INCORPORATING SERVICE PROXIMITY INTO WEB SERVICE RECOMMENDATION VIA TENSORS DECOMPOSITION

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
Zhentao Wu
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.

Zhentao Wu

Signature of candidate

Acknowledgements

Throughout the writing of this dissertation, I have received a great deal of support and assistance.

I would first like to thank my primary supervisor, Dr. Jian Yu, whose expertise was invaluable in the formulating of the research topic and methodology in particular. He provided me with the tools that I needed to choose the right direction and successfully complete my dissertation.

I would also like to thank my secondary supervisor, Dr. Sira Yongchareon, for their valuable guidance.

In addition, I would like to thank my parents for their wise counsel and sympathetic ear. You are always there for me. Finally, there are my friends, who were of great support in deliberating over our problems and findings, as well as providing a happy distraction to rest my mind outside of my research.

Abstract

The sparseness of Mashup-API rating matrix coupled with cold-start and scalability issues have been identified as the most critical challenges that affect most Collaborative filtering based Web APIs recommendation solution. Sparseness deteriorates the rating prediction accuracy. Several Web-API recommendation approaches employ basic collaborative filtering technique which operates on second-order matrices or tensors by decomposing the Mashup-API interaction matrix into two low-rank matrix approximations, and then make prediction based on the factorized tensors. While most existing CF, Matrix factorization-based Web-API recommendation approaches have shown promising improvement in recommendation results, one limitation is that they only focus on 2-dimensional data model in which historical interaction between Mashup-API are mainly used. However, recent works in recommendation domain show that by incorporating additional information into the rating data, Web-API rating prediction accuracy can be enhanced. Inspired by these works, this research proposes a collaborative Filtering method based *Tensors factorization*, an extension of Matrix factorization-that exploits the *ternary* relation among three key entities in Web service ecosystem *Mashup-API-Proximity*. Modelling the Web-API rating data with Tensor decomposition technique enables incorporation proximity information as third additional entity into Web service recommendation application to improve prediction accuracy. Specifically, we employ High Order Singular Value Decomposition approach with regularization term to extend the traditional *Mashup-API* matrix into *Mashup-API-Proximity* tensors.

Experimental analysis on ProgrammableWeb dataset shows promising results compare with some state-of-the-art approaches.

Keywords: Tensors Factorization, ProgrammableWeb, High Order Singular Value Decomposition, Recommendation, Mashups, Web-API recommendation

Contents

Copyright 2

Declaration 3

Acknowledgements 4

Abstract 5

1 Introduction 12
- 1.1 Aim 12
- 1.2 Research Questions 14
- 1.3 Background 15
  - 1.3.1 Recommendation systems and Big Data 15
  - 1.3.2 Recommending Web Services 16
- 1.4 Research Motivation and Significance 20
- 1.5 Research Scope and Methodology 23
- 1.6 Research contributions 26
- 1.7 Thesis Structure 27

2 Literature Review 28
- 2.1 Introduction 28
- 2.2 Web Service Recommendation 30
  - 2.2.1 Content-based Approach 31
  - 2.2.2 QoS-based Approach 31
  - 2.2.3 Context-Aware approach 33
  - 2.2.4 Semantic-Based approach 33
- 2.3 Collaborative Filtering Techniques 34
  - 2.3.1 Model-Based Recommendation Methods 36
  - 2.3.2 Memory-Based Recommendation Methods 37
  - 2.3.3 Challenges of collaborative filtering techniques 39
  - 2.3.4 Common Solutions to collaborative filtering challenges . . . 41
  - 2.3.5 Collaborative Filtering Methods for Web Service Recommendation 42
  - 2.3.6 Clustering-Based Recommendation Approach 43

2.3.7 Hybrid Service Recommendation 45
2.3.8 Hybrid Web Service Recommendation 48
2.4 Matrix Factorization 49
2.5 Matrix Decomposition Models 51
2.5.1 Eigenvalue Decomposition Method 52
2.5.2 Singular Value Decomposition 53
2.5.3 Principal Component Analysis 55
2.5.4 Probability Matrix Factorization 55
2.5.5 Non-Negative Matrix Factorization 57
2.5.6 Matrix Factorization in Web service Recommendation 58
2.6 Tensors Decomposition Techniques 60
2.6.1 Tensors Overview 61
2.6.2 Tucker Decomposition And Higher Order Singular Value Decomposition 61
2.6.3 Parallel Factor Analysis (PARAFAC) 64
2.7 Pairwise Interaction Tensor Factorization 66

3 Research Method **67**
3.1 Overview 67
3.2 Data Acquisition and Processing 69
3.3 Mashup-Oriented MF-Based Recommendation 69
3.4 Tensors Factorization 74
3.4.1 Need for Tensors in Recommender Systems 74
3.4.2 Notations and Operations 75
3.5 High-Order Singular Value Decomposition on Tensors 77
3.5.1 HOSVD Algorithm Description 77
3.5.2 HOSVD Decomposition with Single Contextual Variable . . . 78
3.5.3 Loss Function 80
3.5.4 Regularization 81
3.5.5 Basic Optimization 82
3.6 Geographical Distance Between Services 83
3.6.1 Estimating Geographical Proximity Score 83

4 Analysis **85**
4.1 Introduction 85
4.2 Tools 86
4.2.1 Pandas module 86
4.2.2 Sktensor and Tensorly modules 87
4.2.3 Numpy module 87
4.2.4 Matplotlib module 87
4.3 Implementation 87
4.3.1 Three-dimensional tensor construction 87
4.3.2 Sampling the training set and testing set 90
4.3.3 Web API Prediction With Tensor Decomposition 91

4.3.4 The performance of HOSVD 93
4.3.5 Comparison with PMF 94
4.4 Findings 97
4.4.1 Data sparsity 97
4.4.2 The performance of HOSVD and PMF with different size training set 98
4.4.3 The denoising influence on performance of HOSVD 99
4.4.4 Prediction accuracy comparison between HOSVD and PMF . 99

5 Discussion **102**
5.1 Results Assessment 102
5.1.1 Impact of Dimensionality 103
5.1.2 Impact of HOSVD Tensor Density 103
5.1.3 Impact of Regularization Parameter 104
5.2 Contributions 104

6 Conclusion **106**
6.1 Challenges 106
6.2 Future Work 107

References **109**

Appendices **115**

List of Tables

2.1 Hybridization methods in Recommendation 46

3.1 Statistics of the ProgrammableWeb Dataset 70
3.2 Sample Specification of Web-API (a) and Mashup Profile (b) in PW dataset 70

4.1 The RMSE and MAE of HOSVD and PMF model with different size training set. 98

List of Figures

2.1 Literature Review Layout 29
2.2 Collaborative Filtering Process 36
2.3 Summary of Common Collaborative Filtering Techniques 39
2.4 Summary Classification of Recommender Systems Research 50
2.5 Left pane: Graphical Model of PMF. Right Pane: Constrained PMF . . 57
2.6 The unfolding of 3rd-order tensor A, in the three modes 62
2.7 Tensor decomposition in STS of A in the three modes – 64
2.8 The graphical representation of PARAFAC with diagonal core tensor and the factorization dimensionality (equal for the three modes) 65
2.9 The graphical representation of 2-components PARAFAC of data array X 66

3.1 Matrix Factorization for recommendation 71
3.2 Illustration of four-order tensor Unfolding 76
3.3 An illustration of 3-order HOSVD tensor factorization model of Web service data 79

4.1 The heat-map of Mashup-API matrix 97
4.2 The RMSE of HOSVD and PMF based training set with different size 99
4.3 The influences of denoising ratio on the performance of HOSVD . . . 100
4.4 The comparison between HOSVD and MAE performance 101

Chapter 1

Introduction

1.1 Aim

Recommender systems play an important role in many real-world applications that support users in managing large information space and helping users to overcome the current information overload problem on the internet. However,these systems are generally faced with challenging issues such as *data sparsity*, *scalability* and *cold start* problem. For *data sparsity*, the recent prevalence of open data source coupled with the increasing growth of the number of users and items in various online, distributed system platforms, increases the sparseness of user-to-item rating data. In this case, only a small fraction of items usually have explicit rating information defined by users. Eventually, the data sparseness deteriorates the rating prediction accuracy of conventional collaborative filtering (CF) techniques.

Conventional recommender approaches such as *item-based* (Linden, Smith & York, 2003) or *user-based* (Isinkaye, Folajimi & Ojokoh, 2015) and popular *Matrix Factorization method* are mainly based on two major types entities *user-item*. That is, they operate on second-order tensors representing relations or interactions between users (such as customers/consumers) and items (products, services and resources). Since in real-world, multiple information about object interactions usually involve more than two participating entities, and several additional information about the state of the interactions like location, time, or mood of the user are usually available (W. Wu et al., 2017). Recent techniques are leveraging various auxiliary information currently available on the web to enhance rating prediction accuracy. However, while various side/auxiliary information is available to support the performance of recommendation applications, incorporating multifaceted information such as ternary relation between independent objects into rating matrix is a challenging task. Moreover, many existing approaches for prediction and recommendation can neither handle heterogeneous, large-scale datasets nor deal with *cold-start* problem.

