ we need. And the core idea of Gibbs Sampling is MCMC method.

In LDA, we focus on three parameters: z, θ, and ϕ. Among then, z is the implicit topic corresponding to each word in the corpus, θ is the topic distribution of each document in the corpus, and the ϕ is the term distribution of each topic. In fact, the θ and the ϕ can be obtained by likelihood estimation calculation, if the z is obtained. So, it is necessary to integrate the probability formula $p(w, z, \theta, \phi|\alpha, \beta)$ to remove the θ and ϕ by integral method and obtain the probability formula $p(w, z|\alpha, \beta)$. Then slove the formula of $p(z|w, \theta, \beta)$ (Bíró, 2009):

$$p(z|w, \theta, \beta) = \frac{p(w, z|\alpha, \beta)}{p(w|\alpha, \beta)} \tag{2.15}$$

The task of Gibbs Sampling in LDA is to complete the sampling of $p(z|w, \theta, \beta)$.

2.3.4 Latent Factor Model

The concept of latent factor model published by Simon Funk in 2006 (Bell & Koren, 2007). The core algorithm of latent factor model is Funk-SVD algorithm (Jannach, Lerche, Gedikli & Bonnin, 2013), which is one of the well-known algorithms in the field of recommender system. And the core method of latent factor model is decomposing a three-dimensional matrix that consists of latent factor – user matrix, latent factor – item matrix, and user – item matrix (Koren, Bell & Volinsky, 2009). The practical application of latent factor model has two aspects: user rating prediction, and Top-N popular ranking of items in recommender system (Agarwal & Chen, 2009). It mainly

adopts Stochastic Gradient Descent to process loss function, thus completing the users' lack rating in matrix.

Topic modelling is a proper noun in the field of text mining, while latent factor model is a proper noun in the field of recommender system, but their algorithm ideas are consistent. In the field of text mining, topic model is used to find hidden topics in documents, and the algorithm of latent factor model was first proposed in the field of testing mining to find implicit semantics of documents (Wilson & Zhang, 2016). Therefore, latent factor model is essentially a type of topic model. However, latent factor model has a good performance in calculating users' rating on various items, while the performance of content mining is not ideal.

2.4 Clustering Algorithm

Clustering algorithm is to classify the samples with similar characteristics one group according to some similarity measure, which makes the similarity of differences within the class smaller, but the difference between the classes larger (R. Xu & Wunsch, 2005). It is an important method in the field of data mining. Up to now, there is no academically accepted definition of clustering algorithm (Estivill-Castro & Yang, 2000). The purpose of clustering algorithm is to find the potential natural grouping structure and the relationship of interest in the corpora (R. Xu & Wunsch, 2005). Clustering analysis is a mathematical method to study and deal with the classification of the given objects and the degree of affinity between them, which is a tool to analyze corpora without any assumptions (Hansen & Jaumard, 1997).

The types of clustering algorithm be able to be divided into six categories (Kou, Peng & Wang, 2014): partition-based clustering algorithm, such as k-means (Sarwar, Karypis, Konstan & Riedl, 2002), and Clustering Large Applications based upon RANdomized Search (CLARANS) (Ng & Han, 2002); statistics-based clustering algorithm, such as

LDA clustering (Krestel, Fankhauser & Nejdl, 2009), and PLSA clustering (Hofmann, 2003); hierarchical clustering algorithm, such as Agnes (Cao et al., 2017), Balanced Iterative Reducing and Clustering Using Hierarchies (BIRCH), and Clustering Using Representatives (CURE) (Su, Yeh, Philip & Tseng, 2010); density-based clustering, such as Density-based spatial clustering of applications with noise (DBSCAN) (Ester, Kriegel, Sander & Xu, 1996), and Clustering by fast search and find of density peaks (DP) (Rodriguez & Laio, 2014); grid-based clustering, such as Clustering in Quest (CLIQUE) (J. K. Kim, Cho, Kim, Kim & Suh, 2002), Wave Clustering (Adelfio et al., 2012), and Orthogonal Partitioning Clustering (O-CLUSTER) (Campos & Milenova, 2007), neural network based clustering, such as SpectralNet (Moura, Sadre & Pras, 2014), and Self Organizing Maps (SOM) (Wen-Shung Tai, Wu & Li, 2008). This thesis focus on using partition-based clustering and statistics-based clustering algorithms to achieve the data processing of web APIs and Mashups recommendation.

2.4.1 Partition-based Clustering Algorithm

Traditional partition-based clustering be able to be divided into distance-based clustering and space-based clustering, while most partition-based clustering is distance-based. Among them, distance-based clustering is represented by K-mean algorithm, while space-based clustering is represented by CLARANS algorithm (Kolatch et al., 2001). The working principle of distance-based clustering is to create an initialization partition for a given partition number K. Then, it uses an iterative repositioning technique to partition objects by moving them one group to another (Yan, Fan & Mohamed, 2008). In order to achieve global optimum, clustering based on distance partitioning require to exhaust all possible partitions, which has a huge amount of computation. But the clustering based on spatial partition be able to be extended to subspace clustering instead of searching the whole corpora (Ng & Han, 2002). The general preparation for a good

partition-based clustering is that the objects in the same cluster are as close or related as possible, while the objects in different clusters are as far away or unrelated as possible.

This section will introduce the evolution of partition-based clustering algorithm. The K-means algorithm was proposed by Macqueen in 1967, it is a classical method to solve clustering problems (Park & Jun, 2009). Its main advantage is that the algorithm is simple and fast calculation. While the disadvantage is that different K value may lead to different clustering results; can not find non-convex clusters, or clusters with large distance difference; and sensitive to noise and outlier values, because a small amount of such value be able to have a great impact on the average value. The Partitioning Around Medoid (PAM) and the Clustering Large Applications (CLARA) algorithms were proposed by Kaufman and Rousseeuw in 1990, each of them uses objects close to the center of the class for clustering, thus, they are called k-center algorithm [96]. K-center algorithm can be regards as an improved method of k-mean algorithm, because the center point is not as easily affected by noise and outlier values, so the k-center method is stronger than the k-mean method on large corpora clustering. Many improved partition-based clustering algorithm have been proposed, in order to cluster large-scale data set and deal with complex clustering. Huang proposed the k-modes algorithm in 1999 (Huang & Ng, 1999), which extends the k-means method and replaces the mean value with modes value. The CLARANS was proposed by Ng and Han in 2002, it combines sampling technology with PAM algorithm, which does not consider the whole data set, but randomly chooses a small part of the actual data set as data samples (Ng & Han, 2002). The CLARANS is the first clustering based on spatial partition, which is suitable for clustering analysis of large-scale data set.

This paper is a recommender system based on topic modelling, and partition-based clustering algorithm is used to cluster the clustering result of topic model. In additional, the corpus of the topic model sparse result is not large, so K-mean is more in line with the experimental content of this paper.

2.4.2 Statistics-based Clustering Algorithm

Statistics-based clustering algorithm is essentially model-based clustering algorithm. Its working principle is to establish a model by using statistical methods, and then consider how likely the object is to conform to the model, which are called discordant observation in statistics. Statistics-based clustering algorithm includes: classification tree based statistical model, probability based statistical model.

The representative algorithm of classification tree based statistical model is COBWEB algorithm, which is a popular simple incremental concept clustering algorithm. It creates hierarchical clustering in a form of a classification tree. Each node of the classification tree corresponds to a class, it contains a probability description of the class, which outlines the objects divided under the node of classification tree. All nodes at a certain level of the classification tree form a partition. COBWEB algorithm adopts a method of moving a partial matching function downward along the path of the best matching node in the classification tree to find the best node, in order to classify an object with the classification tree. But if an object does not belong to any of the existing classes in the classification tree, a new class is created for that object. Its advantage is that it does not need user input parameters to determine the number of classes, which be able to automatically correct the number of classes in the partition. However, its disadvantage is that it has a large amount of computation, high storage cost and high hardware requirement.

In the field of machine learning nowadays, probability models can be divided into two type: Frequentists and Bayesians. Frequentists is the repeated sampling, and the more the better, the more infinite; while Bayesians focuses on sampling and distribution. Specifically, the frequentists believe that the model does not need a prior distribution, but prior distribution plays an important role in Bayesians. The representative algorithms of probability based statistical are PLSA clustering and LDA clustering. Which the

PLSA is the representative algorithm of Frequentists, and the LDA is the representative algorithm of Bayesians. As mentioned above, both LDA and PLSA are typical topic model algorithms. Combining clustering algorithm is a kind of unsupervised learning algorithm, and most topic models are also unsupervised learning algorithm, so we can infer that topic model is a type of clustering algorithm.

2.4.3 Development Tend of Clustering Algorithm

This paper combines partition-based clustering algorithm and topic model clustering algorithm to achieve the goal of personalized recommendation of Web API. The sparse recommendation results obtained by LDA algorithm within complex networks, then the k-mean clustering is used to optimize the recommendation results.

2.5 Conclusion

This chapter mainly introduces the development background of the recommender system, the categories of the recommender system, and the working principle of data processing using the topic model and the Mashup clustering method. Nowadays, collaborative filtering based recommendation model is the mainstream predition method in the field of data mining, which the priction results and recommendation accuracy are worthy of recognition. However, this paper mainly introduces the methods of topic model and clustering model as the data processing part of the collaborative filtering based recommendation model, aims on the cold start problem in recommender system. The core method of this paper is based on the theory of probability, since the interpretability of the probability based machine learning model.

Chapter 3

Research Method

3.1 Introduction

In this chapter, we expound the methodology of the our experiment process, which involves the overall of the research designing, the process of data sets collection, the method of topic vector extraction, various collaborative filtering recommendation models, and the assessment methods of recommender systems.

3.2 Research Designing

Since the purpose of this thesis is to implement Web API recommender system based on manifold-learning models, we listed the experimental flow chart of the Web API recommender system in Figure 3.1. The figure display our specific experimental structure by six step, which are Data Collection, Extract Feature Vector, Sparse Vector by Topic Model, Mashup Clustering, recommendation, and Model comparison.

We can sort out the experiments be able to be divided into data collection process, data pre-processing process, data modeling process, unsupervised clustering process, Web API recommendation process, and model comparison process. Among then, the

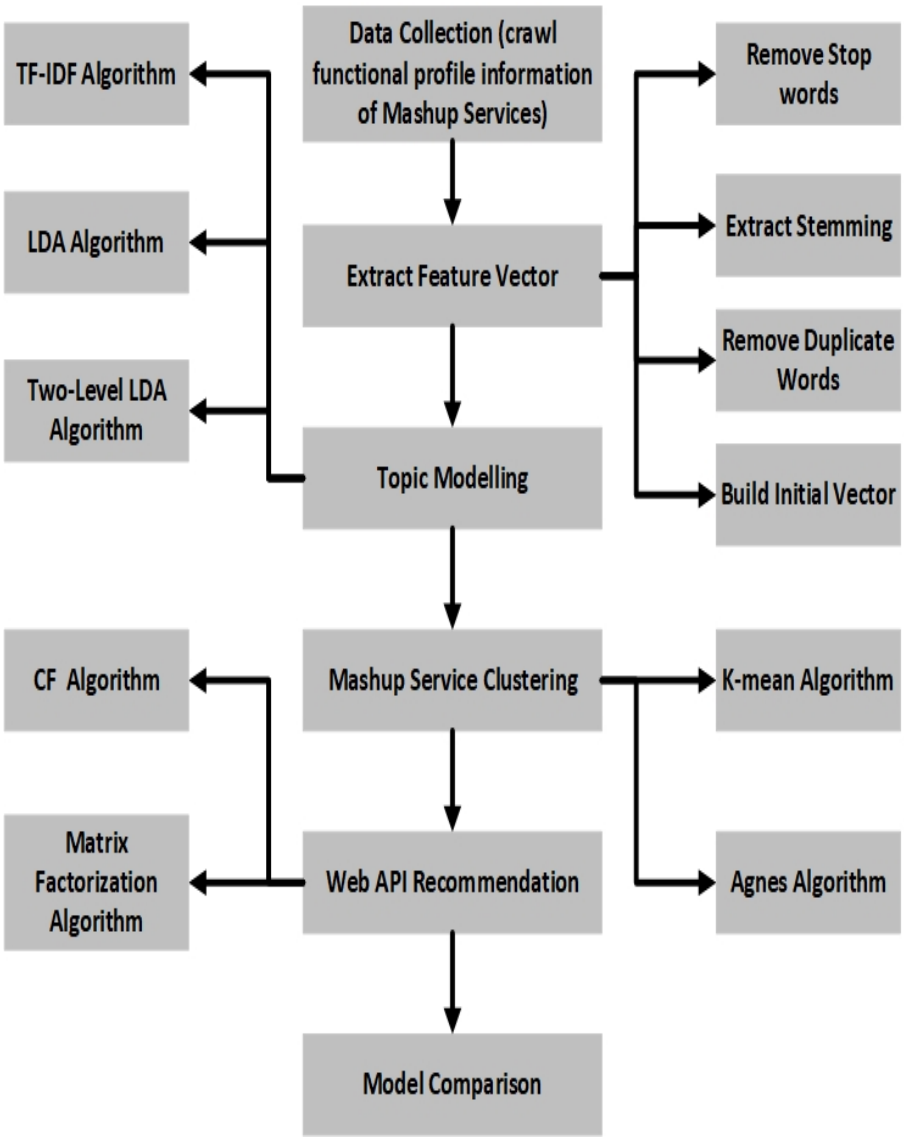

Figure 3.1: Flow Chart of Recommender System

tf-idf algorithm, the LDA model, and the two-level LDA model are used for topic modeling to obtain sparse vector of data set; K-mean algorithm and Agnes algorithm are used for unsupervised clustering learning to achieve the classification of data set; The CF recommendation algorithm and Matrix Factorization recommendation algorithm are used for Web APIs recommendation.

3.3 Data Sources and Data Collection

With the rapid development of Web 2.0 technology and service oriented, Web Application Programming Interfaces (APIs) and Mashup services are more and more widely used in Web development, resulting in more and more Web API repositories emerged. In recent years, the three most famous API repositories are ProgrammableWeb, myExperiment, and Biocatalogue, which the ProgrammableWeb has the largest number of Web APIs and Mashup services (Gao, Chen, Wu & Gao, 2015). Therefore, we adopt the detail of Mahsup services and Web APIs in ProgrammableWeb as the data sets of our data mining models.

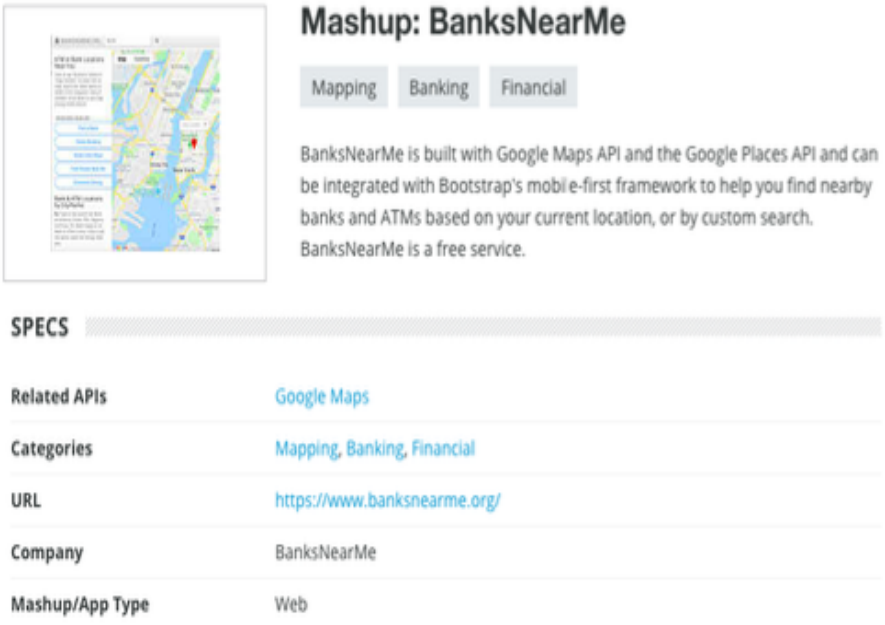

Figure 3.2: An example of Mashup service

The figure 3.2 shows an example of Mashup service named 'BankNearMe' on ProgrammableWeb site, which has three tags, one description content, one related APIs, three categories, one URL link, one affiliated company, and one Mashup type. The Web APIs detail are similar to Mashup services on ProgrammableWeb. We crawled 6343 real Mashup sercice and their related data, and 17800 real Web APIs from the

ProgrammableWeb site by using the method of data crawling in Python. Among them, the data set of Mashup services are mainly used for model training and data mining, while the data set of Web APIs are mainly used for corresponding parameters with Mashup services.

The data set of Mashup services and Web APIs were crawled from ProgrammableWeb site in September 2018. From the data set of Mashup service, we know that the total amount of 6343 Mashup services in 319 categories, and the average number of each category is 19.88. It can be observed that the number difference of different Mashup service categories is very large, such as the Mapping category has 1041 Mashup services, while the Browsers category only has 1 Mashup service. Accoding to statistics, the top 20 categories involve 3999 Mashup services, which account for 63 percent of the total Mashup services in the data set; it has 273 categories involve less than 20 Mashup services, which account for 21 percent of total Mashup services amount in the data set; and it has 81 categories contain only 1 Mashup Service. The amounts distribution of Mashup Services in top 20 categories are shown in Table 3.1, which more than 60 Mashup services in each of category.