In service-oriented computing, various web service recommendation approaches including memory-based, content-based, social-based, and context-based and other hybrid collaborative filtering approach exit (Bobadilla, Ortega, Hernando & Gutiérrez, 2013). Many of these approaches explored the historical, co-invocation and interaction information that exist between Web-service compositions *Mashups* and component services like *Web-APIs* to solve the discovery problem and recommendation/service selection problems that exist in the service computing domain (Yao, Wang, Sheng, Benatallah & Huang, 2018; B. Cao et al., 2017), Generally, most of these works focus on using two entities : *mashups*, (which could be related to Web-APIs users) and Web-APIs (which is the *item* in Web service recommendation context) in their applications. Several other efforts also integrate ***Quality of Service*** (Z. Zheng, Zhang & Lyu, 2012; Z. Zheng, Ma, Lyu & King, 2012), ***Location*** (X. Chen, Zheng, Yu & Lyu, 2013) and ***Trust*** (Su, Xiao, Liu, Zhang & Zhang, 2017) information into their prediction/recommendation systems. However, most of these approaches consider two-dimensional matrix representation techniques, and thus, can not effectively exploit the underlying latent features that exist within multiple object interactions. As a dominant and common implementation of the CF method, the basic matrix factorization technique enables decomposition of the Mashup-API relationship matrix into *2* low-rank matrix approximations and makes predictions based on the resulting factorized matrices (Yao, Wang et al., 2018). Previous research works in web service recommendation has shown that more effective, multi-dimensional information could lead to the enhancement of recommendation results. However, most of the existing web service recommendation solutions only used part of the information available in service ecosystem. One other main challenges of building a recommender system using heterogeneous information is to systematically define features to represent the different types of relationships between entities, and learn, exploit the importance of each relationship type (X. Yu et al., 2013).

This research work proposes an extension of matrix factorization technique that extends the two-dimensional *mashup-api* into a three-dimensional *mashup-api-proximity* matrix, which enables incorporation of *proximity* as third additional entity into Web service recommendation application with the aim of enhancing prediction accuracy. We aim to introduce proximity as context into existing explicit correlations among diverse Web services. Specifically, this research proposes a *Higher-Order Singular Value Decomposition* (HOSVD) approach with implicit correlation regularization to solve and enhance the accuracy of Web-API recommendation application. The intention is to apply HOSVD technique to discover the latent relationships and patterns through representation of web-service information in a *3-order tensor*. We apply tensor factorization techniques on the diffused mashup preferences in order to calculate latent representations for mashups and Web-APIs accordingly. We then combine these latent features and define a global recommendation model.

1.2 Research Questions

With the objective of incorporating explicit correlations information derived from service co-invocation and proximity information using a three-dimensional tensor which is based on a generalization of MF, this research aims to answer the following research questions are to be answered:

1. How to cast the Web-API recommendation problem as a third-order task by completing the missing or unobserved entries in the rating matrix?
2. What is the impact of context information in a three-dimensional matrix factorization prediction model when compared with the conventional two-dimensional Probabilistic Matrix Factorization (PMF) model?- Impact of dimensionality

The above questions will address the application of multi-variate models like MF and its extensions in capturing latent relationships that exist between Web-APIs.

1.3 Background

This section provides background knowledge on the key components of this research work. First, a smart overview of the *Recommendation systems and Big Data*, Second, it provides the background of knowledge of *Collaborative filtering with Matrix-factorization* based Web service recommendation systems and Finally introduced the *Motivation* behind the approach used in this research.

1.3.1 Recommendation systems and Big Data

In recent years, recommendation systems have become increasingly important in service-oriented computing domain. The use of these systems has exploded over the last decades in many industries, making personalized recommendation pervasive online and promoting development of user-centric services. These systems are popular means of assisting users in online services to discover and recommend the whole range of items, including consumer products, movies, friends and web services from a large resource collection.

Companies such as $Facebook$, $Amazon$, $Neflix$, $eBay$, $Google$ and $Twitter$ employ recommender systems in their services to improve customer intimacy and satisfaction. The growth of recommender systems has been in parallel with the web. The emergence of Web 2.0, coupled with the continuous development of various web services, have led to a rapid increase in the amount of digital information generated on various web platforms. For instant, with the increase in the development of e-commerce platforms, more auxiliary data that captures features of both users and items are now available for enhancing the performance of recommender systems. Online social networks contain gigabytes of data, which can be mined to provide recommendations support. The boom of social media has also contributed to the development in recommendation, especially in social recommendation. Generally, social recommendation focuses on modelling social network information as regularization terms to constrain the matrix factorization framework (J. Zheng et al., 2017).They generate similarity of users by leveraging rich social interactions among users, for instance, friendships in Facebook, following relations on Twitter. Emerging technologies such as Big data has played a significant role in developing recommender systems (Y. Zhang, Chen, Mao, Hu & Leung, 2014). The use of $BigData$ technologies enables discovery of latent information within data with large volume features, most of which are now used to support item recommendation.

1.3.2 Recommending Web Services

With the remarkable development in service-oriented computing domain coupled with improvements in internet technologies, there has been a rapid surge in the number of Web services (also referred to as Web APIs) available on the internet. Several web service registries, portals and marketplaces have also emerged, enabling service developers and providers to expose, manage and advertise their services. Service consumers can now explore these repositories to discover services of their interest.

For example, as of April 2019, the current largest, online Web-API repository called *ProgrammableWeb* [1] has over 20,320 Web-APIs belonging to more than 400 predefined categories, and over 7,000 mashups (service compositions). Similarly, a popular Web service marketplace, *Mashape* [2] currently has over 10,000 collections of public and private APIs with about 100,000 records of engaged developers around the globe. The continual growth of these service ecosystems conveys the popular emerging service economy (Tan, Fan, Ghoneim, Hossain & Dustdar, 2016). This also reflects both economic and social impacts of web services in the software development industry. Diverse services can now be used for composing new, value-added application − also known as "*mashups* " −, which combine several Web-APIs from different sources to satisfy complex user requirements. Web-APIs have become ubiquitous due to majority of software applications/services currently been offered as some form of Web-APIs (Adeleye, Yu, Yongchareon, Sheng & Yang, 2019).

The diversity and rapid increase in the number of Web-APIs on the internet poses a great challenge to their consumption and reusability. While there are tens of thousands of web services with diverse functionalities currently available in various repositories, the discovery and selection of appropriate services capable of satisfying some specific and complex user requirements is a great challenge. Software developers and other service users still find it very challenging to select suitable services from a pool of functionally similar and related services, especially for service composition purposes. Recently, recommendation techniques have been employed to tackle service discovery and selection complexities. Web service recommendation involves automatic identification of usefulness or suitability of web services, selecting candidate services based on users' requirements (or behaviour analysis) and proactively recommending most suitable services to end users. Various researchers have made efforts to proactively

[1]https://www.programmableweb.com
[2]https://www.mashape.com

recommend services to users based on certain parameters and preferences. Thus, web service recommendation became an active process of service search, discovery and selection that can be facilitated with various computer science and mathematical techniques such data analysis, machine learning, deep learning, graph theory and probabilistic approaches, to enhance accuracy in prediction (Yao, Sheng, Ngu, Yu & Segev, 2015). Even though existing web service recommendation techniques show improvements, the recommendation tasks continue to pose a great challenge for service engineers due to rapid, continuous increase in the number of web services. Generally, recommender systems provide support for selecting products and conventional services; however, their direct application to web services is not straightforward due to the following challenges:

- Current Scarcity of user feedbacks on web services (Lecue, 2010) and High degree of uncertainty in the feedbacks (Users feedbacks are sometimes subjective) (L. Chen, Wang, Yu, Zheng & Wu, 2013).
- Lack of specific quality of services (QoS) and context information (Su et al., 2017).
- Need to fine-tune web services to the requirement of intended user (L. Liu, Lecue & Mehandjiev, 2013).
- Inadequate formal semantic specifications for describing web services (Yao, Wang et al., 2018).
- Lack of social-awareness among web services (Adeleye et al., 2019).
- Keyword-based approaches have poor recommendation performance and are heavily dependent on correct and complex queries from users.

Currently, most service consumers rely on manual searching (keyword-based) of service registries like *ProgrammableWeb* to find and select relevant web services. Clearly, such search is ineffective and time-consuming. Therefore, effective recommendation approach is required to support service consumers in selecting suitable services for a particular requirement or a given service composition task. Most existing web service recommendation solutions are either based *Content-based approach* such as keyword-dominant web service search engines or *Collaborative filtering and Matrix Factorization approach*. These two recommendation techniques possess the following limitations:

- The main idea of *Content-based* methods is to exploit Information about user's preferences, profiles and web service descriptions including semantic information of service interfaces, functionality descriptions and so on. For instance, a common approach is recommended services for a particular composition based on the similarity between mashups requirements and service profiles. However, the method suffers several set backs: (i) Lack or inadequate formal semantic description for Web-APIs. This is common especially for newly emerged web services, which lack formal semantic information and attributes like the traditional web services. Attributes such as *tags*, *functional category*, *input and output description* are lacking in most newly developed and published services. Without this information, it is somewhat challenging to implement an effective content-based web service recommender system. (ii) Another set back is the problem of over-specialization (Bobadilla et al., 2013), where the same type of services a recommended.

- For *Collaborative filtering* (CF) recommendation methods are based on opinion of past users with similar preferences to the new users. CF approaches rely on rating data such as Quality of Service (QoS) values. For web services, rating value is either determined by service providers (for example, as throughput and price) or derived as an estimation of user-service invocation, service-service co-invocation transactions. Another rating parameter commonly used is user satisfaction, which is subjective. One of the main challenges of CF-based approach is difficulty in obtaining QoS value for web services. Another challenge of CF algorithms is the sparsity of rating matrix and the ever-growing nature of rating data (Bokde, Girase & Mukhopadhyay, 2015). These challenges are usually tackled with various Matrix factorization models such as probabilistic matrix factorization, Single value decomposition (SVD) and principal component analysis (PCA) (discussed in next chapter).

Other recent methods include network-based and hybrid approaches. Hybrid recommendation approaches integrate two or more of the above methods to recommend appropriate services to users. In other to improve recommendation processes, some researchers tried to leverage the capabilities of multiple techniques by combining them.(both network-based and hybrid approaches are later discussed in Literature review chapter).

1.4 Research Motivation and Significance

In this era of information overload and data explosion, internet and web users have to struggle through a rapidly, continually increasing amount of both relevant and irrelevant information. Specifically, in service-oriented computing, tens of thousands of web services are currently available on web and this number has been continuously soaring. Web services have had enormous impact on the Web as a potential means for supporting a distributed service-based economy. However, the discovery and reuse of appropriate web services on a global scale especially for complex service requirements task are still very limited and challenging. Moreover, the diversity of web services present a unique challenge to service composition; it is now more challenging to select appropriate services from myriad of functionally similar services. While there are billions of web pages currently available on the web, very few numbers of web services are publicly available to service consumers. Traditional approach to searching through the web using keywords have proven inefficient. Thus, managing web service information overload and data explosion requires the support of intelligent systems that can leverage the available information, process and filter faster than humans and recommend services based on user's requirements. To satisfy these diverse needs, user-centric or personalized recommender systems have emerged.

Conventional web service recommender systems deal with two major types of entities, which include users service requirements and service profiles (e.g. service functional descriptions or historical information). A popular approach is to recommend web services based on the semantic similarity between user's service requirement and a combination of service description and co-invocation information. Due to increasing complexity of user's service requirement and the surge in large-scale information ecosystems, multi-layer information is required to improve service recommendation quality. Generally, dominant frameworks for web service recommendation are logically two dimensional focusing on the interaction between web service composition, (which usually reflects historical usage of web services) and web services. This exemplifies a sort of interaction that manifests between traditional users (consumers) and items (products), and are normally characterized by a single relation (Vahedian, Burke & Mobasher, 2017). However, the ever-growing increase in social ecosystems, where multi-entity interaction and correlation occur across different domains and context have created multi-dimensional space that reflects complex heterogeneous relationship between various entities. Consequently, large-scale Heterogeneous Information Network that consists of interconnected entities with multi-type relations can exist (Jamali & Lakshmanan, 2013).

Previous studies suggest that more effective utilization of side information or auxiliary data can help improve the quality of recommender system (Yang, Lei, Liu & Li, 2017), (X. Liu et al., 2012), (X. Yu et al., 2014). Through open data initiative and rapid development of APIs, huge amount of auxiliary data, which can be used to improve web service recommendation quality are increasingly becoming available. Although auxiliary data may contain information that could be useful for enhancing recommender system functions, this information and its associated complexity poses the following challenges for the systems:

- The difficulty of integrating a wide variety (heterogeneous) of data effectively into a recommendation framework (Vahedian et al., 2017).
- The challenges of modelling and utilizing these complex and heterogeneous data in enhancing recommender systems (Shi, Hu, Zhao & Philip, 2019) .
- The challenge of developing a generic approach to model these varying data (of different types and attributes) from different platform.

Generally, a handful hybrid recommendation solutions exist to tackle related problems similar to the ones mentioned above, among them, information network-based recommendation approach, which utilizes relationship information between users and items are increasingly gaining more attention in the research world. Most previous prediction approaches based on information network are considered unsupervised methods. These methods are mainly based on direct analysis of graphs or topological structure of the network to get the likelihood that new links might appear (Do, Pham, Phan & Nguyen, 2018). However, this kind of approach does not gain optimal effectiveness and accuracy in the complex dynamic network. Supervised methods allow the system to learn from the previous network dataset to generate the predictive model, which is applied to compute the probability that two nodes will be connected in the future. Efforts in this area are mainly focused on single relationship types and homogeneous networks that do not provide comprehensive semantic meanings behind different links. For example,some algorithms only consider part of information in social network (like Mashup-APIs affiliation network, user-user or item-item network). However, in many modern cases, recommendation problem usually exists in a heterogeneous information network environment, where there are several layers of multi-type relationship. Hence, the homogeneous network method may not be applicable.

1.5 Research Scope and Methodology

This research scope extends to the study and review of recommender systems with focus on various techniques and methods applicable to providing web service recommendation solution. In order to effectively leverage available online information to enhance the quality of web service recommendations, this research work study several collaborative filtering techniques to induce a model from rating a matrix and utilize the model to perform recommendation. Specifically, by modelling web service information with location information as three-order tensors, correlation and interaction among web services, service composition and location can be captured using the core tensor in reduced-dimension form. This study also provides an overview and application of higher-order tensors with respect recommender systems implementation and enhancement. Tensor decomposition is applied to web service data arrays for extracting service attributes.

Recommender systems mainly base their suggestions on rating data of two entities (users and items), which are often placed in a matrix with one representing users and the other representing items of interest. These ratings are given explicitly by users creating a sparse user-item rating matrix because an individual user is likely to rate only a small fraction of the items that belong to the item set. Another challenging issue with this user-item rating matrix is scalability of data (i.e., the large number of possible registered users or inserted items), which may affect the time performance of a recommendation algorithm.

We can deal with all aforementioned challenges by applying matrix decomposition methods (also known as factorization methods). Matrix factorization denotes a process, where a matrix is factorized into a product of matrices. A matrix factorization method is useful for solving plenty of problems, both analytical and numerical; an example of a numerical problem is the solution of linear equations and eigenvalue problems. Its importance relies on the exploitation of latent associations that exist in the data among participating entities (e.g., between users and items). In a trivial form, the matrix factorization method uses two matrices, which hold the information of correlation between the user-feature and item-feature factors, respectively.

Big data has association, high dimension and multivariate features. Tensor as an expression format of high dimension data structure can be used for selecting these high dimensional features. Big data application organize data as tensor format and use high dimensional array theory to process and analysis it. Big data applications organize data in tensor format and use high dimensional array theory to process and analysis it. Tucker proposed multidimensional expended decomposition model for similar factor analysis of the third-order tensors in 1966[51], and had further development by Kroonenberg in the 1980s, and define the Tucker3, Tucher3ALS, 3-Mode SVD and 3-Mode PCA. In the next few years, this decomposition was expended to n-order tensor and concluded by Lathauwer; they call these compositions as HOSVD, N-way SVD and multilinear SVD [40]. In the past years, HOSVD was used in analysis the data based on experience, especially in the chemo metrics and psychometrics. Some methods related to HOSVD was found independently, it makes the related literature is mixed, and the general mathematics theories for these methods are very few, these methods mainly used for the special data type.

In the big data era, HOSVD has many applications in big data analysis, however, these studies nor consider the implementation effectiveness problem, sometimes, there are a lot of methods to process the huge amount number of tensors has low efficiency make it cannot direct used in the real big data scene. Thus the study of higher effective calculation approach of HOSVD is important and urgent.

HOSVD is a method for mapping SVD to high dimensional space; the calculation relies on the calculation of matrix SVD. Thus, the study of HOSVD cannot do without study of matrix SVD. So far, the study of HOSVD mainly focus on some algorithm application effect enhancement and interpretation, research on HOSVD parallelization not a lot, and has a lot of researches on matrix SVD parallelization calculation. For the huge amount of data in tensor, can use HOSVD to compress the data, it can save the storage space.

To solve the problem, avoid the complex calculation process, improve accuracy choose the SVD and HOSVD method. According to build the third-order tensor space matrix based on user, tag and project, use the good characteristic of SVD to process the sparse data problem, expand in the multidimensional tensor space matrix. The approach can guarantee the complete data structure and delete the blank part in the data to reduce the sparse data and produce the recommended results. In addition, use HOSVD can effective reduce the data redundant and improve recommend precision.