3.4 Pre-processing of Data Set

In the field of data mining, the quality of training data set has a decisive influence on the results of machine learning. Pre-processing of data set be able to effectively improve the quality of training data set, which conduce to improve the accuracy and efficiency of training results. High-quality decisions have to rely on high-quality data, thus, pre-processing of data set is an essential step in the data mining process. The pre-process of data set is the process of converting raw data into a form suitable for machine learning, which involves assemble data set, remove stop words, extract stemming, and remove duplicate words. The flow chart of pre-processing of data set are shown in Figure 3.3.

Table 3.1: The Distribution of Mahsup Services in Top 20 Categories

Category	**Number of Mashup Service**	**Category**	**Number of Mashup Service**
Mapping	1041	Sports	119
Search	310	News Services	117
Social	299	Telephony	99
eCommerce	299	Reference	99
Photos	259	Blogging	98
Music	256	Electronic Signature	95
Video	176	Widgets	86
Travel	173	Visualizations	77
Messaging	132	Humor	69
Mobile	127	Real Estate	68

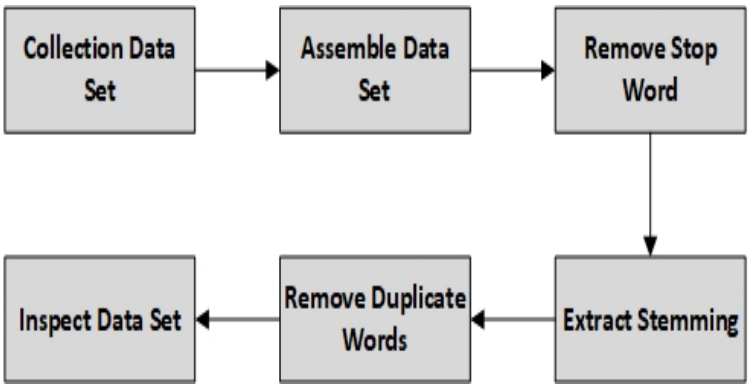

Figure 3.3: The flow chart of data pre-processing

3.4.1 LDA Section

1. Assemble Data Set

   We adopt the Java Excel API (JXL) package to assemble the original data set into an excel file, in order to obtain the Mashup name, Category, Tags, and Description parts from original data set, and save it in an excel file for preparing remove stop words.

2. Remove Stop Word

   This step is removing stop word, punctuation, and special characters by using a stop word document. And save it into a text-file. In the experiment, we adopt FileInputSteam function to read assemble.xls data set file, then use BuffereReader function and InputSteamReader function to read the stop words list, then filter the stop words, Arabic numerals, redundant spaces, special symbols in data set arrays, and finally use FileOutStream function to write the results into a text file.

3. Extract Stemming

   We implement the Porter Stemming Algorithm for extract stemming. The code of Porter Stemming Algorithm is refer the official site:

   https://tartarus.org/martin/PorterStemmer/. The steps of extract stemming be able to be divide into 6 steps. The first setp is to get rid of plurals, as well as the words ending with -ed and -ing. The second step is to turn terminal 'y' to 'i' when there is another vowel in the stem. The third step is to map double suffixes to single ones, such as the words ending with -ational to -ate. The fourth step is to deal with prefixes and suffixes of -ic-, -full, -ness, -ative, -al-. The fifth step is to take off the suffixes of -ant, -ence, -al, -er, -able, -ement, -ment, -ent, -ou, -ism, -ate, -iti, -ous, -ive, -ize. The sixth step to remove the suffixes of -e, if return value is greater than 1 in the above steps.

4. Remove Duplicate Words

   In this part, we input the result of extract stemming, and then remove duplicate words by using Hashset class and resort words by using Treeset class in Java. (note that the data saved by HashSet is unordered, and the data saved by Treeset is ordered, so is we want to save the data in order, we need to use the TreeSet class after HashSet class.) In additional, we also use a loop method to check whether it has non-alphabetic characters.

3.4.2 TF-IDF Section

The pre-processing part of TF-IDF is similar with LDA, while it is more simple. It can be divided into 3 steps: concatenate each of Mashup document into one string, and separated by && symbol between words; remove duplicate words by using Treeset class (it does not need to use HashSet class) in Java; Remove the stop words. At the end of pre-processing, we save the keyword of each mashup document in a text-file.

3.5 Topic Modelling

In this section, we obtain the LDA data set for the LDA model and TF-IDF model by using java function, which are BufferedRead function, FileReader function, IOException function, ArrayList function, Iterator function, and TreeMap function. The LDA model and Gibbs Sampling is adopting com.aliasi.cluster.LatentDirichletAllocation class in Java.

3.5.1 Two-Dimension Array of LDA

The LDA model in our experiment is to invoke the com.aliasi.cluster.
LatentDirichletAllocation class in Java. The LDa model provides a Bayesian model

of document generation where each document is generated by a mixture of topical multinomials. LDA model specifies the number of topic, which plays a crucial factor in a Dirichlet prior over topic mixtures for a document, and a discrete distribution over words for each topic.

In the process of document generation by LDA model, a multinomial over topics is first selected for given the Dirichlet prior. Then LDA model given a token for each word, a topic is generated from the document-specific topic distribution for each token in the document. And then a word is generated from the discrete distribution for that topic. Note that a token represent a word in the Bag of Word.

In the two-dimension array LDA model, it require two-dimension array of documents and words. In each Mashup document involves a fixed vocabulary of discrete outcomes, which represent as a set of integers:

$word = 0, 1, 2, ..., numWords - 1$

The two-dimension array is a LDA corpus which is a ragged array of documents, the array is each document consisting of an array of words:

$int[][]words = words[1], words[2], words[3], ..., words[numDocs - 1]$

Therefore, aims at a given document words[i], and i less then number of documents. The words[i] represent a bag of word in the document i, and each word being represent as an integer:

$int[]words[i] = words[i][0], words[i][1], ..., words[i][words[i].length - 1].$

In the LDA model, we given the number of topic, thus, the topics are also represented as integers:

$topics = 0, 1, 2, ..., numTopics - 1$

The LDA model specifies a discrete distribution over words, which represent as:

$\phi[topic][word]$

Among then, discrete distribution of topic is represent as:

$\phi[topic]$

In addition, topic of each document in the corpus is generated from a document-specific mixture of topic, which represent as:
$\theta[doc]$
the topic distribution of each document is also a discrete distribution over tpoics:
$\theta[doc][topic]$.

There are three parameters need to be given before LDA model training, and they play important role in probability generation of topic, which are β, α, and number of topics. This leaves two Dirichlet priors, one parameterized by α for topics in documents, and one parameterized by β for words in topics. We set the β = 0.01, α = 0.1, num_topics = 20. The LDA generative model process is: generate the topic distribution $\phi[topic]$ from a Dirichlet parameter $Dirichlet(\beta)$; generate the documents topic distribution $\theta[doc]$ from a Dirichlet parameter $Dirichlet(\alpha)$; generate topic distribution of each document $topics[doc][token]$ from $Discrete(\theta[doc])$; generate topic distribution of each word $words[doc][token]$ from $Discrete(\phi[topics[doc][token]])$.

Next, we discuss the mathematical representation for the derivation process of LDA model generation. The topic distribution are chosen at the corpus level for each topic given their Dirichlet prior, and remaining variables for generating topic distribution:

$$p(words, topics, \theta, \phi|\alpha, \beta) = p(\phi[topic]|\beta) * p(words, topics, \theta|\alpha, \phi) \quad (3.1)$$

Among then $\phi[topic]$ represents a continuous multinomial parameters;
$p(\phi[topic]|\beta)$ represents a density, which not a discrete distribution.
The topics and words of documents are generating, after generated topic distribution by given the Dirichlet prior. The mathematical representation for generating of topics and words of document:

$$p(words, topics, \theta|\alpha, \phi) = p(\theta[doc]|\alpha) * p(words[doc], topics[doc]|\theta[doc], \phi) \quad (3.2)$$

Among then topics and words are generated from the multinomial $\theta[doc]$, and the topic distribution ϕ is used the chain rule. Then, next mathematical representation is generating the words by given the topic word's distribution:

$$p(words[doc], topics[doc]|\theta[doc], \phi) = \\ p(topics[doc][token]|\theta[doc]) * p(words[doc][token]|\phi[topics[doc][token]]) \tag{3.3}$$

Finally, the mathematical representation is generating topic distribution of each word by given the topic and document multinomials, which called 'marginalized', or 'collapsing', or 'integrating out':

$$p(words[doc]|\theta[doc], \phi) = p(topic|\theta[doc]) * p(word[doc][token]|\phi[topic]) \tag{3.4}$$

3.5.2 LDA with Gibbs Sampling

The LDA need a collapsed form over the posterior distribution of topic, and the Gibbs Sampling is the most common mainstream method in LDA model. In the experiment, we also adpot the Gibbs Sampling to assignment given the documents and Dirichlet priors. The specific focus on the final result of the two-Dimension array of LDA, which the mathematical representation is:

$$p(topics|words, \alpha, \beta) \tag{3.5}$$

Initial samples are generated by randomly assigning topics to token. Then, Gibbs sampler iterates though the corpus, once for each token, and it samples a new topic

assignment to each token. The process represent as:

$$p(topics[doc][token]|words, topic') \tag{3.6}$$

Note that topic' represents the set of topic assignments other than to topics[doc][token].

The collapsed posterior conditional is estimated directly though the mathematical equation:

$$p(topics[doc][token] = topic|words, topic') = \frac{(count'(doc, topic) + \alpha)}{docLength[doc] - 1 + numTopics * \alpha} * \frac{count'(topic, word) + \beta}{count'(topic) + numWord * \beta} \tag{3.7}$$

Note that the counts are all defined relative to topic'. For the posterior Dirichlet distribution be able to be computed using only the counts.

The posterior Dirichlet distribution for topics in documents is estimated as:

$$p(\theta[doc]|\alpha, \beta, words) = Dirichlet(count(doc, 0) + \beta, count(doc, 1) + \beta, ..., count(doc, numTopics - 1) + \beta) \tag{3.8}$$

The Gabbs sampling distribution is defined from the maximum a posteriori (MAP), which estimates of the multinomial distribution over topics in a document:

$$\theta[doc] = ARGMAX_{\theta[doc]} p(\theta[doc]|\alpha, \beta, words) \tag{3.9}$$

Which the discrete Dirichlet distribution over topics is:

$$\theta[doc][topic] = \frac{count(doc, topic) + \alpha}{docLength[doc] + numTopic * \alpha} \tag{3.10}$$

Similar as Dirichlet distribution over topoics, the maximum a posteriori (MAP)

word distribution in topics is:

$$\phi[topic][word] = \frac{count(topic, word) + \beta}{count(topic) + numWords * \beta} \tag{3.11}$$

Gibbs sampling algorithm in LDA model, first randomly assigns topics to words, then iterates though the tokens in turn, and samples topics according to the distribution of prior definitions. After each run of the whole corpus, the sampler redefine the topic distribution of each words. Sampler sampling the specified times, according to the code setting. In the experiment, we set the number of sample is 20.

3.5.3 Two Level LDA Model

The two level LDA model is adopted in our experiment process, which the two level are content level and network level. The final topic vector of each Mashup service document involvs two parts: topic probability distribution from content, and topic probability distribution from the network weight of Mashup service documents. Among then, the network weight of Mashup service documents are caculated by the Jaccard similarity coefficient, the formula of the weight of network is (Cao et al., 2017):

$$W(MS_i, MS_j) = \lambda * \left| \frac{API(MS_i) \bigcap API(MS_j)}{API(MS_i) \bigcup API(MS_j)} \right| \tag{3.12}$$

Here, $API(MS_i)$ and $API(MS_j)$ are Web API set that invoked by MS_i and MS_j; MS_i and MS_j represent random Mashup service document respectively; the λ is a parameter of user's preference.

The flow chart of two level lda model are shown in Figure 3.4 (Cao et al., 2017).

In the flow chart, MS represents the Mashup service documents; L_{MS} is the network of Mashup service documents; t is the topic distribution of words in content level, z is the topic distribution of words in network level.

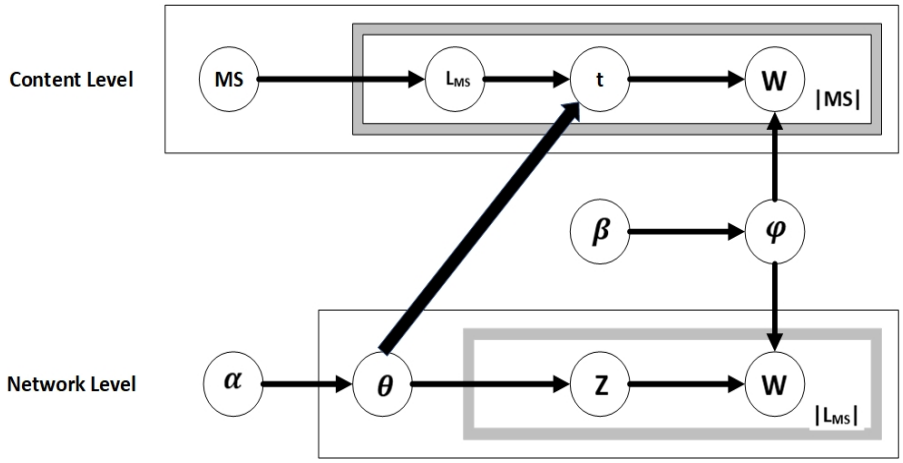

Figure 3.4: The flow chart of Two Level LDA

3.5.4 TF-IDF Model

Term Frequency - Inverse Document Frequency (TF-IDF) is a common modeling technology in the field of information retrieveal and data mining. Since its simplicity and efficiency, it is often used to extract the vector of keyword from documents. The TF-IDF is also one of the method for topic vector extraction in this paper. In the experiment, TF-IDF be able to be divided into 3 part: term frequency, inverse document frequency, and TF-IDF weight.

1. Term Frequency

   Term frequency calculates the occurrences times of term in the document, the formula of term frequency in the experiment is (Gao et al., 2015):

   $$TF(t_{ij}, MSDoc_i) = \frac{frequency(t_{ij}, MSDoc_i)}{|MSDoc_i|} \tag{3.13}$$

   Among them, $MSDoc_i$ represents the i_th Mashup service document; t_{ij} represents j_th word in the i_th Mashup service document; $|MSDoc_i|$ represents the number of words in $MSDoc_i$.

2. Inverse document frequency

The core idea of inverse document frequency (IDF) is that if there are fewer documents containing t_{ij}, and the frequency of t_{ij} appering in $MSDoc_i$ is higher, then the greater the value of IDF which indicating that t_{ij} has a good ability to distinguish topics. If t_{ij} appears in more documents, the value of IDF is smaller. The formula of IDF is (Gao et al., 2015):

$$IDF(t_{ij}, MSDoc) = lb\frac{|MSDoc|}{|MSDoc : t_{ij} \in MSDoc_i| + 1} \tag{3.14}$$

Among them, lb represents to obtain logarithmic values; $|MSDoc|$ represents the number of Mashup service documents; $|MSDoc : t_{ij} \in MSDoc_i|$ represents the number of Mashup service documents containing t_{ij} which appears in $MSDoc_i$. The reseason for plusing 1 in denominator is to avoid 0 in denominator, which is all documents do not contain the word.

3. TF-IDF weight

TF-IDF weight value is the number of times a word appears in a document multiplied by the number of times the word appers in the corpus. The formula of TF-IDF weight is (Gao et al., 2015):

$$W_ij = TF(t_{ij}, MSDoc_i) * IDF(t_{ij}, MSDoc) \tag{3.15}$$

In additional, each document has t_m words, the topic vector of each word is:

$$Word_{Vector} = \{(t_1, W_{i,1}), (t_2, W_{i,2}), ..., (t_m, W_{i,m})\} \tag{3.16}$$

3.6 Mashup Service Clustering

This section performs unsupervised clustering on the Mashup topic vectors from the topic modelling. Firstly, we adopt Jensen-Shannon Divergence to cacluate the similarity between each Mashup topic vectors node, and a matrix is generated to represent the distance between M_i and M_j. Then the average distance between M_i and M_j are calculated, and use the K-means algorithm to generate the clusters of Mashup services. Finally, the Agnes algorithm is adopted based on the result of K-mean clusters, in order to reduce the amount of clusters and improve the accuracy of the Mashup classification. Among them, the role of the Jensen0SHannon Divergence is to provide a average similarity set for each Mashup service which support for K-means clustering, and the Agnes clustering algorithm is to optimize clustering results of K-means.

3.6.1 Jensen-Shannon Divergence

1. Working Principle of JS Divergence

   Jensen-Shannon (JS) Divergence is based on the improvement of Kullback-Leibler (KL) algorithm, that is, JS Divergence sovles the shortcoming of asymmetric divergence of KL Divergence. The JS Divergence algorithm and KL Divergence algorithm are consistent in calculating the vector probability distribution between two Mashup Service nodes.

   KL Divergence is also called relative entropy, information divergence, and information gain. It measures the divergence of probability distribution between $Mashup_i$ and $Mashup_j$. Note that the $Mashup_i$ and $Mashup_j$ are random Mashup service nodes. The formula of KL Diverence is:

$$D_{KL}(MS_i, MS_j) = \sum_{t=1}^{T} p_t ln \frac{p_t}{q_t} \tag{3.17}$$

Note that $D_KL(MS_i, MS_j)$ represent KL divergence between Mashup services M_i and Mashup service M_j; t represent a variable shows common topics in MS_i and MS_j; T represent the total amount of common topics in MS_i and MS_j; p_t represent the probability of topic t in MS_i; q_t represent the probability of topic t in MS_j.