As the development of Web 2.0 technique, Social Network Service provides a new platform for the interaction between users. Such as Facebook, Twitter, Instagram, let many network users to form a publishing and sharing ecosystem of resources, so that users can actively communicate with each other in the circle. If users find resources that meet their interests and hobbies, they often make “tags” to facilitate retrieval. At the same time, the tags that users get after tagging their favourite image, web link, book, music and video resources under different network platforms can provide a reference for other users to find the network resources they need. The concept of Folksonomy arises from this.

Folksonomy is a process in which many information users choose the appropriate network information resources according to their own needs and determine the matching social tags according to their own cognitive level. It emphasizes process and information that involved. Early websites with folksonomy features have strong “tagging” characteristics, such as Movie Lens, people can classify the interested movies according to different tags and grade for the movie. And some typical websites such as picture website Flicker, video website YouTube also have the same characteristic.

The information contained in tags can be an effective basis for recommendation to other users. Therefore, personalized recommendation system has been greatly developed in folksonomy websites. In addition to tags, other social media included in folksonomy websites are also common raw data in recommendation systems. Because of the relationship among users, tags and resources in public annotation website, there is a great space to develop the algorithm of recommendation system. From the more mature collaborative filtering recommendation system to label-based tag recommendation system, each method has its defects and shortcomings. How to make full use of the original data and develop a more efficient, accurate and personalized recommendation system for folksonomy websites has become a hot issue in academic research.

1.6 Research contributions

The contribution of this research work is as follows:
Firstly, with the aim of tackling the sparseness of rating data, this work propose a fusion method for integrating independent, multi-facet, heterogeneous information for web services into a single representation that can be leverage to enhance service recommendation .
Secondly, this research investigates the challenge of Web-APIs recommendation for mashup with a regularized three-order Matrix factorization approach that decomposes *APIs-Mashups-proximity* matrix into three low-dimensional matrices. Secondly, this work investigates the problem of Web-APIs prediction for mashup by incorporating

regularized terms into the weighted Frobenious norm in three-order matrix factorization.

Finally, the report starts from the tensor structure, study how to decomposition tensor and achieve the distributed storage of tensor data, avoid the situation of a general computer cannot process the huge amount of data in tensor. Then in the distributed storage achieve the SVD calculation method to decomposition the data, the experiment express that it can solve the data sparse problem, it provides the idea for SVD in the application of big data analysis. And then based on it, provide HOSVD method based on it, the experiment shows that save the storage space, and improve the accuracy without any other changes, accelerate advantage of HOSVD be more distinct, it has the important significance about improving processing and running efficiency of HOSVD analysis application when incremental data comes.

1.7 Thesis Structure

The rest of this paper is organized into five parts: Chapter 2 presents a literature review of introduces and explores the fundamental concepts related to this research work, and explores previous efforts in providing web service discovery and recommendation solutions. The chapter identifies the various models and characteristics which are likely to be employed in this research. Chapter 3 describes the research methods in detail, provides the preliminary knowledge overview of tensors. Then presents the representation and construction of the tensor model. Finally , the prediction algorithm based on HOSVD decomposition in the proposed recommendation framework is illustrated. Chapter 4 displays the implementation and examination for the hypotheses, and the corresponding measurement. Chapter 5 presents the analysis and discussion of our results. Chapter 6 is a general conclusion part for this research. Moreover, the challenges regarding the research and the future work are proposed in this chapter.

Chapter 2

Literature Review

2.1 Introduction

Recently, there have been several research works conducted in service-oriented computing domain to tackle various issues, and fill research gaps related to Web service discovery, composition and recommendation. Several techniques and models have been explored to support these activities. This chapter introduces and explores the fundamental concepts related to this research work, and explores previous efforts in providing web service discovery and recommendation solutions. The first part of the chapter presents the general overview of Web service recommendation and discovery with respect to service computing. As shown in Figure 2.1, various categories of Web service recommender systems are discussed with different kind of techniques used to implement these systems. In general, three different recommender systems are considered : (i) Content-based filtering techniques, (ii) Collaborative Filtering techniques and (iii) Hybrid approaches. The collaborative filtering technique is further divided into two separate groups: (a) Model-based approaches, and (b) Memory-based approaches.

As for model-based, various data-mining and machine-learning techniques are employed to achieve the recommendation objectives. Most common techniques are

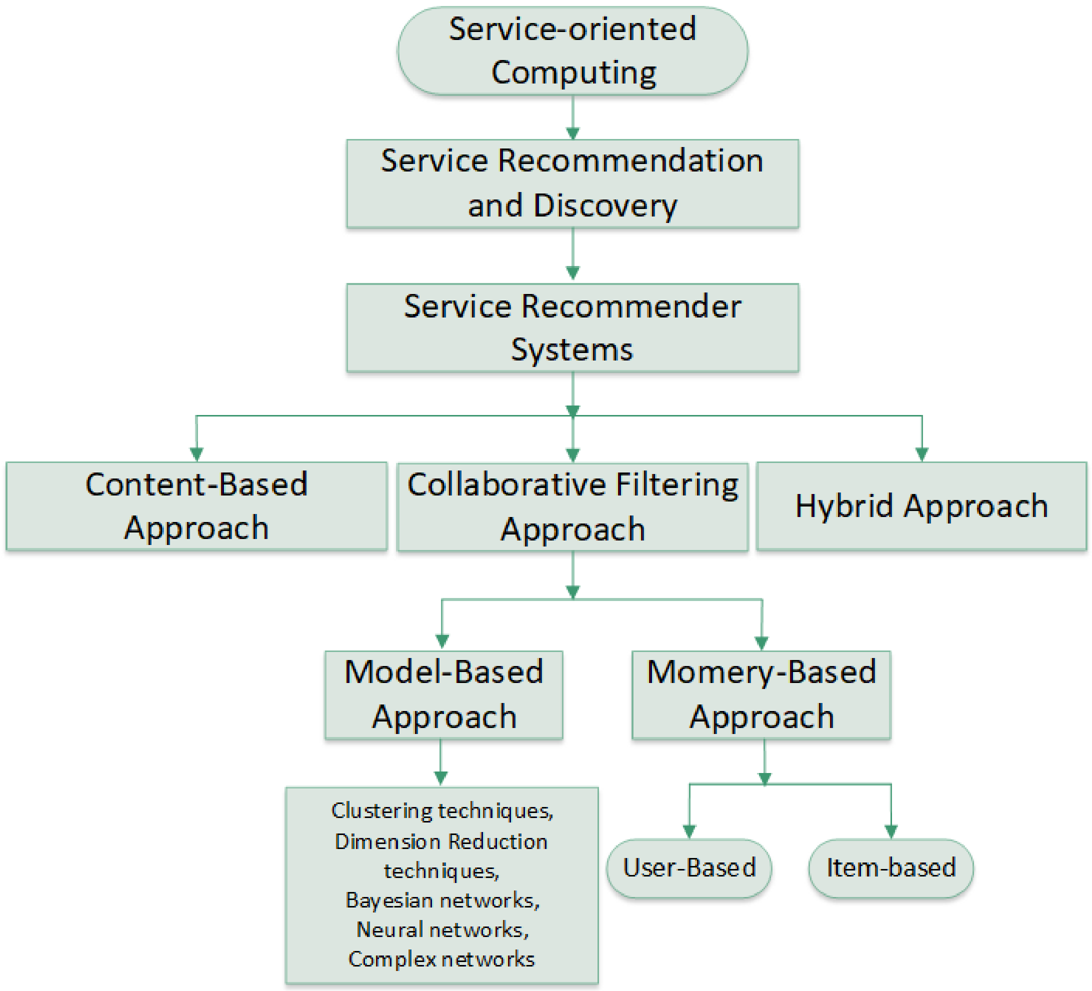

Service-oriented Computing
Service Recommendation and Discovery
Service Recommender Systems
Content-Based Approach
Collaborative Filtering Approach
Hybrid Approach
Model-Based Approach
Momery-Based Approach
Clustering techniques, Dimension Reduction techniques, Bayesian networks, Neural networks, Complex networks
User-Based
Item-based

Figure 2.1: Literature Review Layout

discussed in this chapter. Model-based techniques utilize user-item matrices to identify relationships and interactions between recommendation entities. Recently, dimension reduction techniques such as Matrix completion techniques, Singular Value Decomposition (SVD) techniques, Matrix factorization (MF), Tensor decomposition techniques have been employed by researchers to simplify and implement model-based, collaborative filtering approaches. These techniques and other related methods are also reviewed in this chapter. Finally, various decomposition methods from the perspective of tensor are discussed. Different recommender systems evaluation metrics are discussed with respect to each techniques covered in this review.

2.2 Web Service Recommendation

Web service recommendation involves automatic identification of usefulness or suitability of web services, selecting candidate services based on users' requirements (or behaviour analysis) and proactively recommending most suitable services to end users. The proliferation of web services coupled with myriad of functionally similar web services makes it harder for service consumers to select most suitable web services among the large amount of service candidates. Generally, recommender systems provide support for selecting products and conventional services; however, their direct application to web services is not straightforward due to the following challenges:

- Current Scarcity of user feedbacks on web services (L. Liu et al., 2013) and High degree of uncertainty in the feedbacks (Users feedbacks are sometimes subjective) (X. Chen, Zheng & Lyu, 2014)
- Lack of specific quality of services (QoS) and context information (Su et al., 2017)
- Need to fine-tune web services to the requirement of intended user. (L. Liu et al., 2013)
- Inadequate formal semantic specifications for describing web services (Yao et al., 2015)
- Lack of social-awareness among web services.

Currently, most service engineers and developers rely on manual searching (keyword-based) of service registries or other public sites such as Google Developers, ProgrammableWeb and Yahoo-pipe to discover and select require web services. Clearly, such search is ineffective and time consuming. Therefore, effective recommendation approach is required to support service consumers in selecting suitable services for a particular requirement or a given service composition task. Despite recent enhancements made to existing web service recommendation techniques, the task is many gaps in the process. A number of research work have been done on web service recommendation. We classify these works based on the implementation method and discuss them under five groups: (1) Content-based approach, which is further divided into three sub-groups (i) QoS-based (uses of Quality of Service information) (ii) Semantic based (uses semantic information similarity and formal ontology), (iii) Context-based (uses context information such as user/service locations (2) Collaborative filtering and Matric Factorisation –based, (3) Clustering-based approach, (4) Network-based approach and (5)Hybrid-based approach. We present the review of these approaches with more focus on their underlying mechanisms and limitations.

2.2.1 Content-based Approach

This group of service recommender solutions rely on syntactic and semantic information of web services and users to facilitate service recommendation process. The key idea of content-based methods is to exploit information about user's preferences and services content descriptions, which include semantic information of service interfaces, functionality descriptions, QoS values and so on. They recommend web services based on the similarity of user preferences and the descriptive information of web services (Yao et al., 2015). We further describe this group of work under three sub-sections as follow.

2.2.2 QoS-based Approach

This approach relies on the use of various information that describes nonfunctional characteristics of web services such as cost, response time, availability, reliability and throughput to recommend services to users (Su et al., 2017). Most QoS-based approaches use collaborative filtering algorithms to rank or filter out most suitable services using the QoS historical information of the user Chen et al. (X. Chen et al., 2014) proposed a QoS Aware service recommendation approach that uses a user-collaborative mechanism for collecting past QoS information of web services from different services users. Based on this data, authors employed a collaborative filtering technique to predict service QoS value, which is in turn used to rank services. Similar approach used in (Z. Zheng, Ma, Lyu & King, 2013). In (C. Yu & Huang, 2016), authors leverage the capabilities of memory-based and model-based CF algorithm to improve recommendation accuracy based on QoS information. The main limitation of this approach is that QoS properties are subjective and vary widely among different service users due to various factors such as network conditions, location, taste and so on. Hence it difficult to acquire or estimate. QoS properties are measured at the client-side, which makes it susceptible to various uncertainty (L. Chen et al., 2013). Recent research efforts in this area have been channelled towards improving QoS information reputation. (Su et al., 2017) addresses the problem of data credibility caused by dishonest users. The authors proposed a trust-aware approach for reliable personalized QoS prediction for service recommendation. The approach used on K-mean clustering algorithm to identify the honest user's cluster on each service and classify the QoS feedback submitted by each user as positive or negative feedbacks based on the cluster. Similarly, Wu et al. (C. Wu, Qiu, Zheng, Wang & Yang, 2015) proposed a credibility-aware QoS prediction approach to address unreliability of QoS data. The authors based their approach on two-phase K-mean clustering to identify the dishonest users by creating cluster values for untrustworthy index calculation in first phase and another cluster for users in second K-mean phase. Differentiating between honest and dishonest users is not a trivial task and considering the subjectivity of user's information, there is a still lot of gap in this area.

2.2.3 Context-Aware approach

This group of service recommenders employ context and location information to cluster users and services and make recommendation. Chen et al., (X. Chen et al., 2014) used location and QoS information to recommend personalize services to users. In (Fan et al., 2017), authors proposed a context-aware service recommendation approach based on temporal-spatial effectiveness. The authors model spatial correlation between user's location and the web services' location on user preference expansion, then compute their similarities. Based on this computation, services are ranked and recommended to the users. In (Xu, Yin, Deng, Xiong & Huang, 2016), authors employ context-information of both service and user to improve the performance of QoS based recommendation. For users, author employs geographical information as user-context and identify similar neighbour for each user based their similarity. The authors mapped the relationships between the similarity value and geographical distance, and for the services, affiliation information was used as context. Recommendation is made based on QoS record of user, service and the neighbours.

2.2.4 Semantic-Based approach

This group of service recommenders is based on the use of formal ontology to measure similarity for recommendation. They are usually supported by a lightweight semantic similarity assessment model that originated from ontology-based conceptual similarity (H. Xia & Yoshida, 2007). Ontology-based comparison is the backbone of semantic-based web service matching measure (W. Chen, Paik & Hung, 2015). Wang et al. (Wang, Xu, Qi & Hou, 2008) discussed various semantic matching algorithms with their defects. The ontologies are usually defined as set of semantic attributes that denote services' functionality, category, input and output parameters and so on. For instance, in Liu et al. (L. Liu et al., 2013), a semantic content-based recommendation approach was introduced.

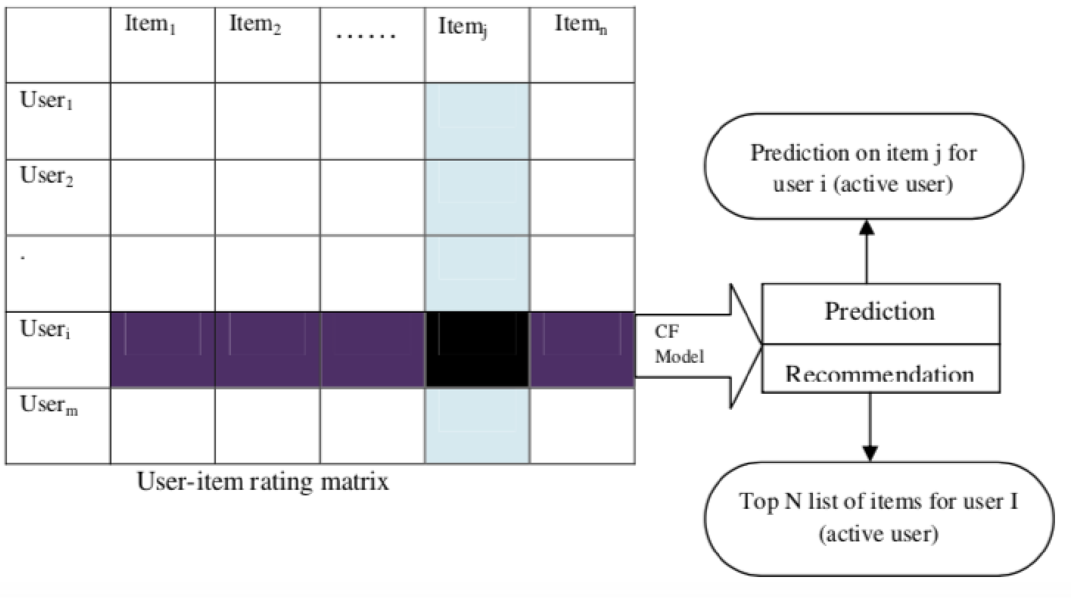
The approach estimates services similarities based on five different components: I/O, functionality, category, precondition and effect. It measures semantic similarity based on these aspects and filter services with different functionalities and categories. Lee and Kim (Lee & Kim, 2011) proposed an ontology learning method for RESTful web service, which allows web services to be grouped into concepts so as to capture relationships between words using pattern The model enables automatic generation ontologies from web application description languages (WADL). Elmeleegy et al. (Elmeleegy, Ivan, Akkiraju & Goodwin, 2008) exploited a repository of composition to estimate the popularity of a specific output, and make recommendations using conditional probability that an output will be included in a composition. The authors use a semantic matching algorithm and a meter planner to modify the composition to produce the suggested output. Building ontology to support this approach is very challenging, as it requires massive amount of expert knowledge and multiple ontology development to cater for diverse users and domains.

2.3 Collaborative Filtering Techniques

Collaborative filtering (CF) techniques are widely used in recommender systems that recommend items such as web services to users based on the similarity of different users. CF is still active and interesting research area (Bobadilla, Hernando, Ortega & Bernal, 2011; Bobadilla, Hernando, Ortega & Gutiérrez, 2012; Ekstrand, Riedl, Konstan et al., 2011; Schafer, Frankowski, Herlocker & Sen, 2007). Unlike content-based which is domain dependent, CF techniques utilize domain-independent prediction algorithms for content that cannot be adequately represented or described using descriptive data (metadata) (Isinkaye et al., 2015). Collaborative filtering algorithms are commonly used techniques in the data mining and information retrieval. They are based on using historical behaviour of past users to establish connections between users and item to recommend. The techniques work by building a database of preferences that reflect interaction between user-item. Recommendation is based on opinion of past users with similar preferences to the new users. CF matches users with related and relevant preferences by estimating the similarities between their profiles to perform recommendation (Herlocker, Konstan, Terveen & Riedl, 2004). In this setting, users rely on the neighbourhood information, and a user gets recommendations to items that he/she has not rated before but have been positively rated by other users in his/her neighbourhood.