JS Divergence is also an algorithm to measure the divergence of probability distribution between MS_i and MS_j, which achieve the asymmetric between Mashups. In the original data, the interrelationship of two topics in Mashups is asymmetric, but the $D_{KL}(MS_i, MS_j)$ is not equal to the $D_{KL}(MS_j, MS_i)$. Thus, the JS Divergence is more practical for calculating the divergence of topic probability between Mashups. The formula of JS Divergence in Mashup services is:

$$D_JS(MS_i, MS_j) = \frac{1}{2}[D_{KL}(MS_i, \frac{MS_i + MS_j}{2}) + D_{KL}(MS_j, \frac{MS_i + MS_j}{2})] \tag{3.18}$$

In the Mashup service similarity matrix, it compose of an average similarity set (ASS) for each Mashup service topic vector node. For the average similarity of MS_i, it represent the average similarity to all Mashup service topic vector nodes, which the formula is:

$$AverageSim(MS_i) = \frac{\sum_{j=1}^{N} D_JS(MS_i, MS_j)}{N} \tag{3.19}$$

2. the programming of JS Divergence

Fistly, the topic distribution results of LDA model are sealed off a corresponding number a corresponding number of Mashup text; then, the topic distribution results are encapsulated into the corresponding DOcBean program; and then,

the JS divergence of each DocBean entity are calculated by using JS algorithm; finally, the Average Similarity Set (ASS) of JS Divergence are encapsulated into a program entity. The experimental result are waiting for the call of clustering experiment. In additional, the java.util.List function is used to construct the average similarity set of the Mshup service nodes, and the java.util.ArrayList function are used to invoke JS Divergence of Mashup service topic vector nodes.

3.6.2 K-means Clustering Algorithm

The K-means algorithm is an iterative clustering process, it consists of 3 steps: 1) K-mean cluster centroids in data space are random selected as initial centers, and each cluster centroids represents a clustering center; 2) Each Mashup is assigned to the nearest clustering center, according to JS Divergence; 3) The average value of each clustering are taken for updating the cluster centers. Note that the number of K-mean cluster centroids are depending on the average value of the Average Similarity Set (ASS).

The process of programming be able to be divided into 4 steps: 1) Obtain the ASS of JS divergence matrix by java.util.ArrayList function; 2) set the average value of the ASS as the initial threshold of K-means clustering; 3) divide MS_i into cluster centroids, and remove the MS_i from ASS; 4) divide MS_j into cluster centroids, and remove the MS_j from ASS.

3.6.3 Agnes Clustering Algorithm

Agnes clustering algorithm as the representative of Agglomerative Nesting in Hierarchical clustering. Agnes algorithm first treats each object as a cluster, then merges these cluster into larger and large clusters until some termination condition is satisfied. In this section, the Agnes clustering algorithm is based on the result of the K-means clustering.

The formula of the similarity between the K-means clusters is:

$$Sim(M_x, M_y) = \frac{\sum_{p=1}^{P} \sum_{q=1}^{Q} sim(MS_p, MS_q)}{P * Q} \tag{3.20}$$

Among them, M_x and M_y represent an random Mashup cluster; P and Q represent the number of Mashup service in M_x and M_y respectively; p and q are elements in the data set of P and Q.

In the experiment process, we input arrays of K-means cluster, then set the threshold of Agnes algorithm, and then calculate the similarity between K-means clusters which refer the formula of $Sim(M_x, M_y)$, next, constrct similarity matrix by matrix element $Sim(M_x, M_y)$. If the similarity value between M_x and M_y are greater than or equal to the threshold, then merge clusters. The purpose of executing Agnes algorithm on K-means algorithm is to optimize and merge K-means cluster results.

3.7 Web APIs Recommendation

3.7.1 Web APIs Recommendation based on CF Algorithm

We can suppose that Mashup services are the user in the recommender system, and Web APIs are the item in the recommender system. We sort out the invoked frequency of Web APIs in Mashup services as training data, according to Web APIs usage history in Mashup Services. In this section, we be able to calculate the interest degree of uninvoked Web APIS in Mashup Services, according to the CF algorithm.

1. Personalized Similarity

   In the Web APIs recommadation based on CF algorithm, the Persona Correlation Coefficient (PCC) and Tanimoto Coefficient are used to calculate similarity in recommender system. Item-based CF algorithm adopts PCC to calculate

the similarity between Web APIs, and Tanimoto Coefiicient to caluculate the similarity between Mashup Services.

The formula of PCC in our recommender system can be denoted:

$$sim(WA_i, WA_j) = \frac{\sum_{m \in M}(r_{m,WA_i} - \overline{r_{WA_i}})(r_{m,WA_j} - \overline{r_{WA_j}})}{\sqrt{\sum_{m \in M}(r_{m,WA_i} - \overline{r_{WA_i}})^2}\sqrt{\sum_{m \in M}(r_{m,WA_j} - \overline{r_{WA_j}})^2}} \tag{3.21}$$

Note that M represents a set of Mashup service that invoked both WA_i and WA_j; r_{m,WA_i} and r_{m,WA_j} represent the frequency of Mashup m historical invoked WA_i and WA_j; $\overline{r_{WA_i}}$ and $\overline{r_{WA_j}}$ represent average frequency values of WA_i and WA_j invoked by different Mashup.

Tanimoto Coeflicient is a method of measuring the similarity between to sets, which can be used to calculate similarity between users in Recommender system. The formula of Tanimoto Coeflicient be able to be denoted:

$$sim(m_i, m_j) = \frac{\sum_{m=1}^{M} m_i * m_j}{\sum_{m=1}^{M} {m_i}^2 + \sum_{m=1}^{M} {m_j}^2 - \sum_{(} m = 1)^M m_i * m_j} \tag{3.22}$$

2. Missing Value Prediction

We be able to obtain a WPI APIs similarity matrix and a Mashup Serives matrix, after calculating the similarity in Web APIs and Mashup Services. We will adopt top-K algorithm to identify a set of similar neighbors for Web APIs. For example, we predict the missing values of WA_i, the set of similar neighbors is $S(WA_i)$. The formula of Missing value Prediction can be denoted:

$$p(r_{m,WA_i}) = \overline{WA_i} + \frac{\sum_{WA_{ik} \in S(WA_i)} sim(WA_{ik}, WA_i)(r_{m,WA_{ik}} - \overline{WA_{ik}})}{\sum_{WA_{j_k} \in S(WA_i)} sim(WA_{ik}, WA_i)} \tag{3.23}$$

Note that $\overline{WA_i}$ represents an average frequency value of the target Web APIs item WA_i invoked by different Mashup; $\overline{WA_{ik}}$ is a average frequency value of

the similar Web APIs item WA_{ik} invoked by different Mashups.

3.7.2 Web APIs Recommendation based on Popularity

The Popularity based recommendation algorithm according to the number of Web APIs in each Mashup service cluster and recommends the results to uses by the popularity rank. In our experiment, we recommend Web APIs which are most popular in the matching Mashup cluster for user's requirement. The popularity of Web APIs is calculated by the number of Mashup service in the cluster which contains this Web API. We match the cluster associated with the user requirement, and then recommend a popularity-based recommendation list.

3.7.3 Web APIs Recommendation based on Matrix Factorization

Probabilistic Matrix Factorization (PMF) algorithm is one of the mainstream algorithm in modern recommender system, which brings in probability model on the basis of Regularized Matrix Factorization (RMF). In this paper, PMF algorithm is introduced into the recommender systems to compare the effect with PCC based collaborative filtering algorithm. Same as the PCC based collaborative filtering algorithm, the PMF also processes the reuslt of Mashup service cluster for recommendation.

Since the large number of Web APIs in each custer, the Mashup document clusters as users and the Web APIs as items in the PMF recommendation model. We recommended top Web APIs for each Mashup document cluster in the recommendation model. Assume that we have a set of Mashup clusters A and a set of Web APIs B, and introduce a dimension of latent feature D. The $A * B$ perference matrix R is given by the product of an $A * D$ user coefficient matrix U^T and a $D * B$ factor matrix V. In order to predict a missing value corresponding to each Mashup cluster A_i and Web APIs B_j, we need

to user the dot product of two vectors based on the following formula:

$$R = U^T * V = \sum_{d=1}^{D} A_{id} * B_{jd} \tag{3.24}$$

The PMF recommendation model assumes that the element R_{ij} in the predict matrix R is determined by the inner product of the latent preference vector U_i of user and the latent attribute vector V_j of the item, the expression of predict matrix R in probability is:

$$R_{ij} = A(U_i^T V_j, \sigma^2) \tag{3.25}$$

Among then, σ^2 represent the variance in Gaussian distribution. Therefore, the predict matrix of the conditional distribution over the observed rating on the following formula:

$$p(R|U, V, \sigma^2) = \prod_{i=1}^{A} \prod_{j=1}^{B} [A(R_{ij}|U_i^T V_j, \sigma^2)]^{I_{ij}} \tag{3.26}$$

Note that $A(R_{ij}|U_i^T V_j, \sigma^2)$ is the probability density function, I_{ij} is the indicator function that is equal to 1 if the R_{ij} is observed, and equal to 0 if the R_{ij} is not observed.

In the PMF recommendation model, the logistic function is added based on the Gaussian function, in order to limit the range of user scoring which limits the value of matrix vectors in $[-1, 1]$. The formula of logistic function on the following formula:

$$g(x) = \frac{1}{1 + exp(-x)} \tag{3.27}$$

Therefore, the final equation of predict scored in the predict matrix R is:

$$p(R|U, V, \sigma^2) = \prod_{i=1}^{A} \prod_{j=1}^{B} [A(R_{ij}|g(U_i^T V_j), \sigma^2)]^{I_{ij}} \tag{3.28}$$

3.8 Evaluation Metrics

The quality of recommender system be able to be evaluated from two metrics: the quality of recommendation ranking and the diversity of recommendation content. The quality of of recommendation ranking is reflected in the accuracy and relevance of recommendation, while the diversity of recommendation content is reflected in the inclusion of popular Web APIs and unpopular Web APIs. In this section, we adopts two method corresponding to these two metrics, which are Discounted Cumulative Gain (DCG) algorithm and Hamming Distance (HMD). The DCG evaluates the quality of recommendation ranking, while HMD evaluates the diversity of recommendation content (Cao et al., 2017).

1. Discounted Cumulative Gain

   DCG is an effective measure of recommendation ranking quality, which often used to measure the effectiveness of network search algorithm or related applications. DCG measures the validity of the recommendation results, according to the position of Web APIs in the result list. The evaluation of recommendation ranking based on DCG has two criteria: 1) the higher the top Web APIs has the higher the validity; 2) The high relevant Web APIs are more valuable than slightly related Web APIs, while slightly related documents are more valuable than unrelated documents.

   The DCG algorithm is developed by Cumulative Gain (CG) algorithm, while CG algorithm does not consider the ranking of recommended results when calculating the practicability of the result set. The formula of CG algorithm is:

$$CG = \sum_{i=1}^{R} r(i) \tag{3.29}$$

   Note that i is the rank position in the rank results of Web APIs recommendation

list; $r(i)$ represents the relevant score of the i_th rank position on the rank results of Web APIs recommendation list, and the value of $r(i)$ is between 0 and 1; R is the number of Web APIs in the recommendation list.

DCG algorithm adds the hierarchical correlation of recommendation ranking on the basis of CG algorithm, which the presece of highly relevant Web APIs in the recommendation result list at lower ranking position will reduce the DCG value. The formula of DCG is shown in below:

$$DCG = \sum_{t=1}^{R} \frac{2^{r(i)} - 1}{\log_2(1+i)} \tag{3.30}$$

2. Hamming Distance

HMD detects the diversity of recommended result set by calculating the similarity of topic vectors, which the higher the similarity of topic vectors has the smaller the HMD distance. Thus, popular Web APIs and unpopular Web APIs are evaluated under the same condition, which be able to intuitively detect the diversity of recommendation result set. The formula of HMD algorithm in our experiment is:

$$HMD(m_i, m_j) = 1 - \frac{Q(m_i, M_j)}{R} \tag{3.31}$$

Here, m_i and m_j represent a random Mashup cluster; $Q(m_i, m_j)$ is the number of common Web APIs in the recommendation lists of Mashup service clusters m_i and m_j; R represents the number of Web APIs in the recommendation result set. If the two Web APIs recommendation results in the same cluster, then m_i equal to m_j and $H(m_i, m_j)$ equal to 0; if there are not any Web APIs in common cluster, then $H(m_i, m_j)$ equal to 1.

Chapter 4

Results

4.1 Introduction

In this chapter, we mainly introduce how to conduct experiment through appropriate methods, how to collect cluster training data and recommendation data, and what the results are. And we also discuss the advantages and disadvantages of different topic model and recommendation models. To confirm our hypothesis, a series of experiments were performed using existing tools based on the mathematical method discussed in Chapter 3.

This chapter mainly introduces the recommended results of various recommendation models. The results of the Mashup clustering model based the topic model to solve the problem of cold-start for collaborative filtering will also be introduced in detail. We will also list the comparison of the four collaborative filtering recommendation model. Finally, the experimental results are presented in the form of tables, charts and diagrams that provide common ground for further discussion.

4.2 Mashup Service Clustering

4.2.1 Topic Modelling

In the LDA section, we set the number of topics as 20, the prior parameter of topics set α = 0.1, the prior parameter of words set β = 0.01, and set the number of Gibbs sampling as 20. Since the top 20 categories involves 63 percent of the total Mashup services in the original data set, and it is difficult to obtain an ideal topic vector with a few parameters. Thus, we set the initial number of topics as 20, in order to be integrating minority categories into mainstream categories. The details of LDA model training result is shown in Table 4.1.

Table 4.1: The full report of LDA result

Parameter Content	**Initial Parameter**	**report of LDA result**	**Parameter of report**
NUM TOPICS	20	NUM Topics	20
DOC TOPIC PRIOR	0.1	NUM Tokens	149290
TOPIC WORD PRIOR	0.01	NUM Words	6128
NUM SAMPLES	20	epoch	19
docNum	6342		

We trained the original data set in the LDA model, it generates the probability distribution of each word, and then calculates the probability distribution of each Mashup Service according to the probability distribution of each token. In the probability distribution of each word, it includes words, symbol number of each word, occurrence number of each word in the topic, the parameter of probability distribution for each word in network level and content level. In the probability distribution of each Mashup

service, it involves topics of documents, topic distribution parameter of documents, occurrence number of each topic, and topic related words in the document.

For the topic distribution probability of words, we obtained 6128 words from 149290 tokens, which expressions topic distribution information. We sorted out 10 words with the highest probability distribution in first topic as an example in Table 4.2.

Table 4.2: The topic distribution probability of words in topic 0

TOPIC 0				
SYMBOL	**WORD**	**COUNT**	t	z
4258	real	309	0.052	10.8
1582	estat	193	0.032	9.0
3055	map	171	0.029	0.4
1203	deal	79	0.013	5.9
1915	free	70	0.012	3.3
1094	coupon	66	0.011	7.8
2902	list	65	0.011	1.9
2136	googl	62	0.010	0.1
4473	sale	62	0.010	4.1
196	applic	57	0.010	1.2

In additional, we sorted out the overall of the topic distribution probability of words are in Table 4.3, which involves topic number, total number of words, total count of words in documents, and the high probability distribution and representational significance of words as the keyword for each topic.

In the two level LDA model, it calculates the probability distribution of words, in order to generate the latent topic of each Mashup service document. In the experimental

Table 4.3: The Overall of the Topic Distribution Probability

Topic	Total Words	Total count	Keyword
Topic 0	459	5936	estate
Topic 1	493	7720	e-commerce
Topic 2	463	7041	travel, hotel
Topic 3	472	6438	storage, file
Topic 4	439	9441	electron signature
Topic 5	488	7165	blog, social
Topic 6	482	6659	music, voice
Topic 7	487	7303	food, search
Topic 8	480	5920	government, location
Topic 9	471	7127	visual, widget
Topic 10	490	6894	transport, message
Topic 11	482	7646	mobile, phone
Topic 12	480	8845	Facebook, friend
Topic 13	486	6994	weather
Topic 14	506	7013	event, movie
Topic 15	491	8512	video, YouTube
Topic 16	478	8624	travel, photo
Topic 17	498	8527	game, slide-show
Topic 18	471	7581	news, feed
Topic 19	484	8264	earth, park

results, we list 4 result parameters for each Mashup service document: 1) topics number that probability distribution greater than 0.01; 2) related words of each topics in the document; 3) count of related of each topics; 4) the parameters of probability distribution. We list the first three LDA results of Mashup service documents latent probability distribution as an example in Figure 4.1. In the full Mashup service documents latent probability distribution, it includes 6342 documents of probability distribution. These probability distribution will encapsulate in Mashup service clustering experiment as the topic vector of each Mashup service document.