Collaborative filtering algorithm, as one of the most mature recommendation algorithms, has been used as the basis of recommendation theory to expand in the direction of diversification. Traditional collaborative filtering method uses the preferences of similar users to generate personalized recommendation. Generally, collaborative filtering technology can be divided into two categories: memory-based and model-based methods. The former method mainly includes two classical collaborative filtering algorithms, user-based and project0based. It uses the existing scores in the whole database to predict the scores of other items to generate recommendations. This method is widely used in real-time online applications and has high efficiency. (M.Deshpande and G.Karypis 2004).

However, when the data matrix is too sparse, the efficiency and accuracy of collaborative filtering algorithm will be greatly reduced. In addition, model-based collaborative filtering algorithm uses existing scoring data to build models to generate recommendation results, singular value decomposition (B.Sarwar et al. 2000) and non-negative matrix factor decomposition in matrix factor decomposition can be effectively applied to model-based collaborative filtering algorithm. The basic theory of model-based collaborative filtering algorithm is to decompose the low-order vector according to the paired data in user item matrix. Used to generate relationships between unnoticed users and projects in the system to form recommendations.



Source: (Isinkaye et al., 2015)
Figure 2.2: Collaborative Filtering Process

Collaborative filtering recommendations can either be in form of prediction or ranked-list recommendation (Isinkaye et al., 2015). Prediction is a numerical value, R_ij, expressing the predicted score of item j for the user i, while ranked-list recommendation is a list, usually of *top-N* items, which the user will prefer most as shown in Figure 2.2.

(Bobadilla et al., 2013) and (Isinkaye et al., 2015) present a widely accepted taxonomy division used to group recommendation approaches into two broad categories, namely: (i) Model-based CF method (ii) Memory-based CF method. The groups are discussed as follow:

2.3.1 Model-Based Recommendation Methods

The model-based technique uses recommendation information such as previous user ratings to learn and/or create a model to enhance the recommendation performance of CF technique. Generally, several underlining techniques that are rooted in machine learning or data mining domain are employed in building such model. Some examples of commonly used models include Matrix factorization (Luo, Xia & Zhu, 2012) , Bayesian classifier (M.-H. Park, Hong & Cho, 2007; Friedman, Geiger & Goldszmidt, 1997), Latent features (J. Zhong & Li, 2010; Yao et al., 2015), Dimensionality Reduction techniques (Van Der Maaten, Postma & Van den Herik, 2009; Sarwar, Karypis, Konstan & Riedl, 2000) such as Singular Value Decomposition (SVD) (S. Zhang, Wang, Ford, Makedon & Pearlman, 2005),(Vozalis & Margaritis, 2007), Matrix Completion Technique, Latent Semantic methods, and Regression and Clustering and so on. Model-based techniques analyze the user-item matrix to establish relations between items (such as web services). These relations are then in turn used to compare the list of *top-N* recommendations.

2.3.2 Memory-Based Recommendation Methods

The items that were already rated by the user before play a relevant role in searching for a neighbor that shares appreciation with him (Zhao & Shang, 2010; Zhu, Ye & Gong, 2009). Once a neighbor of a user is found, different algorithms can be used to combine the preferences of neighbors to generate recommendations. Due to the effectiveness of these techniques, they have achieved widespread success in real life applications. Memory-based CF can be achieved in two ways through user-based and item-based techniques. User based collaborative filtering technique calculates similarity between users by comparing their ratings on the same item, and it then computes the predicted rating for an item by the active user as a weighted average of the ratings of the item by users similar to the active user where weights are the similarities of these users with the target item. Item-based filtering techniques compute predictions using the similarity between items and not the similarity between users. It builds a model of item similarities by retrieving all items rated by an active user from the user-item matrix, it determines how similar the retrieved items are to the target item, then it selects the k most similar items and their corresponding similarities are also determined. Prediction is made by taking a weighted average of the active users rating on the similar items k. Several types of similarity measures are used to compute similarity between item/user. The two most popular similarity measures are correlation-based and cosine-based. Pearson correlation coefficient is used to measure the extent to which two variables linearly relate with each other and is defined as (Isinkaye et al., 2015) :

$$Sim(a,u) = \frac{\sum_{i=1}^{n}(r_{a;i} - \overline{r_a})(r_{u;i} - \overline{r_u})}{\sqrt{\sum_{i=1}^{n}(r_{a;i} - \overline{r_a})^2}\sqrt{\sum_{i=1}^{n}(r_{u;i} - \overline{r_u})^2}} \tag{2.1}$$

From the above equation, $Sim(a,u)$ denotes the similarity between two users a and u, ra;i is the rating given to item i by user a, ra is the mean rating given by user a while n is the total number of items in the user-item space. Also, prediction for an item is made from the weighted combination of the selected neighbors' ratings, which is computed as the weighted deviation from the neighbors' mean. The general prediction formula is:

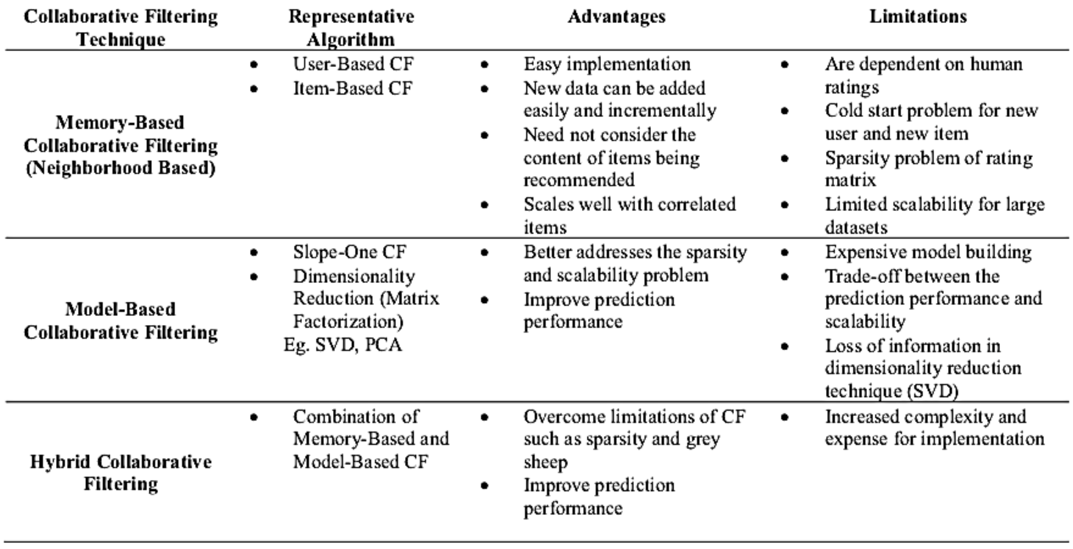
$$P(a,i) = \frac{\sum_{i=1}^{n}(r_{a;i} - \overline{r_a}) \times Sim(a,u)}{\sum_{i=1}^{n} Sim(a,u)} \tag{2.2}$$

Cosine similarity is different from Pearson-based measure in that it is a vector-space model which is based on linear algebra rather that statistical approach. It measures the similarity between two n-dimensional vectors based on the angle between them. Cosine-based measure is widely used in the fields of information retrieval and texts mining to compare two text documents, in this case, documents are represented as vectors of terms. The similarity between two items u and v can be defined as (Adomavicius & Tuzhilin, 2005; Bobadilla et al., 2013; Herlocker et al., 2004) follows:

$$CoSim(u,v) = \frac{\sum_i r_{u;i} r_{v;i}}{\sqrt{\sum_i {r^2}_{u;i}} \times \sqrt{\sum_i {r^2}_{v;i}}} \tag{2.3}$$

Similarity measure is also referred to as similarity metric, and they are methods used to calculate the scores that express how similar users or items are to each other. These scores can then be used as the foundation of user- or item-based recommendation generation. Depending on the context of use, similarity metrics can also be referred to as correlation metrics or distance metrics (Adomavicius & Tuzhilin, 2005).

Table 2.3 provides summary of different CF techniques used in building recommendation solutions as discussed in (Isinkaye et al., 2015) .