In additional, we also attempt to bring the TF-IDF vector result into the JS based clustering model, but the TF-IDF model only generate the weights of vector, which does not includes data mining of latent probability distribution. Therefore, the result of

```
DOC 0
TOPIC    COUNT    PROB
----------------------
   18        8   0.623
   15        2   0.162
   14        1   0.085

pound(18) world(18) "pound(15) world(18) mashup(15)
citi(18) citi(18) visual(18) visual(18) data(18) rate(14)

DOC 1
TOPIC    COUNT    PROB
----------------------
   13       11   0.483
   16        4   0.178
   17        4   0.178
   18        2   0.091

pictur(13) area(18) guess(17) humor(13) mashup(18) guess(17)
geoguessr(13) look(13) game(17) world(16) look(13) geoguessr(13)
look(13) explor(13) map(16) "pound(16) "pound(13) map(16) googl(17)

DOC 2
TOPIC    COUNT    PROB
----------------------
   17       18   0.431
    3       11   0.264
   11        6   0.145
   12        5   0.121

abiq(3) access(3) live(17) video(17) fantasi(12) data(11) stream(17) sport(17)
mashup(12) abiq(3) api(17) mashup(3) sport(17) game(17) applic(3) desktop(12) stream(17)
desktop(3) "pound(3) databas(11) viewer(3) stream(17) gambl(11) stream(17) plai(17) viewer(3)
```

Figure 4.1: An Example of Mashup Service Document Latent Probability Distribution

the JS based clustering model with TF-IDF clustering, which the number of clusters are always same with the number of Mashup service documents. Fundamentally, the JS based clustering model is calculating the probabilities distribution of Mashup service documents, while TF-IDF model can only calculates the vector weights of each word. Thus, we determine to generate probability topic vectors by adopting LDA model.

4.2.2 Cluster Results

In the Mashup service clustering section, we first extract the topic vector model generated by the two level LDA model, then adopt JS algorithm to calculate the average distance between MS_i and MS_j, and use K-means algorithm to cluster of Mashup

service document, finally apply Agnes hierarchical clustering algorithm to cluster based on the result of K-means clusters. Among then, the JS algorithm generated the array of average distance between each Mashup service document. An example of clustering result is shown in Figure 4.2, the figure has exhibited the thresh of each clustering algorithm, the number of clusters, and the number of Mashup service document in each cluster.

```
Thresh K=0.4, obtains 21 K-means clusters
cluster 1: 121; cluster 2: 80; cluster 3: 47; cluster 4: 40;
cluster 5: 15; cluster 6: 9; cluster 7: 8; cluster 8: 8;
cluster 9: 3; cluster 10: 1; cluster 11: 6; cluster 12: 5;
cluster 13: 2; cluster 14: 3; cluster 15: 1; cluster 16: 1;
cluster 17: 1; cluster 18: 1; cluster 19: 1; cluster 20: 1;
cluster 21: 1;
==========================================
Thresh S=0.04, obtains 5 Agnes clusters
cluster 1: 125; cluster 2: 85;
cluster 3: 56; cluster 4: 43;
cluster 5: 45;
first time obtains 21 K-means clusters
second time obtains 5 Agnes clusters
```

Figure 4.2: An example of clustering result

For clustering result, we can compare the categories classification in the original Mashup service data set. Categories are the result of user manual classification, which results in a certain degree of subjectivity, while it ahs a great significance to compare them with machine learning clustering algorithm. We use different number of Mashup service documents to conduct multiple clustering learning, and compare the clustering results with corresponding categories.

In the experimental process, five different number of Mashup service documents are used for clustering training, which are whole data set of 6342 Mashup service documents with 319 categories; 3999 Mashup service documents with 20 categories;

750 Mashup service documents with 10 categories; 354 Mashup service documents with 5 categories; and 94 Mashup service documents with 1 category.

In the results of clustering training, we found that it has best cluster traingin result when the number Mashup service documents around 354. In the experiment process, first we used complete data set of 6342 Mashup service document, the final cluster number of K-means algorithm and Agnes hierarchical algorithm are both 3016. It has 1685 clusters only involve 1 Mashup service document, which the training result very unsatisfactory. Then, we reduce the number of Mashup service documents to 3999 with 20 categories, the final cluster number of K-means algorithm and Agnes algorithm are both 1818, which the result still not ideal. Subsequently, we reduce the number of Mashup documents to 750 with 10 categories, and get that the number of K-means clusters are 42, while the number of Agnes clusters are 41. So far, the result of cluster training have been greatly improved. Then, we reduce the number of Mashup service documents to 354 with 5 categories, and get that the number of K-means clusters are 21, while the number of Agnes clusters are 5. Compared with the original data set, we found that when the number of Mashup service documents is 354, the clusters content are close to the categories content. Finally, we reduce the number of Mashup service documents to 94 with 1 category, and get that the number of K-means clusters are 12, and Agnes clusters are 6. It indicates that it is difficult to achieve ideal cluster training when the data set is to small. We sort out the overall of JS based cluster with LDA model in Table 4.4.

Table 4.4: The overall of JS based Cluster with LDA model

Number of document	Number of categories	K-means Clusters	Agnes clusters
6342	319	3016	3016
3999	20	1818	1818
750	10	42	41
354	5	21	5
94	1	12	6

4.3 Problem Definition of Recommendation

4.3.1 User-item Matrix

The relationship matrix between users and items is the decisive factor of recommendation algorithm based on collaborative filtering. In our experiment of recommendation algorithm, we adopt clusters by JS based clustering algorithm as the users, and the Web APIs of each Mashup service document as items. Thus, the user set be able to be represented as $K = M_1, M_2, ..., M_k$, and the item set can be represented as $N = WA_1, WA_2, ..., WA_n$. Among then, K is the data set of clusters, k is the number of clusters, M represent each of cluster, N is the data set of Web APIs, n is the number of Web APIs, WA represent each of Web API. The relationship matrix between users and items is the invocation relationship between clusters of Mashup service documents and Web APIs. The invocation relationship matrix be able to represented as $R = [r_{ij}]_{K*N}$, which the R is the invocation relationship matrix, the r_{ij} is an random value of matrix.

In the matrix R, the value of r_{ij} is a normalized popularity of an WA in an M, and

its value is between 0 and 1. If all Mashup service documents in M_i do not invoked WA_j, then r_{ij} is 0; if the the Mashup service documents M_i only invoked WA_j, then r_{ij} is 1. While the users in our experiment are the clusters of Mashup service document, and all of users will not only invoked 1 Web APIs, so the value of r_{ij} is hardly as 1. We randomly selected five users and items in data set as an example to demonstrate the matrix R in Table 4.5.

Table 4.5: An Example of Matrix R

	Google Maps	**Google Maps Flash**	**Microsoft Bing Maps**	**YahooMaps**	**Amazon A9Open Search**
mapping	0.4345	0.0339	0.0427	0.0271	0.0375
search	0.0538	0.0313	0.1115	0.0263	0.0363
social	0.0356	0.0011	0.0034	0.0045	0.0057
shopping	0.0759	0.0607	0.0455	0.0334	0.0759
photo	0.0024	0	0.0012	0	0

4.3.2 Define User Preferences

For defining user preferences, we provide a pro-processed user rating for the parameter r_{ij} of 0 in user-item matrix R. The parameters of user preferences denpend on the values of λ, and we set the values of λ to be 0, 0.3, 0.5, 0.7, and 1 respectively. In additional, the parameter of λ is between 0 and 1. After uesr preference setting, all parameters in matrix R are not be 0, which may get better Web APIs recommendation effect. And we will demonstrate the specific recommendation effect for different user

preference parameter settings. The formula of user preferences is:

$$userpreference = \lambda * item + (1 - \lambda) * user \tag{4.1}$$

In the process of calculating user preferences, we adopt the EstimatePrefence function in the Mahout framework. The function will find the parameter of the invocation relation of 0 in the user-item matrix R, and then gives a specific prior score for the value by the user preferences formula. The Estimateprefence function first dectects whether each parameter of the matrix is 0, and then executes the user-preference formula if the parameter is 0. Assuming that we set the parameter of λ to 0.3, the Web API of Google Maps Flash (which in Table 4.5) has 0.7 user-preference score in Mashup cluster of Photo; while the user's socre of this Web APIs still have a great deal of difference with other paramter in matrix R, which might more conducive to user score prediction.

4.4 Web APIs Recommendation

In the Web APIs recommendation experimental process, we adopt 4 recommendation model to compare and analysis data, which are User-item based Recommendation model, Popularity based Recommendation, User-item with popularity based Recommendation Model, and Probabilistic Matrix Factorization (PMF) based Recommendation model. In each model, we applies the Discounted Cumulative Gain (DCG) algorithm to measure the quality of recommendation ranking, which represents the accuracy of recommendation result based on the relevance of top-N recommended item in the recommendation list; the Hamming Distance (HMD) to calculate the diversity of recommendation lists, which represents the similarity between the recommendation lists. At the same time, we sampled the top 5, top 10, top 15, top 20, top 50, top 100 recommendation lists respectively. In additional, we set λ parameters to 0, 0.3, 0.5, 0.7, 0.9, 1 for user

preference respectively. Finally, we generate the HMD value chart and the DCG value chart for each model.

4.4.1 User-item based Recommendation Model

Data statistics show that when λ is 0, the overall evaluation results of DCG value are relatively higher; when λ is 1, the recommended lists of top 100 is very close to the evaluation results of λ is 0 in User-item based Recommendation Model. And the more the number of recommended results, the better the evaluation results. While the HMD algorithm has the highest evaluation results when λ is 0.7. And the more the recommended lists have the worse the evaluation results of HMD alorithm in User-item based Recommendation Model. Compared with DCG evaluation algorithm, HMD algorithm is less affected by the number of recommended lists and the value of λ in the User-item based Recommendation Model.

1. The evaluation result of DCG algorithm

   According to the DCG evaluation results of User-item based Recommendation model, we can sum up two rules: 1) with the increase of the user parameter λ, the overall recommendation effect gradually becomes worse, and when the DCG value of recommendation result drops to the lowest point, it increases with the increase of λ; 2) the DCG evaluation quality of recommendation result is higher, when the number of recommended lists is larger. When the number of recommended lists is from the top 5 to top 50, the DCG evaluation results are the worst with the λ set as 0.9; when the number of recommended lists increases to top 100, the DCG evaluation results are the worst with the λ set as 0.7. Therefore, the lowest point of the DCG evaluation results will gradually approach the center value of λ as the number of recommended lists increases in the User-item based

collaborative filtering. In the Table 4.6, we demonstrate the DCG result of User-item based Recommendation model. In addition, we also compiled a chart with corresponding to this table in Figure 4.3, in order to visually interpret the DCG results.

Table 4.6: The DCG result of User-item based Recommendation

	$\lambda = 0$	$\lambda = 0.3$	$\lambda = 0.5$	$\lambda = 0.7$	$\lambda = 0.9$	$\lambda = 1$
Top 5	1.46	1.21	1.03	0.82	0.69	0.71
Top 10	1.87	1.58	1.38	1.14	1.01	1.05
Top 15	2.17	1.85	1.64	1.39	1.26	1.31
Top 20	2.39	2.077	1.86	1.60	1.48	1.54
Top 50	3.28	2.99	2.79	2.54	2.47	2.58
Top 100	4.09	3.94	3.84	3.66	3.68	3.85

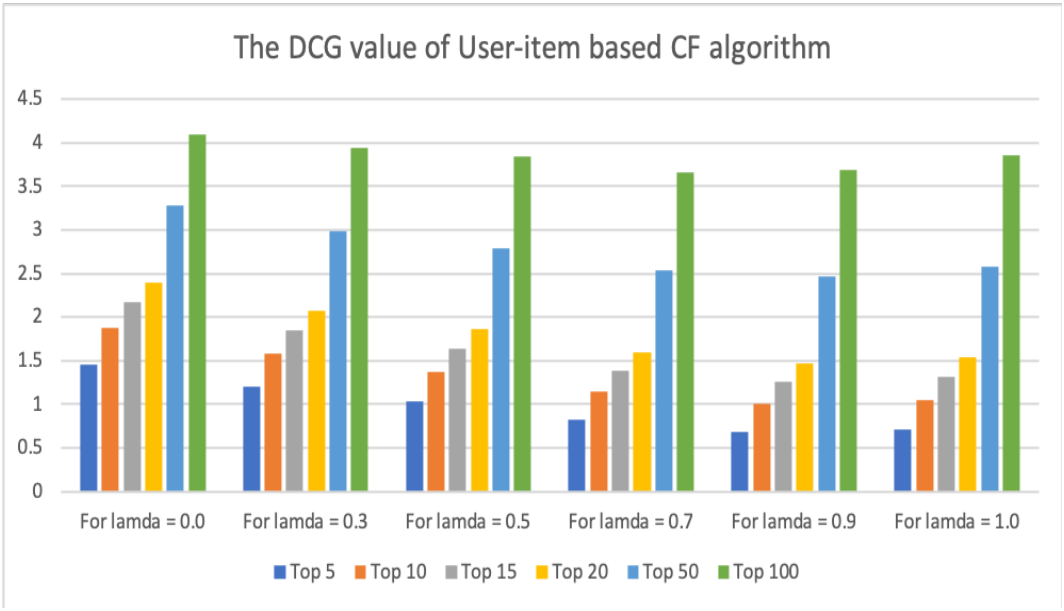

Figure 4.3: The chart of DCG result for User-item based recommendation

2. The evaluation result of HMD algorithm

Table 4.7: The HMD result of User-item based Recommendation

	$\lambda = 0$	$\lambda = 0.3$	$\lambda = 0.5$	$\lambda = 0.7$	$\lambda = 0.9$	$\lambda = 1$
Top 5	0.46	0.46	0.46	0.50	0.489	0.47
Top 10	0.44	0.45	0.46	0.47	0.48	0.47
Top 15	0.41	0.42	0.44	0.46	0.47	0.46
Top 20	0.39	0.40	0.43	0.45	0.45	0.44
Top 50	0.34	0.35	0.36	0.39	0.38	0.35
Top 100	0.29	0.30	0.31	0.33	0.32	0.31

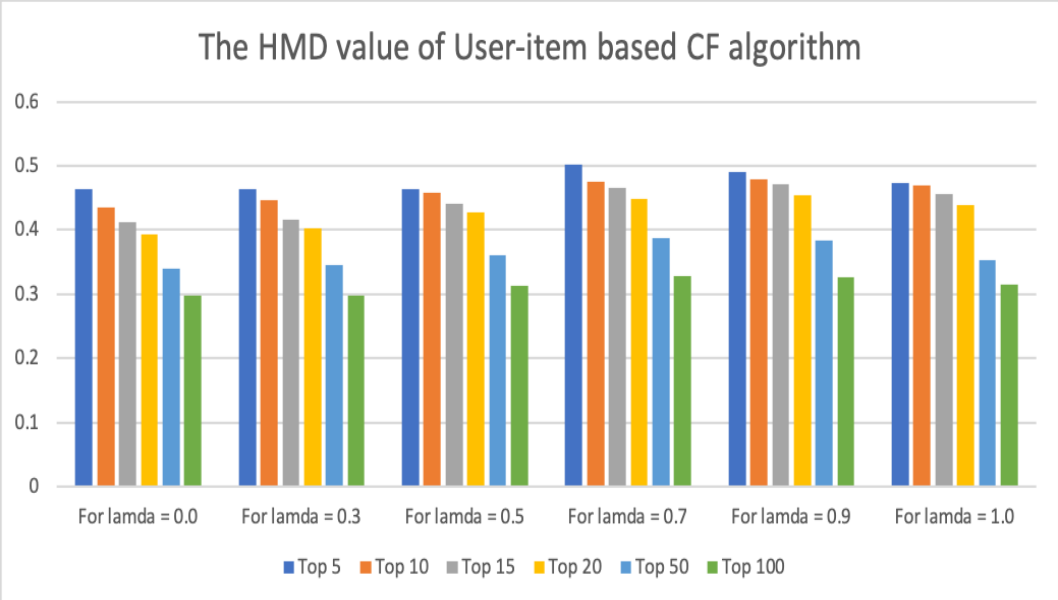

Figure 4.4: The chart of HMD result for User-item based recommendation

According to the HMD evaluation result of User-item based Recommendation model, we can also sum up two rules: 1) the HMD evaluation result of recommendation sults are almost not affected by the user parameters of λ; 2) the diversity of

recommended results decreases gradually, with the number increase of recommended lists. While looking at the overall change curve, we can find that the diversity evaluation of user-item model is only suitable for small corpus recommendation from the perspective of diversity, and the diversity of recommendation results decreases greatly with the increase of the number of recommended list. In the Table 4.7, we demonstrate the DCG result of User-item based Recommendation model. In addition, we also compiled a chart with corresponding to this table in Figure 4.4, in order to visually interpret the DCG results.

4.4.2 Popularity based Recommendation Model

The popularity based Recommendation Model has always been a popular recommendation algorithm. According to the data statistics, we can find that it does have advantages that User-item based recommendation model can not match. The algorithm is simple but the recommendation effect is still good, and the algorithm is completely unaffected by user parameters regardless of DCG evaluation or HMD evaluation. While, the shortcoming is also obvious, which the diversity of the recommend content is poor, and the content recommended in the practical process lacks novelty.

1. The evaluation result of DCG algorithm

   The user parameter of λ has no effect on the result of recommended lists, and the quality of the recommendation results increases linearly with the increase of the number of recommended lists in the DCG evaluation. We can find that the popularity based recommendation model always has better DCG evaluation effect by comparing the User-based recommendation model. In the practical application of recommendation, the higher the ranking with the grater the role of recommendation. In the DCG evaluation, the popularity based recommendation model got the 3.86 point in the top 5 recommended lists, but the User-item based

recommendation model only got the 1.5 points in the top 5 recommended lists. we sorted the DCG evaluation result of the popularity based recommendation model in Table 4.8, and compiled a chart with corresponding to this table in Figure 4.5, in order to visually interpret the DCG results.

Table 4.8: The DCG result of Popularity based Recommendation

	$\lambda = 0$	$\lambda = 0.3$	$\lambda = 0.5$	$\lambda = 0.7$	$\lambda = 0.9$	$\lambda = 1$
Top 5	3.86	3.86	3.86	3.86	3.86	3.86
Top 10	4.31	4.31	4.31	4.31	4.31	4.31
Top 15	4.52	4.52	4.52	4.52	4.52	4.52
Top 20	4.65	4.65	4.65	4.65	4.65	4.65
Top 50	5.03	5.03	5.03	5.03	5.03	5.03
Top 100	5.22	5.22	5.22	5.22	5.22	5.22

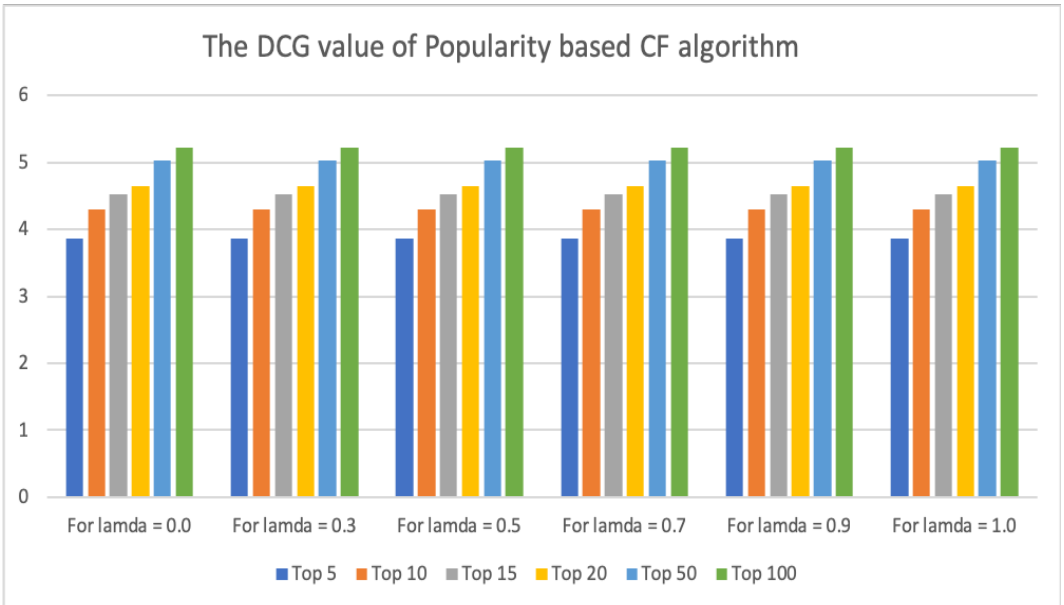

Figure 4.5: The chart of DCG result for Popularity based recommendation

2. The evaluation result of HMD algorithm

Table 4.9: The HMD result of Popularity based Recommendation

	$\lambda = 0$	$\lambda = 0.3$	$\lambda = 0.5$	$\lambda = 0.7$	$\lambda = 0.9$	$\lambda = 1$
Top 5	0.31	0.31	0.31	0.31	0.31	0.31
Top 10	0.31	0.31	0.31	0.31	0.31	0.31
Top 15	0.32	0.32	0.32	0.32	0.32	0.32
Top 20	0.33	0.33	0.33	0.33	0.33	0.33
Top 50	0.36	0.36	0.36	0.36	0.36	0.36
Top 100	0.42	0.42	0.42	0.42	0.42	0.42

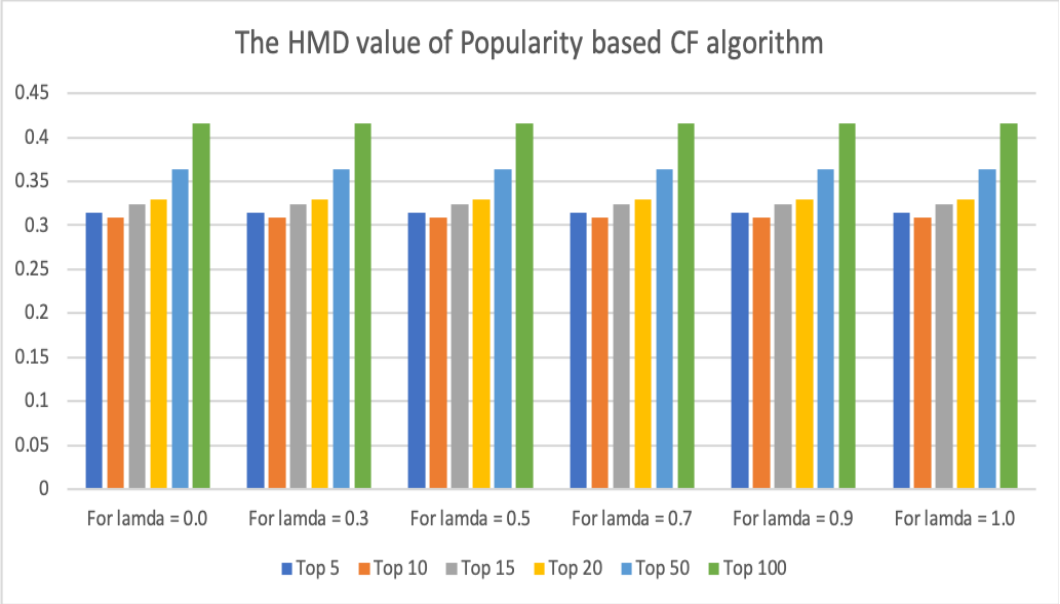

Figure 4.6: The chart of HMD result for Popularity based recommendation

The stability of popularity based recommendation model is an affirmative feature. The evaluation result of DCG evaluation algorithm is stable and higher than that the user-item based recommendation model. Its staility is better in HMD

evaluation, but the evaluation of diversity is not ideal. In the top 5 and top 10 recommended lists, the value of HMD evaluation is only 0.21, while the value of HMD evaluation become better with the increase of the recommended lists. Nevertheless, the higher the top of recommended lists with the greater applied value. Therefore, the effect of popularity based recommendation model is lower than the user-item based recommendation model in terms of overall diversity. we sorted the DCG evaluation result of the popularity based recommendation model in Table 4.9, and compiled a chart with corresponding to this table in Figure 4.6, in order to visually interpret the DCG results.

4.4.3 User-item with Popularity based Recommendation Model

We find that the popularity based recommendation model is better than the user-item based reocmmendation model in the DCG evaluation of recommendation ranking, while the diversity evaluation results are opposite. Therefore, the user-item with popularity based recommendation model aims on combine the advantages of the two recommendation models to establish a relatively comprehensive model of ranking accuracy and diversity coexisting. Moreover, the implementation principles of the two models are not conflict, which provides a good practical basis for the user-item with popularity based recommendation model. The experimental results show that the quality and diversity of rankings have been improved in varying degrees by combining the two recommendation models.

1. The evaluation result of DCG algorithm

   Compared with the first two recommendation models, the DCG evaluation result have been greatly improved in different number of recommended lists. In the top 5 recommended lists, the highest DCG evaluation results reached 4.12, while the

user-item based recommendation model only got 1.46 and the popularity based recommendation model got 3.86. With the increase of recommended lists, the DCG evaluation results also continued to grow. In the top 100 recommended lists, the highest DCG evaluation results reach 7.43, which the evaluation results are much better than the first two recommendation model. However, the DCG evaluation results are influenced by user parameters of λ, and the lowest DCG evaluation results gradually shift the centre value of λ with the increase of the number of recommended lists, which same as the user-item based recommendation model.

Therefore, the recommended results of user-item with popularity based recommendation model is better than the first two recommendation model. The model is a good collaborative filtering based recommendation model if does not consider the diversity factor. Moreover, the method of combining popularity with item similarity is a good research orientation in the field of collaborative filtering based recommendation model. We sorted the DCG evaluation result of the popularity based recommendation model in Table 4.10, and compiled a chart with corresponding to this table in Figure 4.7, in order to visually interpret the DCG results.

2. The evaluation result of HMD algorithm

Although the user-item with popularity based recommendation model has far more effect on the DCG ranking quality assessment than the first two models, while the performance of HMD diversity assessment is not satisfactory. In the top 5 and top 10 recommended lists, the HMD diversity assessment of this recommendation model is slightly higher than the popularity based recommendation model, but the HMD diversity assessment is evaluated as the worst model of these three models in the top 15 and more recommended list. However, the assesment effect of recommendation diversity increases with the user parameters of λ

Table 4.10: The DCG result of User-item with Popularity based Recommendation

	$\lambda = 0$	$\lambda = 0.3$	$\lambda = 0.5$	$\lambda = 0.7$	$\lambda = 0.9$	$\lambda = 1$
Top 5	4.12	4.04	3.99	3.91	3.88	3.88
Top 10	4.75	4.63	4.57	4.45	4.38	4.39
Top 15	5.13	4.99	4.90	4.76	4.69	4.70
Top 20	5.42	5.26	5.16	5.00	4.93	4.95
Top 50	6.49	6.31	6.20	6.02	5.97	6.05
Top 100	7.43	7.35	7.32	7.19	7.23	7.36

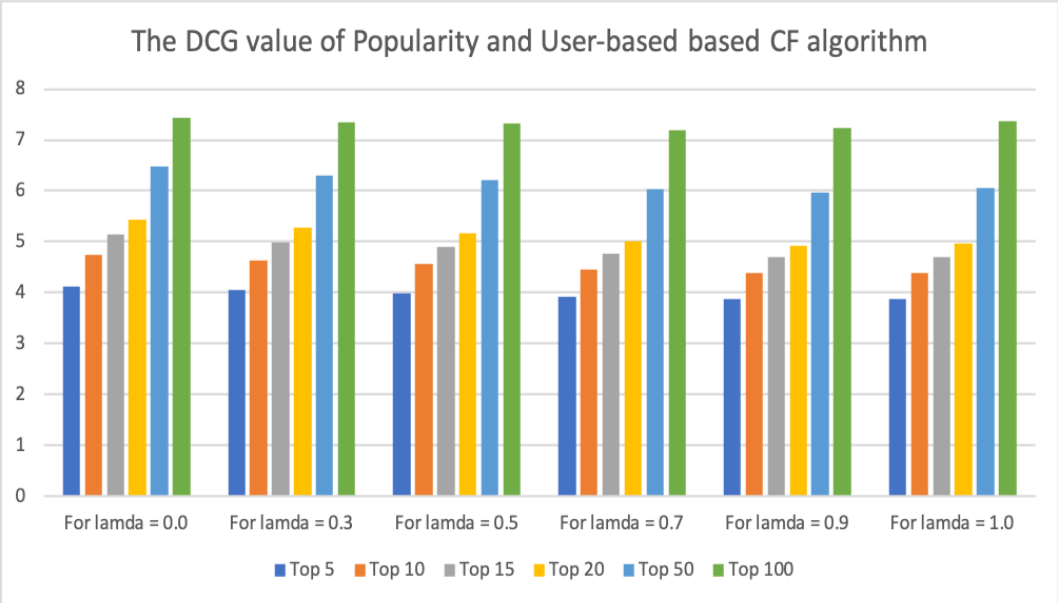

Figure 4.7: The chart of DCG result for User-item with Popularity based recommendation

increase in the HMD evaluation results. Moreover, the higher ranking of recommended lists have the higher practical value in the personalized recommendation. Therefore, the overall evaluation of this recommendation model is superior to the popularity based recommendation model, regardless of the ranking quality

of DCG evaluation or the diversity of HMD evaluation. We have compiled the overall HMD evaluation results of user-item with popularity based recommendation model in Table 4.11, and organize a chart to correspond to the contents of the table in Figure 4.8, in order to visualize the trend of HMD evaluation result.

Table 4.11: The HMD result of User-item with Popularity based Recommendatio

	$\lambda = 0$	$\lambda = 0.3$	$\lambda = 0.5$	$\lambda = 0.7$	$\lambda = 0.9$	$\lambda = 1$
Top 5	0.33	0.32	0.31	0.30	0.33	0.34
Top 10	0.32	0.31	0.31	0.32	0.34	0.36
Top 15	0.31	0.31	0.32	0.34	0.37	0.39
Top 20	0.29	0.30	0.33	0.34	0.38	0.40
Top 50	0.24	0.25	0.28	0.31	0.36	0.37
Top 100	0.18	0.21	0.24	0.28	0.30	0.31

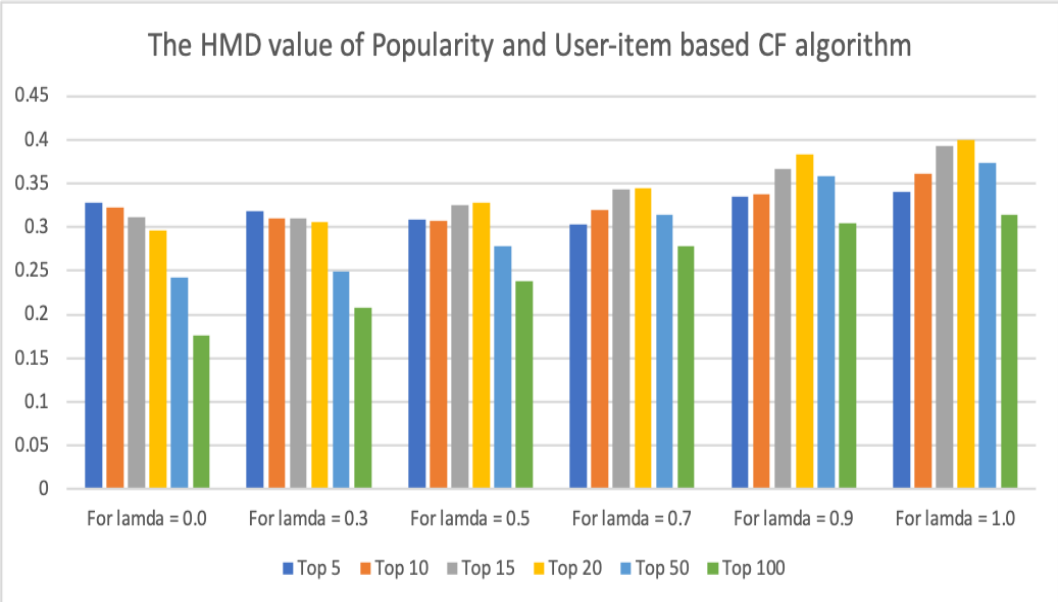

Figure 4.8: The chart of HMD result for User-item with Popularity based recommendation

4.4.4 PMF based Recommendation Model

Nowadays, Matrix Factorization is one of the mainstream algorithms in recommender system. The Probabilistic Matrix Factorization (PMF) is a probabilistic version of Matrix Factorization, which implementation principle is very similar to the topic model of LDA algorithm. Therefore, we choose the PMF recommendation model to implement the Web APIs recommendation, and compare with the first three recommendation model based on the HMD and DCG evaluation algorithms. The experimental results show that the overall evaluation result of PMF based recommendation model is much higher than the first three recommendation models, regardless of the ranking quality or diversity of the recommended lists. Moreover, the Web APIs recommendation results of PMF recommendation model is very little affected by user parameters of λ, compared with the user-item based recommendation model. However, the fluctuation of the prediction score is large in prediction matrix of R, since the PMF recommendation model is randomly split when splitting the training data set and testing data set.

1. The evaluation result of DCG algorithm

   The DCG assessment showed that the quality of the recommended lists increased linearly as the increase of recommended lists, and its almost unaffected by user paramenters of λ. When the recommended list is the top 5, the DCG evaluation results is 9.79, which is far superior to the best results of the first three recommendation modes. The DCG evaluation results have also increased significantly, with the increase of the recommended lists. When the recommended list is the top 100, the DCG evaluation result has reached 69.57. Moreover, the DCG quality assessment of the recommendation results is almost unaffected by user parameters, after compared the DCG evaluation result with the λ of 0, 0.3, 0.5, 0.7, 0.9, and 1. We sorted the DCG evaluation result of the PMF based recommendation model in Table 4.12, and compiled a chart with corresponding

to this table in Figure , in order to visually interpret the DCG results.