Collaborative Filtering Technique	Representative Algorithm	Advantages	Limitations
Memory-Based Collaborative Filtering (Neighborhood Based)	• User-Based CF • Item-Based CF	• Easy implementation • New data can be added easily and incrementally • Need not consider the content of items being recommended • Scales well with correlated items	• Are dependent on human ratings • Cold start problem for new user and new item • Sparsity problem of rating matrix • Limited scalability for large datasets
Model-Based Collaborative Filtering	• Slope-One CF • Dimensionality Reduction (Matrix Factorization) Eg. SVD, PCA	• Better addresses the sparsity and scalability problem • Improve prediction performance	• Expensive model building • Trade-off between the prediction performance and scalability • Loss of information in dimensionality reduction technique (SVD)
Hybrid Collaborative Filtering	• Combination of Memory-Based and Model-Based CF	• Overcome limitations of CF such as sparsity and grey sheep • Improve prediction performance	• Increased complexity and expense for implementation

Figure 2.3: Summary of Common Collaborative Filtering Techniques

2.3.3 Challenges of collaborative filtering techniques

As discussed in previous section, collaborative filtering approach exhibit some unique advantages over content-based approaches in providing recommendation solution, especially when dealing with domains with where there is few content information about the items in consideration and when it is challenging to analyze the content. Collaborative filtering technique also can recommend items which are relevant to the user without the content being available in the user's description (Schafer et al., 2007). However, with all this success and advantages, recent widespread use of collaborative technique has uncovered some potential challenges. The challenges are discussed as follows:

- **Cold-start problem :** The cold-start problem (Adomavicius & Tuzhilin, 2005; Schafer et al., 2007) occurs when there is no adequate information (e.g. ratings) about a user or an item to make reliable recommendations. Cold-start is one of the major issues that limit the performance of CF-based recommendation solutions. There are three kinds of Cold-start problem (Bobadilla et al., 2013): (i) New-user problem (ii) New-item problem (iii) New-community Problem. For *new-user problem* (Rashid, Karypis & Riedl, 2008), the challenge here is similar to content-based approaches. For the User-based CF recommender system to make accurate recommendations, the system needs to learn the user's preferences from ratings provided by the user. Since new users in the system lack ratings yet, the users cannot get any personalized recommendations with memory-based collaborative filtering.

  New-item problem occurs due to lack of ratings for new items. Initially, new items do not normally have rating, so, until the items are rated by a considerable number of item users, they are not likely to be recommended to users. Therefore, most new items that lack rating become isolated and unnoticed by item consumers. If such item can be discovered via other means, then the new-item issue would have less impact. *New-community problem* (Lam, Vu, Le & Duong, 2008; Schein, Popescul, Ungar & Pennock, 2002) is common challenge that occurs when newly starting up recommendation system without sufficient information or data for making an efficient and reliable prediction.

- **Data Sparsity problem :** Sparsity problem occurs due to insufficient recommendation information. When there are few numbers of ratings available for recommendation process compare to the number of rating sufficient for a reliable recommendation (D. H. Park, Kim, Choi & Kim, 2012; Burke, 2007). The lack sufficient information usually result to a *sparse user-item matrix* making it difficult to locate successful neighbour and eventually results to inaccurate or weak recommendation performance. For instant, in a Web service recommendation solution, there are many Web services that have been invoked, consumed and ranked by very few service consumer (mashups), hence they are less popular and they would be recommended very rarely. Data sparsity can also result to coverage problem (Isinkaye et al., 2015).

- **Scalability problem :** Scalability problem is another challenge associated to recommendation systems due to linear growth of users and items number (D. H. Park et al., 2012). For recommendation techniques that are effective and efficient mainly when limited number of data or information are used, such techniques may not be able to adapt to growth in volume of dataset. The techniques may no longer be efficient when the volume of the data increases. Therefore, it is important to employ recommendation algorithms that can adapt to change in data volume, and can effectively scale up as the volume of recommendation data increases.

2.3.4 Common Solutions to collaborative filtering challenges

In general, the main challenges of collaborative filtering algorithms include the *cold-start issues*, *sparsity* of rating matrix and the ever-growing nature of rating data (*scalability*). These challenges are usually tackled with various Matrix Factorization (MF) models such as Probabilistic Matrix Factorization (PMF), Single value decomposition (SVD) and Principal Component Analysis (PCA) (Bokde et al., 2015). Most of these models are based on latent factor models (Koren, Bell & Volinsky, 2009). The basic form of MF characterizes both users and items by vectors of factors deduced from item rating pattern. High correlation between item and user factors results to a recommendation. PCA and SVD techniques are more suitable for identifying latent semantic factors in information retrieval, especially when dealing with CF challenges. More details of various matrix decomposition methods are further discussed in Section 2.4

2.3.5 Collaborative Filtering Methods for Web Service Recommendation

As discussed in previous section, the main idea behind CF approach is to recommend new item of interest (such as Web service) to a consumer based on the experiences of other consumers of the same item. Most collaborative filtering-based Web service recommendations are either based on model or memory-based methods. Over the years, memory-based methods have been very popular partly. This is because intuitively their recommendation result is easier to interpret (Goldberg, Nichols, Oki & Terry, 1992). Collaborative filtering approach has been used in Web service recommendation especially in QoS-based Web service recommendations (Z. Zheng et al., 2013; Shao et al., 2007; J. Cao, Wu, Wang & Zhuang, 2013). In QoS-based collaborative filtering recommender system, the similarity between users and services is estimated and missing QoS values are predicted based on the historical QoS records of similar users or similar services. In (Z. Zheng, Ma, Lyu & King, 2010), a collaborative filtering approach for predicting Quality of Service values of Web services and recommending Web services was introduced. (Shao et al., 2007) proposed a user-centric Collaborative filtering approach that employs Pearson Correlation Coefficient (PCC) to calculate the correlation between users with respect to the historical Web service usage information of the users.

In web service, CF-based recommender systems recommend services to a user based on historical preferences of past users who share similar service taste. Most traditional services recommender systems ranked services based on services' QoS values. Such recommendation approach requires explicit specification of users' requirements to be able to recommend appropriate services (Yao, Wang et al., 2018). On the other hand, CF algorithms are capable of capturing users' implicit requirements. In (Z. Zheng et al., 2010), authors used a combination of user-based and item-based CF approach to improve QoS value prediction used for their recommendation system. They estimated similarity between services and users using Pearson correlation coefficient algorithm, predicted missing values in the service user matrix and recommend top-k services to users. Hu et al. (Hu, Peng, Hu & Yang, 2015) proposed QoS prediction approach based on temporal dynamics of QoS attributes and personalized factors of service consumers. The authors combined improve time series forecasting method with collaborative filtering to compensate for shortcomings of ARIMA models. Authors in (Zhou, Wang, Guo & Pan, 2015) used collaborative filtering for making web service recommendation by exploiting past usage experiences of service consumers.

2.3.6 Clustering-Based Recommendation Approach

Service clustering technique is a recent approach used to improve the quality of service discovery and support service recommendation solutions. The method enables creation of web services clusters with similar functionalities in order to reduce service's searching space during service discovery and recommendation. Generally, this technique uses service documents including tags as the main information sources for clustering (L. Chen, Yu, Philip & Wu, 2015). Existing methods focus on two aspects: (i) some methods first analyze user's service requirements and the service description documents, and then create web services clusters based on their functionality similarity (Platzer, Rosenberg & Dustdar, 2009; Sun & Jiang, 2008) (ii) Others utilize tags contributed by users and perform clustering by introducing the similarity of service document and tag information (B. Cao et al., 2016).

In order to optimize discovery and recommendation solutions, a number of existing works utilize clustering approach. (L. Chen et al., 2013) proposed a clustering method called WTLDA that utilizes both web service description documents and tags to cluster web services. The method enables seamless integration of tags and WSDL documents through LDA. However, the clustering accuracy of such method depends on the number of terms and tags used for similarity measurement. Xia et al. (B. Xia et al., 2015) proposed a category-aware web services clustering and recommendation method, which supports automatic composition development. The authors use K-means variant method based on topic Latent Dirichlet Allocation (LDA) to enhance service categorization and develop on top of the Kmean variant a category relevance-ranking model, which combines CF and machine learning. The ranking model is used to decompose composition requirements and explicitly predict relevant service categories. There results show improvement in prediction rate. However, authors only consider independent composition services for clustering without considering the interactions among them. Recently, researchers have introduced several methods into existing service clustering approaches to improve their accuracy and diversity. Common methods used are Factor analysis, topic modelling (Blei, Ng & Jordan, 2003), mining Latent Factors and MF (Z. Zheng et al., 2013). (Gao, Chen, Wu & Gao, 2015) proposed a method for ranking web services by categorize existing service composition into functionally similar clusters and then deploy manifold ranking algorithm on each cluster to recommend services. The author used Text frequency method (TF-IDF) to estimate the similarities among the service documents. However, latent semantic correlation behind the terms of service document was not considered in the work.(B. Cao et al., 2017) developed a two-level topic model using the relationships among composition services to extract the latent topics for improved service clustering accuracy. Based on the clustering result, the authors employ CF algorithm to recommend web services for composition development.