Table 4.12: The DCG result of PMF based Recommendation

	$\lambda = 0$	$\lambda = 0.3$	$\lambda = 0.5$	$\lambda = 0.7$	$\lambda = 0.9$	$\lambda = 1$
Top 5	9.80	9.80	9.80	9.80	9.80	9.80
Top 10	15.09	15.10	15.10	15.10	15.10	15.10
Top 15	19.48	19.48	19.48	19.47	19.47	19.48
Top 20	23.39	23.39	23.39	23.39	23.39	23.40
Top 50	42.86	42.86	42.86	42.86	42.85	42.85
Top 100	69.57	69.57	69.57	69.57	69.57	69.57

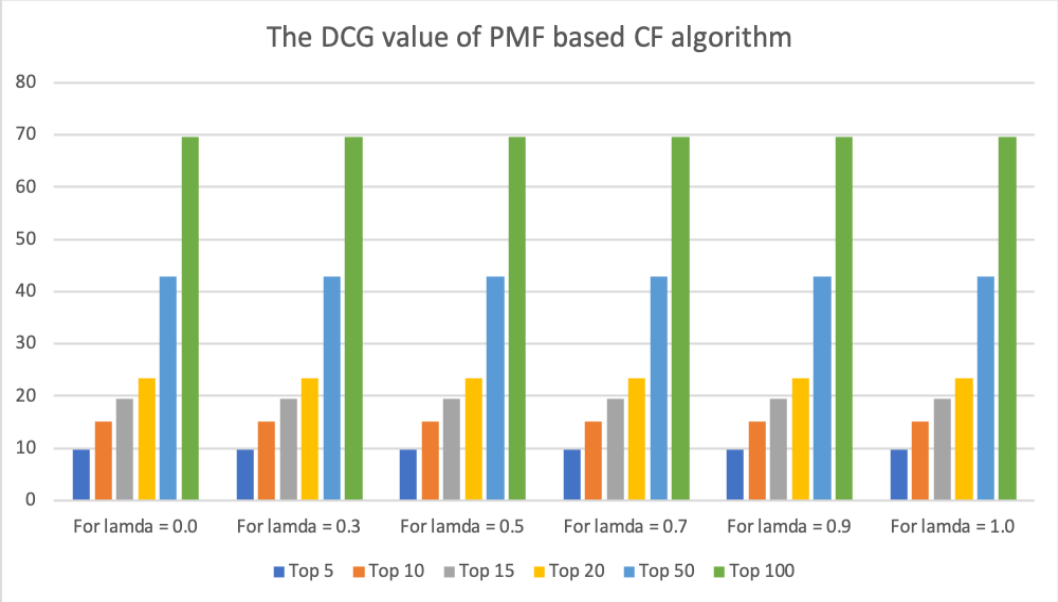

Figure 4.9: The chart of DCG result for PMF based recommendation

2. The evaluation result of HMD algorithm

   The HMD evaluation results of PMF based recommendation model has two rules: 1) the diversity decreases with the increase of the recommended lists; 2) the

Table 4.13: The HMD result of PMF based Recommendation

	$\lambda = 0$	$\lambda = 0.3$	$\lambda = 0.5$	$\lambda = 0.7$	$\lambda = 0.9$	$\lambda = 1$
Top 5	0.59	0.56	0.59	0.49	0.49	0.59
Top 10	0.51	0.45	0.45	0.42	0.40	0.53
Top 15	0.47	0.39	0.38	0.41	0.36	0.45
Top 20	0.42	0.34	0.34	0.37	0.33	0.40
Top 50	0.23	0.20	0.19	0.21	0.21	0.22
Top 100	0.29	0.30	0.27	0.24	0.25	0.25

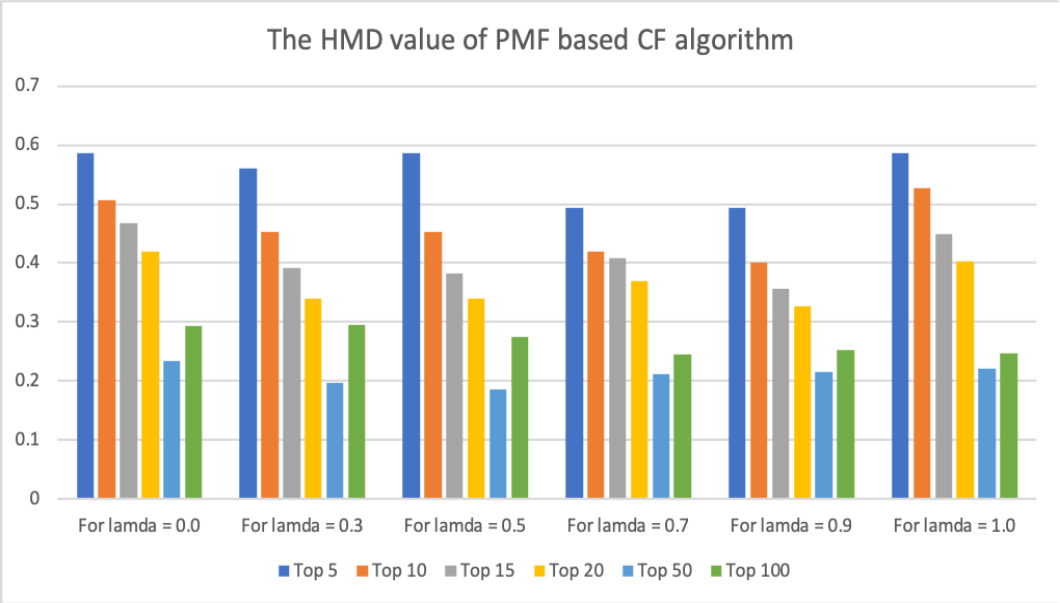

Figure 4.10: The chart of HMD result for PMF based recommendation

diversity of the recommended lists are slightly affected by the user parameters of λ, and the diversity is the lowest when the λ is 0.7. The HMD evaluation value of PMF based recommendation model be able to reach 0.59, which is the best HMD evaluation value among the four recommendation models. The diversity of the PMF based recommendation model has the best diversity assessment results

from the top 5 to the top 20 recommended lists. Based on the principle that the higher the recommended lists with the significance of the recommendation content, thus, the diversity of PMF recommendation model is the best among the four recommendation models.

When the λ is 0.7, the diversity of recommended lists are the worst: top 5 HMD evaluation value is 0.49; top 10 HMD evaluation value is 0.42; the top 15 HMD evaluation value is 0.41; the top 20 HMD evalutaion value is 0.37. Which the worst diversity results of PMF based recommendation model still has the best diversity effect, compare the first three recommendation models. We have compiled the overall HMD evaluation results of PMF based recommendation model in Table 4.13, and organize a chart to correspond to the contents of the table in Figure 5.2, in order to visualize the trend of HMD evaluation result.

4.5 Conclusion

In this chapter, we first adopt LDA and TF-IDF algorithm for topic vector extraction, and user JS divergence to measure the similarty between Mashup services, then apply K-means and Agnes algorithm to generate initial user data set, and finally user and compare a variety of collaborative filtering recommendation model to recommend Web APIs. The experimental results show that the LDA based clustering model has excellent classification effect, and the PMF based recommendation model significantly higher than other recommendation models. In addition, both of the LDA and the PMF are derived algorithms based on SVD algorithm. In the next chapter, we will disccuss and compare the experimental process in detail.

Chapter 5

Discussion

5.1 Introduction

In the previous chapter, the experimental results have been presented, but there is no excessive analysis and comparisons. In this chapter, we will focus on analyzing the results of experiments and comparing the effects and accuracy of different models. Finally, we will explain the reason for using exists models and contributions we made in this thesis.

In the Recommender system, two core processes are topic vector extraction and Web APIs recommendation. Thus, we discuss the LDA model and TF-IDF model for topic vector extraction, and then compare the four collaborative filtering recommendation algorithm: User-item based recommendation model, Popularity based recommendation model, User-item with popularity based recommendation model, and PMF based recommendation model. Finally, we illustrate the contribution of this reserach comined with the whole experimental process.

5.2 Topic Vector Extraction

Nowadays, the mainstream models of topic vector extraction can be divided into three categories: Bag-of-Words, feature statistics, and N-gram. The representative application of the Bag-of-Words technology is the topic model, which is also one of the most mature methods of topic vector extraction in the field of data mining. Feature statistics are based on the statistical features of vocabulary as a feature set, which representative technique is TF-IDF. The N-gram is a new topic vector extraction model introduced by Google in recent years, which is based on neural network model. Although the training result of neural network model are worthy of recognition, while the process of data training lacks interpret-ability. In addition, the N-gram model requires a large amount of training data, and our data sets are difficult to support it. Therefore, we adopted the LDA model and TF-IDF as the topic vector extraction method in the experimental process. The fundamental purpose of topic vector extraction is to effectively classify the data set to solve the cold-start problem of the recommender system.

Assessing how well a topic model is modeled depends on the quality of the keywords extraction. The keywords in each topic should have 4 features: 1) the frequency of occurrence of keywords is large enough in training data set; 2) the extracted keywords are sufficient to distinguish different topics; 3) the frequency of keywords co-occurrence in Mashup service documents with the same topic should be similar; 4) The semantics of the keywords in each topic should be very close, such as "apple" and "oranges" in the fruit topic, or "love" and "hate" in the emotional topic.

In general, the TF-IDF vectorization method is a good choice if there are many words in the data set that appear frequently in multiple documents. These frequently occurring words are treated as noise data, which affects the fitting effect of the model. And the actual effect of TF-IDF is not ideal in distinguishing between topic and keyword semantics. The LDA model introduces prior distribution and Gibbs sampling method,

which greatly alleviates the over-fitting problem and works well in distinguishing between topic and semantic extraction. The method of introdcing complex network into the LDA model can greatly solve the problem of insufficient training of LDA models in short documents. And the training result of processing short documents within one thousand documents based on the network structure LDA model is similar to the result of manual classification.

Therefore, the LDA model based Mashup service document clustering in the personalized recommendation is an effective method for processing raw data set, which be able to greatly reduce the workload of manual classification. However, experiments have shown that after the raw data set is large enough, it is different to deal with the over-fitting problem whether it is the TF-IDF model or the LDA model. Aiming the over-fitting problem, this paper adopts the method the data set after the LDA model training, and performing multiple clustering by K-means algorithm and Agnes algorithm, which has achieved remarkable results.

The data results show that TF-IDF does not consider the semantics of the data sets, and only performs vector extraction based on the frequency of words. Although the TF-IDF is simple and straightforward, but it lacks the accuracy of vector extraction and difficult to implement vector cluster training. Therefore, the TF-IDF model is not suitable for solving the training data sets cold-start problem of recommender system.

The LDA model be able to get the topic distribution of each Mashup service document: $p(topic|doc)$, then we can calculate the distance between Mashup service documents by JS algorithm, according to the topic vectors: $JS(MS_i, MS_j)$. Finally, the K-means and Agnes clustering algorithm are used to realize the clustering of Mashup service documents, and generate the Mashup clusters as user in recommender system to sovle the cold-start problem.

5.3 Recommendation Model Comparison

We selected the best result for each recommendation model in the same number of recommended list to compare the best performance of the four recommendation models in this section. We conduct further analysis based on the experimental results in the Result chapter to more intuitively compare the recommended effects of different recommendation models. However, the PMF based recommendation model is superior to other models, whether it is the ranking quality of recommended lists or the diversity of recommended content. This reflects the root cause of the Matrix Factorization methods becoming the mainstream algorithm in the field of recommendation algorithms in recent years. Next, we will analyze and compare the HMD assessment results and DCG assessment results.

5.3.1 DCG Assessment

In the DCG assessment results, we selected the best assessment for the same number of recommended lists in each recommendation model. Since the DCG assessment results of the first three recommendation models are not much different, we adopted a stacked line curve chart to achieve a more intuitive models comparison. We can find that the user-item based recommendation model has the worst DCG evaluation results, and the popularity based recommendation model is slightly better than the user-item recommendation. The DCG evaluation result of the item-user with popularity based recommendation results are better than those of the former two recommendation models. The DCG evaluation results of the PMF based recommendation are much better than the first three recommendation models. The best DCG assessment results of each recommendation model are shown in Table 5.1, and the stacked line curve chart of the comparison results are demonstrated in Figure 5.1, which are intuitively exhibition the difference among the four different recommendation models.

Table 5.1: The DCG comparison results

	User-item	**Popularity**	**User-item with popularity**	**PMF**
Top 5	1.46	3.86	4.12	9.80
Top 10	1.87	4.31	4.75	15.09
Top 15	2.17	4.52	5.13	19.48
Top 20	2.40	4.65	5.42	23.39
Top 50	3.28	5.03	6.49	42.86
Top 100	4.09	5.22	7.43	69.57

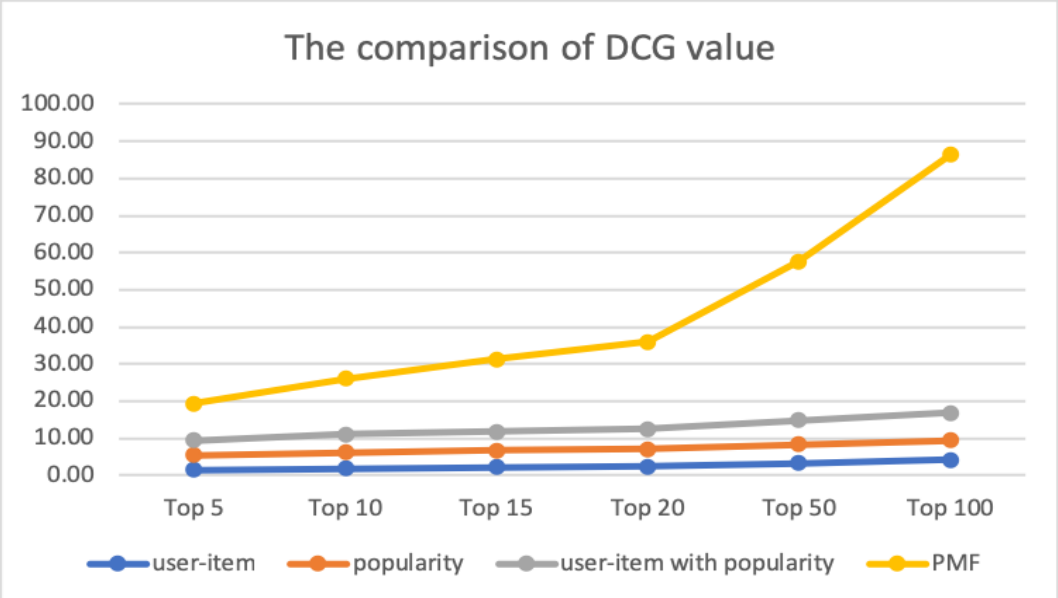

Figure 5.1: The stacked line curve chart of the DCG comparison results

The user-item based recommendation model and the popularity based are the most original theoretical idea of collaborative filtering. The basid idea of User-item based recommendation model is to calculate the similarity between items based on historical

preference data of all users, and then recommend the similar items of user's preference items to the user. The recommendation algorithm model was first introduced by Amazon in 1998, which is able to offer recommendations to millions of users based on millions of items.

The popularity based recommendation algorithm is simpler and more rude, mainly for hot items or information recommendations. The User-item based recommendation model and the popularity based popularity recommendation model are able to be called the most primitive collaborative filtering recommendation algorithm. The implementation of the first two recommendation models are not conflict with each other, so they be able to achieve the superposition effect of the algorithm. From the Figure 5.1, we are able to clearly find that the DCG evaluation of the popularity based recommendation model is better than the User-item based recommendation model, and the combination of the first two recommendation algorithm model be able to obtain better DCG evaluation results of recommended ranking.

The PMF is essentially a recommendation algorithm model that combines the user-item based collaborative filtering recommendation model and the latent feature mining of topic model. The recommendation model mainly explores the latent features of Web APIs and Mashup clusters, and give Web APIs and Mashup clusters a specified number of low-dimension features. Therefore, we were concerned about the uncertainty of the quality of recommended ranking in the PMF based recommendation model. However, we are surprised to find that the recommendation model is not only very stable in the recommendation content of recommended lists, but also far higher than the first three traditional collaborative filtering algorithm in DCG ranking quality evaluation after repeatedly experiment.

The DCG ranking quality of these four recommendation algorithm models increased with the increase of the recommended lists. In the PMF based recommendation model, the DCG ranking quality score of top five recommended lists has reached 9.8, which is

more than double the DCG evaluation of user-item with popularity based recommendation model. The DCG ranking quality assessment of PMF based recommendation model reached up to 69.57 in the top 100 recommended lists. Therefore, the PMF based recommendation model is undoubtedly the best recommendation of these four recommendation in terms of DCG quality assessment.

5.3.2 HMD Assessment

In the HMD diversity assessment results, we also selected the best assessment for the same number of recommended lists in each recommendation model. Since the HMD evaluation assessment value range is $[0, 1]$, and the higher the assessment score with the higher the diversity of the recommended lists. Thus, the difference between the HMD assessment results of the four recommendation model will not particularly large, and we selected the line curve chart to achieve the purpose of comparing the HMD evaluation results of the four recommendation. We can find that the PMF based recommendation model has the best recommendation content diversity in the top 5 and top 10 recommended lists; the PMF based recommendation model and the User-item with popularity based recommendation model have the same recommendation diversity evaluation results in the top 15 recommended lists; the User-item with popularity based recommendation model has the best recommendation content diversity in the top 20 and top 50 recommended lists; the popularity based recommendation model has the best recommendation content diversity in the top 100 recommended lists. The best HMD assessment results of each recommendation model are shown in Table 5.2, and the line curve chart of the comparison results are demonstrated in Figure 5.2, which are intuitively exhibition the difference among the four different recommendation models.

The user-item based collaborative filtering algorithm only considers the similarity of Web APIs content to achieve the purpose of Web APIs recommendation. Therefore,

Table 5.2: The HMD comparison results

	User-item	**Popularity**	**User-item with popularity**	**PMF**
Top 5	0.50	0.31	0.33	0.59
Top 10	0.48	0.31	0.32	0.53
Top 15	0.47	0.32	0.31	0.47
Top 20	0.45	0.33	0.30	0.42
Top 50	0.39	0.36	0.24	0.23
Top 100	0.33	0.42	0.18	0.29

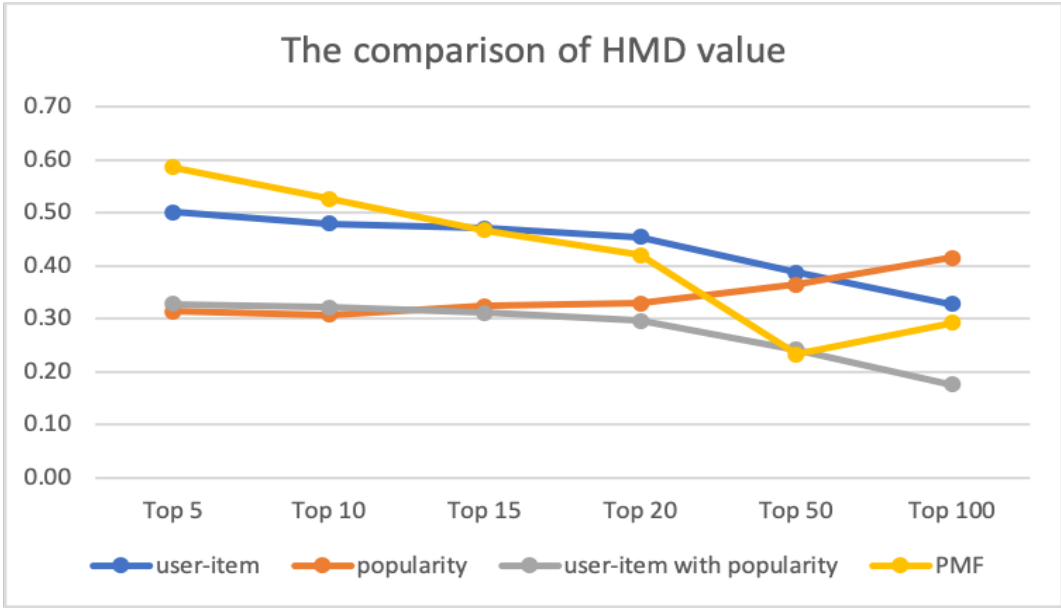

Figure 5.2: The chart of HMD result for PMF based recommendation

the diversity of recommended content of the algorithm model is the worst in the four recommendation models. Although similar items that users prefer can be recommended, but the recommended content lacking in originality. This most primitive collaborative

filtering recommendation algorithm has been gradually eliminated with the development of the recommender system. However, the most of collaborative filtering recommendation algorithms are based on the recommendation algorithm model. Therefore, its importance in the recommender system is undoubted, although the recommended effect of this recommendation model is not ideal compared with other recommendation models.

The popularity based collaborative filtering algorithm only considers the invoking relationship between Web APIs and Mashup service clusters, which the implementation principle is simpler and more straightforward than the user-item based recommendation model. However, the diversity effect of the algorithm increases as the recommended lists increases. The diversity of the algorithm is best in these 4 recommendation models, when the recommended lists reaches the top 100. But the algorithm also has the worst diversity effect in top 5 and top 10 recommended lists. Since the higher the recommended list with the higher the reference value, so the algorithm also face the problem of lacking in originality. But this algorithm still has good effects in some specific areas, such as movie and music recommendation.

The user-item with popularity based recommendation model combine the first two recommendation model, which the diversity effect is much better than the first two algorithms. The algorithm is more stable diversity of different recommended lists, and it has the best diversity effect in the top 15, top 20, and top 50 recommended lists. We can find that the collaborative filtering algorithm combining the similarity and the invoking relationship be able to achieve improvement effect in algorithm evaluation results. The algorithm is the commonly used collaborative filtering recommendation method in the personalized recommender system.

The PMF based recommendation model introduces the idea of latent feature mining based on the collaborative filtering of similarity and invoking relationships. The model

also has the best diversity of recommendation content in top 5, top 10, and top 15 recommended lists, which demonstrate the reliability of the model in terms of the diversity of recommendations. However, we can find that the diversity of the recommended content of the model fluctuates greatly, and the diversity of recommendation content is the worst in top 50 recommended lists. The diversity evaluation of the algorithm gradually begins to rise again, when the recommended list exceeds the top 50. Therefore, the PMF recommendation model is worth affirming not only in ranking quality evaluation but also in diversity evaluation.

5.4 Contributions

In this thesis, we first crawled 17,799 Web APIs and 6,342 Mashup service document in the largest Web APIs repository of ProgrammableWeb, and adopted the methods of Remove Stop Words, Extract Stemming, and Remove duplicate words to pre-process the training data set. Then the topic vector extraction is performed by using the algorithms of TF-IDF and LDA in topic model, and the K-means and Anges clustering algorithms are implemented based on the pre-process of training data set, which generated 20 Mashup clusters, and each cluster is used as a user of the Recommender system. Next, we applied the four recommendation model of User-item based recommendation model, Popularity based recommendation model, User-item with popularity based recommendation model, and PMF based recommendation model for Web APIs recommendations. Finally, the DCG algorithm is used to evaluate the ranking quality of the recommended lists of recommendation model, and the HMD algorithm is adopted to evaluate the diversity of the ranking content. The advantages and disadvantages of the four recommendation model, and the two topic vector extraction model are compared by the experimental results.

After decades of development in data mining, TF-IDF model and LDA model

are widely used in data retrieval and recommendation system as two classical vector extraction algorithms. This thesis uses the two models to extract the topic vector and generate clustering results by K-means and Agnes algorithms to solve the cold-start problem in the collaborative filtering recommendation algorithm. The experimental results show that the semantic mining effect of TF-IDF on Mashup service documents are not ideal, so it is difficult to generate a good Mashup service document clustering results; the LDA model combined with complex network model has excellent clustering results for a certain number of Mashup service documents, which close to manual classification. Therefore, we conclude that LDA combines complex network is used to extract topic vectors, and K-means combined with Agnes algorithm be able to effectively solve the cold-start problem of collaborative filtering recommendation algorithm.

In the recommendation algorithm section, we adopted four collaborative filtering recommendation algorithms to achieve the purpose of model comparison, which are User-item based recommendation model, popularity based recommendation model, User-item based recommendation model, and PMF based recommendation model. The DCG algorithm and the HMD algorithm are used to achieve the purpose of comparing the ranking quality of recommended and the diversity of recommended content. The user-item based recommendation model and the popularity based recommendation model are widely used on different products in the network, including Youtube, Netflix. The success of theses two algorithm is that they are simple, scalable, can interpret the recommendation content in an easy-to-understand way. We tired to combine the first two collaborative filtering recommendation algorithms and get better ranking quality and diversity.

Moreover, we also applied the PMF based recommendation algorithm to compare with first three recommendation models, which adds latent semantic exploration attributes while considering the similarity and popularity of Web APIs. Finally, we find that

the PMF recommendation algorithm model is far superior to the first three recommendation models in terms of diversity and ranking quality. Therefore, the more factors considered by the algorithm, the more comprehensive the recommendation result, if without considering the over-fitting.

In this research, the probabilistic theory represented by the topic model is used to implemnt the complete recommender system model. In the Literature Review chapter, we conclude that the Latent Factor Model (LFM) based Matrix Factorization model is a variant model of the topic model in the field of collaborative filtering algorithms, thus, Matrix Factorization be able to be regarded as a kind of topic model. The two core algorithms in this thesis are LDA model and PMF model respectively. The LDA model is a recognized representative algorithm of topic model, and the PMF model is a probabilistic version of the Matrix Factorization which the implementation principle can intuitively demonstrate the similarity between Matrix Factorization and topic model.

Finally, we extract the topic vectors from the crawled data of ProgrammableWeb by two topic model algorithms, and Web APIs recommendation by manifold collaborative filtering algorithms. Then, several widely used topic vectors extraction and collaborative filtering algorithms are compared. We conclude that the topic vector extraction of the LDA model is significantly better than the TF-IDF model, and the PMF model is significantly better than the collaborative filtering recommendation algorithms based on item similarity and item popularity. Moreover, the research project provides a practial and guiding solution for the personalized recommender system of related websites.

Chapter 6

Conclusions

In this thesis, we propose and build a complete Recommender system based on the crawled training data set from the public web pages of the website $ProgrammableWeb$. The interpret-ability of the Recommender system is the core idea of this research, thus, this paper adopts the Recommender system related algorithm based on content and probability theory. A variety of classical topic vector extraction and collaborative filtering recommendation algorithms are implemented and compared. Finally, we find that the LDA model be able to obtain good topic vector extraction results for Mashup service documents clustering, and the PMF based collaborative filtering can obtain excellent recommendation results under the influence of user parameters. However, there are still some limitations in the Recommender system which are identified as follows. Meanwhile, the further research is presented accordingly.

6.1 Limitations

In this research, several classical algorithm models in the field of Recommender system are analyzed and compared. However, these algorithm models still has some limitations, with the rapid development of science and the explosive growth of Internet information.

The limitation of this Recommender system can be divided into five parts: 1) the over-fitting problem of the LDA model, which it is difficult to obtain the ideal topic model extraction effect when the amount of documents are too many or too few; 2) Small numbers of Mashup services lead to poor service clustering; 3) the probabilistic based model has strong interpretative, while the scalability is weak; 4) the user's interest migration problem; 5) the recommendation data sparsity problem in the context of big data.

1. Over-fitting problem of the LDA model

   The LDA model introduces the Bayesian framework and prior probability distribution for the PLSA model, which greatly alleviates the over-fitting problem of the traditional topic models. However, we found that the LDA model still can not completely solve the over-fitting problem, even if combine with the complex network in Gibbs sampling. When the number of Mashup service document is 354, the LDA model has the best effect on the top vector extraction of Mashup service documents; when the number of Mashup service documents reach 750, the over-fitting problem begins to be highlighted; when the number of Mashup service documents exceed 1000, the over-fitting problem has a serious impact on clustering results. Therefore, we performed fractional topic vector extraction for 6342 Mashup documents during the experiment.

2. Small numbers of Mashup services lead to poor service clustering

   The training data set of Mashup service documents are crawled from the public web page of the website $ProgrammableWeb$, which top 20 categories involve 3999 Mashup services, accounting for 63 percent of the total Mashup services in the data set; and 273 categories involve less than 20 Mashup service documents. However, it is difficult to cluster the minority of Mashup service categories, whether it is the K-means clustering algorithm or the Agnes clustering algorithm.

Thus, we separate the categories with a small number of Mashup service documents to improve the accuracy of clustering results. However, this approach can not insure the clustering effect of the small number of categories.

3. The scalability problem

   With the rapid development of artificial intelligence in recent years, deep learning algorithm and reinforcement learning algorithm have become popular research in the field of machine learning. Most machine learning algorithms have been tried to combine with them. However, deep learning and reinforcement learning are based on the neural network model, ultimately achieve the desired training results after multi-level training. While, the model based probability is difficult to carry out effective multi-level training. Thus, although the interpret-ability of our model is guaranteed, but the scalability problem in artificial intelligence is still difficult to be effectively solved.

4. The user's interest migration problem

   Interest migration is also an important factor affecting the rating of the Recommender system, and the user's preference for the item will decay over time. Since the training data set is offline data, and the interest attenuation mechanism has no effect on this experiment, so this thesis does not consider the interest migration factor.

5. The recommendation data Sparsity problem

   The data sparsity problem is the main problem faced by the Recommender system, and it is an important reason for declining the quality of Recommender system. In this thesis, dimensional reduction and clustering methods are used to effectively alleviate this problem, but when the size of data sets are too large, the sparsity problem is still difficult to guarantee.

6.2 Further Research

In this thesis, we compared the classical and representative algorithmic models of Recommender system with the core idea of topic model. This research has provided us with a deep understanding of the algorithm model of Recommender system, but the model still needs further optimization and upgrading to cope with the explosive development of Internet information.

In the future work, we will combine our experimental model with the frontier algorithm models of Recommender system in order to improve the quality of recommendation ranking and the diversity of recommendation, in view of the limitation of existing Recommender system. We will introduce Word2Vec topic vector extraction model to combine the idea of topic model, and Tensor Factorization to to combine the Matrix Factorization model, and finally achieve the goal of perfectly combine the experimental model with deep learning and reinforcement learning algorithm to improve the accuracy of the algorithm by introducing artificial intelligence.

We will provide some imaginary solutions to the limitations of the Recommender system in the following. We will focus on the over-fitting problem, the clustering problem of minority items, scalability problem of probabilistic model, user interest migration problem, and data sparsity problem. Finally, we would like to build an advanced Recommender system to promote the development of the field of data mining.

1. Over-fitting problem

   Shi et al. proposed the WE-LDA model in 2017, which combines Word2Vec with the LDA model (M. Shi, Liu, Zhou, Tang & Cao, 2017). We are also going to follow the method to combine Word2Vec with LDA model, in order to solve the over-fitting problem of the LDA model topic model extraction.

2. The clustering problem of minority items

The most of deep learning and reinforcement learning algorithms are based on supervised learning and semi-supervised learning, which may effectively solve the clustering problem of minority items. Therefore, we plan to adopt semi-supervised training and supervised training to compare unsupervised clustering, in order to find the best solution for the clustering problem of minority items.

3. Scalability problem of probabilistic model

   Kim et al. combine Matrix Factorization with convolutional neural networks and achieve excellent predict results (D. Kim et al., 2016). Therefore, we find that the Matrix Factorization be able to perfectly fit the deep learning algorithm, and the PMF model is the probabilistic interpretation version of the Matrix Factorization. Thus, we believe that the probabilistic model be able to convert to the neural network model. Therefore, combining the advantages of probability and neurology is also the main direction of our future research.

4. User interest migration problem

   Aiming at the problem of user interest migration problem, we plan to introduce a attenuation mechanism, that is, to keep the degree of user's preference for each keyword in the corpus attenuated periodically. In addition, we are going to set a threshold L to clear keywords whose preference value is less than L after each user's attenuation update is completed.

5. Data sparsity problem

   Deep learning and reinforcement learning have become the mainstream algorithms in the field of data mining, since they have good effect in dealing with the data sparsity problem (Gharia et al., 2018). Therefore, it is an effective method to solve the problem of data sparsity by introducing our experimental model into convolutional neural network model with multi-layer training.

References

Abdelkhalek, R., Boukhris, I. & Elouedi, Z. (2016). Evidential item-based collaborative filtering. In *International conference on knowledge science, engineering and management* (pp. 628–639).

Adelfio, G., Chiodi, M., D'Alessandro, A., Luzio, D., D'Anna, G. & Mangano, G. (2012). Simultaneous seismic wave clustering and registration. *Computers & geosciences*, *44*, 60–69.

Adomavicius, G. & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge & Data Engineering*(6), 734–749.

Agarwal, D. & Chen, B.-C. (2009). Regression-based latent factor models. In *Proceedings of the 15th acm sigkdd international conference on knowledge discovery and data mining* (pp. 19–28).

Alonso, G., Casati, F., Kuno, H. & Machiraju, V. (2004). Web services. In *Web services* (pp. 123–149). Springer.

AlZu'bi, S., Hawashin, B., EIBes, M. & Al-Ayyoub, M. (2018). A novel recommender system based on apriori algorithm for requirements engineering. In *2018 fifth international conference on social networks analysis, management and security (snams)* (pp. 323–327).

Ayadi, R., Maraoui, M. & Zrigui, M. (2015). Lda and lsi as a dimensionality reduction method in arabic document classification. In *International conference on information and software technologies* (pp. 491–502).

Badaro, G., Hajj, H., El-Hajj, W. & Nachman, L. (2013). A hybrid approach with collaborative filtering for recommender systems. In *2013 9th international wireless communications and mobile computing conference (iwcmc)* (pp. 349–354).

Bagde, U. & Tripathi, P. (2018). An analytic survey on mapreduce based k-means and its hybrid clustering algorithms. In *2018 second international conference on computing methodologies and communication (iccmc)* (pp. 32–36).

Bayarri, M. J. & Berger, J. O. (2004). The interplay of bayesian and frequentist analysis. *Statistical Science*, 58–80.

Bell, R. M. & Koren, Y. (2007). Lessons from the netflix prize challenge. *SiGKDD Explorations*, *9*(2), 75–79.

Bergamaschi, S. & Po, L. (2014). Comparing lda and lsa topic models for content-based movie recommendation systems. In *International conference on web information*

systems and technologies (pp. 247–263).

Bíró, I. (2009). Document classification with latent dirichlet allocation. *Unpublished Doctoral Dissertation, Eotvos Lorand University*, *4*.

Blanco-Fernández, Y., López-Nores, M., Gil-Solla, A., Ramos-Cabrer, M. & Pazos-Arias, J. J. (2011). Exploring synergies between content-based filtering and spreading activation techniques in knowledge-based recommender systems. *Information Sciences*, *181*(21), 4823–4846.

Blei, D. M. & Lafferty, J. D. (2009). Topic models. In *Text mining* (pp. 101–124). Chapman and Hall/CRC.

Blei, D. M., Ng, A. Y. & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, *3*(Jan), 993–1022.

Campos, M. M. & Milenova, B. L. (2007, February 6). *Orthogonal partitioning clustering.* Google Patents. (US Patent 7,174,344)

Cao, B., Liu, X., Rahman, M. M., Li, B., Liu, J. & Tang, M. (2017). Integrated content and network-based service clustering and web apis recommendation for mashup development. *IEEE Transactions on Services Computing*.

Chang, T.-M. & Hsiao, W.-F. (2013). Lda-based personalized document recommendation. In *Pacis* (p. 13).

Crossno, P. J., Wilson, A. T., Shead, T. M. & Dunlavy, D. M. (2011). Topicview: Visually comparing topic models of text collections. In *2011 ieee 23rd international conference on tools with artificial intelligence* (pp. 936–943).

Darling, W. M. (2011). A theoretical and practical implementation tutorial on topic modeling and gibbs sampling. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies* (pp. 642–647).

De Lathauwer, L., De Moor, B. & Vandewalle, J. (2000). A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, *21*(4), 1253–1278.

Dieng, A. B., Wang, C., Gao, J. & Paisley, J. (2016). Topicrnn: A recurrent neural network with long-range semantic dependency. *arXiv preprint arXiv:1611.01702*.

Ester, M., Kriegel, H.-P., Sander, J. & Xu, X. (1996). Density-based spatial clustering of applications with noise. In *Int. conf. knowledge discovery and data mining* (Vol. 240).

Estivill-Castro, V. & Yang, J. (2000). Fast and robust general purpose clustering algorithms. In *Pacific rim international conference on artificial intelligence* (pp. 208–218).

Fang, Y. & Guo, Y. (2013). A context-aware matrix factorization recommender algorithm. In *2013 ieee 4th international conference on software engineering and service science* (pp. 914–918).

Gallé, M. (2014). Review of bayesian reasoning and machine learning by david barber. *ACM SIGACT News*, *45*(2), 27–29.

Gao, W., Chen, L., Wu, J. & Gao, H. (2015). Manifold-learning based api recommendation for mashup creation. In *2015 ieee international conference on web services* (pp. 432–439).

Gharia, K. N., Desai, P. V. & Gandhi, M. R. (2018). Review paper on novel recommendation. In *2018 second international conference on computing methodologies and communication (iccmc)* (pp. 346–348).

Goldberg, D., Nichols, D., Oki, B. M. & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, *35*(12), 61–71.

Griffiths, T. L., Jordan, M. I., Tenenbaum, J. B. & Blei, D. M. (2004). Hierarchical topic models and the nested chinese restaurant process. In *Advances in neural information processing systems* (pp. 17–24).

Hansen, P. & Jaumard, B. (1997). Cluster analysis and mathematical programming. *Mathematical programming*, *79*(1-3), 191–215.

Hofmann, T. (1999). Probabilistic latent semantic analysis. In *Proceedings of the fifteenth conference on uncertainty in artificial intelligence* (pp. 289–296). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. Retrieved from `http://dl.acm.org/citation.cfm?id=2073796.2073829`

Hofmann, T. (2001). Unsupervised learning by probabilistic latent semantic analysis. *Machine learning*, *42*(1-2), 177–196.

Hofmann, T. (2003). Collaborative filtering via gaussian probabilistic latent semantic analysis. In *Proceedings of the 26th annual international acm sigir conference on research and development in informaion retrieval* (pp. 259–266).

Huang, Z. & Ng, M. K. (1999). A fuzzy k-modes algorithm for clustering categorical data. *IEEE Transactions on Fuzzy Systems*, *7*(4), 446–452.

Jannach, D., Gedikli, F., Karakaya, Z., Juwig, O. et al. (2012). *Recommending hotels based on multi-dimensional customer ratings*. na.

Jannach, D., Lerche, L., Gedikli, F. & Bonnin, G. (2013). What recommenders recommend–an analysis of accuracy, popularity, and sales diversity effects. In *International conference on user modeling, adaptation, and personalization* (pp. 25–37).

Jannach, D., Resnick, P., Tuzhilin, A. & Zanker, M. (2016). Recommender systems—beyond matrix completion. *Communications of the ACM*, *59*(11), 94–102.

Jeh, G. & Widom, J. (2002). Simrank: a measure of structural-context similarity. In *Proceedings of the eighth acm sigkdd international conference on knowledge discovery and data mining* (pp. 538–543).

Jia, Y., Zhang, C., Lu, Q. & Wang, P. (2014). Users' brands preference based on svd++ in recommender systems. In *2014 ieee workshop on advanced research and technology in industry applications (wartia)* (pp. 1175–1178).

Jin, X., Zhou, Y. & Mobasher, B. (2004). Web usage mining based on probabilistic latent semantic analysis. In *Proceedings of the tenth acm sigkdd international conference on knowledge discovery and data mining* (pp. 197–205).

Kawasaki, M. & Hasuike, T. (2017). A recommendation system by collaborative filtering including information and characteristics on users and items. In *2017 ieee symposium series on computational intelligence (ssci)* (pp. 1–8).

Khan, M. W., Chain, G.-Y., Chua, F.-F., Haw, S.-C., Hassan, M. & Saaid, F. A. (2018). Context-aware ontological hybrid recommender system for iptv. In *2018 6th international conference on information and communication technology (icoict)*

(pp. 152–157).

Kim, D., Park, C., Oh, J., Lee, S. & Yu, H. (2016). Convolutional matrix factorization for document context-aware recommendation. In *Proceedings of the 10th acm conference on recommender systems* (pp. 233–240).

Kim, J. K., Cho, Y. H., Kim, W. J., Kim, J. R. & Suh, J. H. (2002). A personalized recommendation procedure for internet shopping support. *Electronic commerce research and applications*, *1*(3-4), 301–313.

Kolatch, E. et al. (2001). Clustering algorithms for spatial databases: A survey. *PDF is available on the Web*, 1–22.

Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R. & Riedl, J. (1997). Grouplens: applying collaborative filtering to usenet news. *Communications of the ACM*, *40*(3), 77–88.

Koren, Y. (2009). Collaborative filtering with temporal dynamics. In *Proceedings of the 15th acm sigkdd international conference on knowledge discovery and data mining* (pp. 447–456).

Koren, Y., Bell, R. & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*(8), 30–37.

Kou, G., Peng, Y. & Wang, G. (2014). Evaluation of clustering algorithms for financial risk analysis using mcdm methods. *Information Sciences*, *275*, 1–12.

Krestel, R., Fankhauser, P. & Nejdl, W. (2009). Latent dirichlet allocation for tag recommendation. In *Proceedings of the third acm conference on recommender systems* (pp. 61–68).

Kumar, R., Verma, B. & Rastogi, S. S. (2014). Social popularity based svd++ recommender system. *International Journal of Computer Applications*, *87*(14).

Kun, Y., Xiao-Ling, W. & Ao-Ying, Z. (2004). Underlying techniques for web services: A survey. In *Journal of software.*

Kwon, O. & Jung, D. (2013). An association model based reasoning method for individualized service recommender. *Expert Systems*, *30*(1), 54–65.

Landauer, T. K. (1998). Learning and representing verbal meaning: The latent semantic analysis theory. *Current Directions in Psychological Science*, *7*(5), 161–164.

Landauer, T. K., Foltz, P. W. & Laham, D. (1998). An introduction to latent semantic analysis. *Discourse processes*, *25*(2-3), 259–284.

Lin, T., Tian, W., Mei, Q. & Cheng, H. (2014). The dual-sparse topic model: mining focused topics and focused terms in short text. In *Proceedings of the 23rd international conference on world wide web* (pp. 539–550).

Liu, C., Sharan, L., Adelson, E. H. & Rosenholtz, R. (2010). Exploring features in a bayesian framework for material recognition. In *2010 ieee computer society conference on computer vision and pattern recognition* (pp. 239–246).

Lops, P., De Gemmis, M. & Semeraro, G. (2011). Content-based recommender systems: State of the art and trends. In *Recommender systems handbook* (pp. 73–105). Springer.

Lu, Y., Mei, Q. & Zhai, C. (2011). Investigating task performance of probabilistic topic models: an empirical study of plsa and lda. *Information Retrieval*, *14*(2), 178–203.

Ma, X. & Ye, L. (2018). Career goal-based e-learning recommendation using enhanced collaborative filtering and prefixspan. *International Journal of Mobile and Blended Learning (IJMBL)*, *10*(3), 23–37.

Madan, S. & Dana, K. J. (2016). Modified balanced iterative reducing and clustering using hierarchies (m-birch) for visual clustering. *Pattern Analysis and Applications*, *19*(4), 1023–1040.

Miaskiewicz, T., Sumner, T. & Kozar, K. A. (2008). A latent semantic analysis methodology for the identification and creation of personas. In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 1501–1510).

Mikolov, T., Chen, K., Corrado, G., Dean, J., Sutskever, L. & Zweig, G. (2013). word2vec. *URL https://code. google. com/p/word2vec*.

Mnih, A. & Salakhutdinov, R. R. (2008). Probabilistic matrix factorization. In *Advances in neural information processing systems* (pp. 1257–1264).

Moura, G. C., Sadre, R. & Pras, A. (2014). Bad neighborhoods on the internet. *IEEE communications magazine*, *52*(7), 132–139.

Munemasa, I., Tomomatsu, Y., Hayashi, K. & Takagi, T. (2018). Deep reinforcement learning for recommender systems. In *2018 international conference on information and communications technology (icoiact)* (pp. 226–233).

Nanopoulos, A., Rafailidis, D., Symeonidis, P. & Manolopoulos, Y. (2010). Musicbox: Personalized music recommendation based on cubic analysis of social tags. *IEEE Transactions on Audio, Speech, and Language Processing*, *18*(2), 407–412.

Neal, R. M. & Hinton, G. E. (1998). A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models* (pp. 355–368). Springer.

Newman, D. J. & Block, S. (2006). Probabilistic topic decomposition of an eighteenth-century american newspaper. *Journal of the American Society for Information Science and Technology*, *57*(6), 753–767.

Ng, R. T. & Han, J. (2002). Clarans: A method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge & Data Engineering*(5), 1003–1016.

Ouhbi, B., Frikh, B., Zemmouri, E. & Abbad, A. (2018). Deep learning based recommender systems. In *2018 ieee 5th international congress on information science and technology (cist)* (pp. 161–166).

Park, H.-S. & Jun, C.-H. (2009). A simple and fast algorithm for k-medoids clustering. *Expert systems with applications*, *36*(2), 3336–3341.

Paterek, A. (2007). Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of kdd cup and workshop* (Vol. 2007, pp. 5–8).

Pazzani, M. J. & Billsus, D. (2007). Content-based recommendation systems. In *The adaptive web* (pp. 325–341). Springer.

Pennacchiotti, M. & Gurumurthy, S. (2011). Investigating topic models for social media user recommendation. In *Proceedings of the 20th international conference companion on world wide web* (pp. 101–102).

Perotte, A. J., Wood, F., Elhadad, N. & Bartlett, N. (2011). Hierarchically supervised latent dirichlet allocation. In *Advances in neural information processing systems*

(pp. 2609–2617).

Qu, Z., Yao, J., Wang, X. & Yin, S. (2018). Attribute weighting and samples sampling for collaborative filtering. In *2018 ieee international conference on big data and smart computing (bigcomp)* (pp. 235–241).

Ramage, D., Hall, D., Nallapati, R. & Manning, C. D. (2009). Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 conference on empirical methods in natural language processing: Volume 1-volume 1* (pp. 248–256).

Ramos, J. et al. (2003). Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning* (Vol. 242, pp. 133–142).

Ricci, F., Rokach, L. & Shapira, B. (2015). Recommender systems: introduction and challenges. In *Recommender systems handbook* (pp. 1–34). Springer.

Rodriguez, A. & Laio, A. (2014). Clustering by fast search and find of density peaks. *Science*, *344*(6191), 1492–1496.

Sadamitsu, K., Mishina, T. & Yamamoto, M. (2007). Topic-based language models using dirichlet mixtures. *Systems and Computers in Japan*, *38*(12), 76–85.

Sahlgren, M. (2002). Towards a flexible model of word meaning. In *Aaai spring symposium* (pp. 25–27).

Salakhutdinov, R., Mnih, A. & Hinton, G. (2007). Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on machine learning* (pp. 791–798).

Sarwar, B. M., Karypis, G., Konstan, J. & Riedl, J. (2002). Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In *Proceedings of the fifth international conference on computer and information technology* (Vol. 1, pp. 291–324).

Shah, A., Pareek, J., Patel, H. & Panchal, N. (2013). Nlkbidb-natural language and keyword based interface to database. In *2013 international conference on advances in computing, communications and informatics (icacci)* (pp. 1569–1576).

Sharma, R., Gopalani, D. & Meena, Y. (2017). Collaborative filtering-based recommender system: Approaches and research challenges. In *2017 3rd international conference on computational intelligence & communication technology (cict)* (pp. 1–6).

Shi, M., Liu, J., Zhou, D., Tang, M. & Cao, B. (2017). We-lda: a word embeddings augmented lda model for web services clustering. In *2017 ieee international conference on web services (icws)* (pp. 9–16).

Shi, Y., Larson, M. & Hanjalic, A. (2014). Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Computing Surveys (CSUR)*, *47*(1), 3.

Shuai, Y., Song, T., Wang, J. & Zhan, W. (2018). Hybrid reliability parameter selection method based on text mining, frequent pattern growth algorithm and fuzzy bayesian network. *Journal of Shanghai Jiaotong University (Science)*, *23*(3), 423–428.

Simon, F. (2006). *Netflix update: try this at home.* (`https://sifter.org/simon/journal/20061211.html`)

Smith, B. & Linden, G. (2017). Two decades of recommender systems at amazon. com. *Ieee internet computing*, *21*(3), 12–18.

Spärck Jones, K. (2004). Idf term weighting and ir research lessons. *Journal of documentation*, *60*(5), 521–523.

Steyvers, M. & Griffiths, T. (2007). Probabilistic topic models. *Handbook of latent semantic analysis*, *427*(7), 424–440.

Su, J.-H., Yeh, H.-H., Philip, S. Y. & Tseng, V. S. (2010). Music recommendation using content and context information mining. *IEEE Intelligent Systems*, *25*(1), 16–26.

Takimoto, S. & Hirose, H. (2009). Recommendation systems and their preference prediction algorithms in a large-scale database. *Information*, *12*(5), 1165–1182.

Tan, Z. & He, L. (2017). An efficient similarity measure for user-based collaborative filtering recommender systems inspired by the physical resonance principle. *IEEE Access*, *5*, 27211–27228.

Tao, Z., Cheung, M., She, J. & Lam, R. (2014). Item recommendation using collaborative filtering in mobile social games: A case study. In *2014 ieee fourth international conference on big data and cloud computing* (pp. 293–297).

Teh, Y. W., Jordan, M. I., Beal, M. J. & Blei, D. M. (2005). Sharing clusters among related groups: Hierarchical dirichlet processes. In *Advances in neural information processing systems* (pp. 1385–1392).

TIAN, Y., KOCHHAR, P. S. & LO, D. (n.d.). An exploratory study of functionality and learning resources of web apis on programmableweb.(2017). In *Ease'17: Proceedings of the 21st international conference on evaluation and assessment in software engineering, karlskrona, sweden, june 15* (Vol. 16, pp. 202–207).

Wallach, H. M., Mimno, D. M. & McCallum, A. (2009). Rethinking lda: Why priors matter. In *Advances in neural information processing systems* (pp. 1973–1981).

Wen-Shung Tai, D., Wu, H.-J. & Li, P.-H. (2008). Effective e-learning recommendation system based on self-organizing maps and association mining. *the electronic library*, *26*(3), 329–344.

Wilson, D. J. & Zhang, W. (2016). Integrating topic models and latent factors for recommendation. *arXiv preprint arXiv:1610.09077*.

Wooff, D. (2004). Logistic regression: a self-learning text. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, *167*(1), 192–194.

Wu, Y., Xi, S., Yao, Y., Xu, F., Tong, H. & Lu, J. (2018). Guiding supervised topic modeling for content based tag recommendation. *Neurocomputing*, *314*, 479–489.

Xia, B., Fan, Y., Wu, C., Bai, B. & Zhang, J. (2017). A method for predicting service deprecation in service systems. *Tsinghua Science and Technology*, *22*(01), 52–61.

Xu, G., Zhang, Y. & Zhou, X. (2005). A web recommendation technique based on probabilistic latent semantic analysis. In *International conference on web information systems engineering* (pp. 15–28).

Xu, H., Zeng, W., Gui, J., Qu, P., Zhu, X. & Wang, L. (2015). Exploring similarity between academic paper and patent based on latent semantic analysis and vector space model. In *2015 12th international conference on fuzzy systems and knowledge discovery (fskd)* (pp. 801–805).

Xu, R. & Wunsch, D. C. (2005). Survey of clustering algorithms.

Yan, Y., Fan, J. & Mohamed, K. (2008). Survey of clustering validity evaluation [j]. *Application Research of Computers*, *6*.

Yu, L., Jiang, L., Zhang, L. & Wang, D. (2018). Weight adjusted naive bayes. In *2018 ieee 30th international conference on tools with artificial intelligence (ictai)* (pp. 825–831).

Zhang, W., Yoshida, T. & Tang, X. (2011). A comparative study of tf* idf, lsi and multi-words for text classification. *Expert Systems with Applications*, *38*(3), 2758–2765.

Zhang, Y., Liu, H. & Deng, B. (2013). Evolutionary clustering with dbscan. In *2013 ninth international conference on natural computation (icnc)* (pp. 923–928).

Zhou, L., Tang, H. & Dong, T. (2017). A hybrid collaborative filtering recommendation model-based on complex attribute of goods. In *2017 international conference on security, pattern analysis, and cybernetics (spac)* (pp. 232–241).

Zhou, Z., Sellami, M., Gaaloul, W., Barhamgi, M. & Defude, B. (2013). Data providing services clustering and management for facilitating service discovery and replacement. *IEEE Transactions on Automation Science and Engineering*, *10*(4), 1131–1146.

Appendix A

Code Implementation

All codes and output can be found in the following link:

https://drive.google.com/open?id=1siwe9tlFT4HFcytPYdL-vKRumyXtm6QV