2.3.7 Hybrid Service Recommendation

Generally, hybrid recommendation technique (Burke, 2002; B. Cao et al., 2017; Porcel, Tejeda-Lorente, Martínez & Herrera-Viedma, 2012) combines two or more different recommendation techniques to enhance the recommendation quality and performance, and gain better system optimization to avoid some drawbacks of pure, traditional recommendation techniques (Adomavicius & Zhang, 2012). The ideal is to leverage the advantages of individual technique to gain better recommendation performance as the disadvantages of one technique could be minimized or removed by another techniques. Using hybrid recommendation approach could suppress or limit the weaknesses of an individual method in an integrated recommendation model. In most cases, collaborative filtering approach is integrated with some other techniques in an effort to avoid ramp-up problem (Burke, 2002). Table 2.1 show the summary of different hybridization methods commonly used in building recommendation systems. Various ways in which the integration or combination of pure recommendation techniques can be realized are discussed as follows:

Switching Hybridization

In a switching hybridization, recommender system uses some strategies and criterion to swap or switch between recommendation techniques according to certain heuristic that reflects the ability of the system to produce an effective rating (Isinkaye et al., 2015). By swapping techniques, switching hybridization can avoid inadequacy specific to a particular technique. For instant, the cold start problem in related to content-based recommendation techniques can be solved by switching to a collaborative recommendation system. The main advantage of this hybridization strategy is that the recommender

Hybridization Combination methods	
Switching	The recommended system swaps from one pure recommendation technique to another recommendation techniques depending on the current situation.
Mixed	Various pure recommendation technique are combined such that recommendation results of these techniques are presented at thesame time instead of having a single recommendation per item.
Cascade	One recommender renes the recommendations given by another.
Feature augmentation	Output from one technique is used as an input feature to another.
Feature combination	Output from one technique is used as an input feature to another.
Weighted	The scores (or votes) of several recommendation techniques are combined together to produce a single recommendation.

Table 2.1: Hybridization methods in Recommendation

system is very sensitive and responsive to the strengths and deficiencies of its component recommender techniques. However, the strategy also suffers from complexity associated with the recommendation processes, which is due to switching procedure and criterion (Isinkaye et al., 2015). The switching criterion normally leads to more complexity in the recommendation processes due to the increasing number of parameters, which has to be determined by the recommender system (Burke, 2002). A popular example of switching hybrid recommender system called *DailyLearner* is discussed in (Billsus & Pazzani, 1999). *DailyLearner* employed both content-based and collaborative hybrid. In a scenario where content based technique cannot make recommendations due to cold start problem or lack of sufficient information, content-based recommendation is used first and then followed by collaborative filtering approach.

Mixed Hybridization

In mixed hybrid approach, different pure recommendation technique is combined such that recommendation results of these techniques are presented at the same time instead of having a single recommendation per item. In this case, each item has multiple recommendation results from different techniques. Therefore, individual performances do not always have impact on the general performance. Mixed hybrid approach is usually employed where it is practical to make large number of recommendations simultaneously, An example of mixed hybridization method was discussed in (Smyth & Cotter, 2000).

Cascade Hybridization

In cascade hybrid recommendation method, one recommendation technique is first used to produce a coarse ranked list of recommendation candidates, then another technique is employed to refine the candidate set (Burke, 2002). This strategy is considered to be very effective and efficient due to the coarse-to-finer nature of the iteration. Cascade hybrid recommendation method is by nature tolerant to noise since the recommendation made by high-priority technique can only be refined by another not overturned. An example of cascade hybrid recommender system called *EntreeC* was discussed in (Burke, 2002).

Feature-augmentation and Feature-Combination Hybridization

In feature-augmentation, one technique incorporates the ratings and other information produced by another technique into its recommendation processing. For feature-combination, the features produced by a specific recommendation technique are given to another recommendation technique (Isinkaye et al., 2015). For instance, the classification or rating of similar users that is a feature of collaborative filtering is employed in a case-based reasoning recommender system as one of the features to determine the correlation between items.

2.3.8 Hybrid Web Service Recommendation

In web service recommendation, various hybridization recommendation approaches have been proposed. In general, hybrid service recommendation involves combination of collaborative filtering approaches with content-based methods to recommend services. In (Kang, Tang, Liu, Liu & Cao, 2016), authors proposed a hybrid recommendation solution which incorporated user's potential QoS preferences, functional interest on web services, and diversity feature of user interests to recommend top-ranked diversified Web services. (Y. Zhong, Fan, Huang, Tan & Zhang, 2015) leveraged several techniques to enhance recommendation performance. The author's combined web service evolution, collaborative filtering and content-based matching to build a time-aware service recommendation. Cao et al. (B. Cao et al., 2017) proposed a service recommendation technique that leverages both the content of web services repository and social network. The authors developed a two-level topic model by using social relationship among service compositions to mine the latent topics for improving service clustering accuracy. Based on the clustering results of the compositions, the authors employ collaborative filtering based service recommendation algorithm to rank services. Yao et al. (Yao et al., 2015) introduced a novel hybrid web service recommendation method based on combination of collaborative filtering and content-based approaches. The author considers simultaneously QoS rating of data and semantic content web service information such as functionality using a probabilistic generative model.

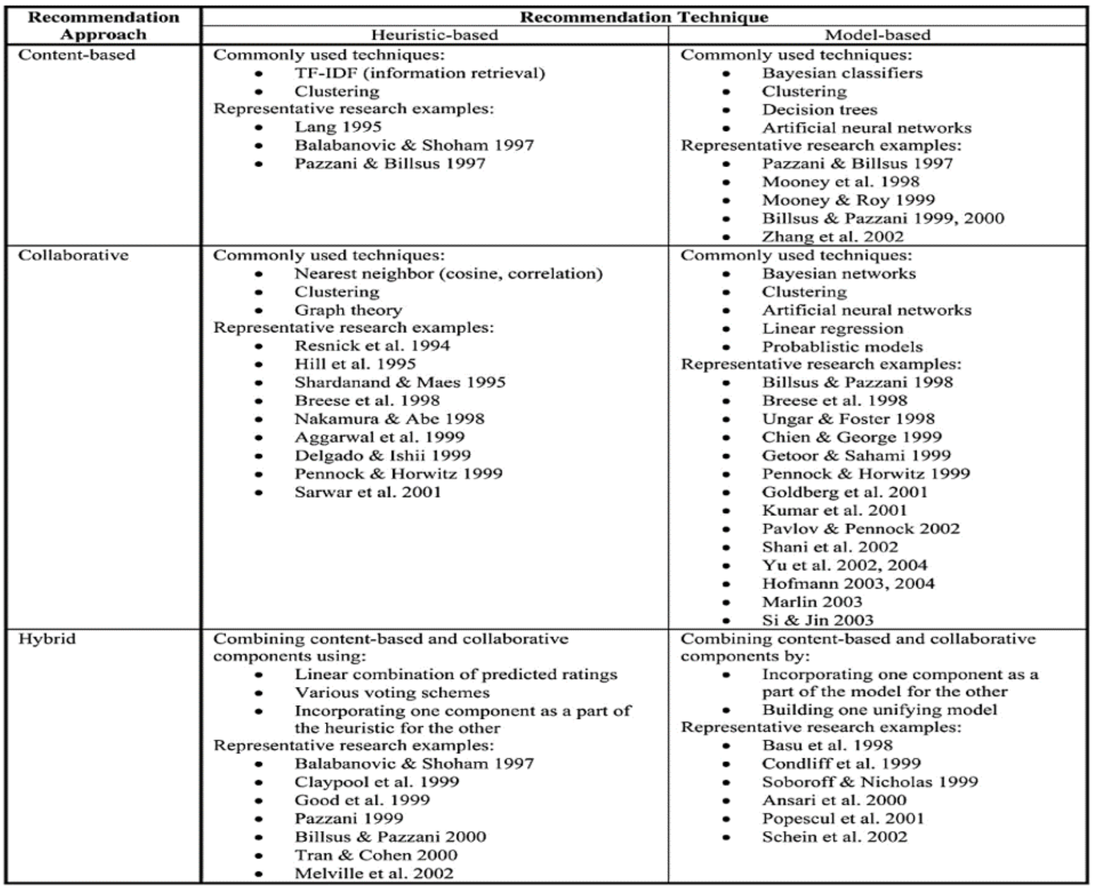
Liu.D.R proposed a new hybrid recommend method which combines segmentation-based sequence rules with KNN-CF. Firstly, the user's past purchase preferences can be analyzed to extract the corresponding sequence rules. According to the rules, the initial recommendation can be generated for users with purchase behavior in the user group. Then, the second step recommendation can be generated based on the current purchase data of the target user by the segmentation-based algorithm. Finally, the final result can be obtained by combining the two.

Albadvi.A (2009) study the hybrid recommendation algorithm for online retail stores, which improve six steps of the recommendation process: product structure classification, subdivision classification, user file establishment, attribute classification, user-to-user and user-to-product similarity calculation, product development, and finally generates recommendation. Experiments show that the improvement of the above steps has better recommendation effect than other collaborative filtering algorithms. Chang.C.C (2009) [70] use neural network nodes to propose a hybrid recommendation algorithm. By training artificial neural network, the same group of users are clustered in different kinds, and Karnaugh model is used for different clusters to discover users' needs. This method can be used in websites where information is often overloaded, such as travel recommendation websites.

2.4 Matrix Factorization

As discussed in section 2.3.3, various challenges attributed to collaborative filtering methods such as sparsity of rating matrix and scalability of recommendation data can be tackle by Matrix Factorization (MF) . This section presents an overview of MF and also discuss various MF models such Principal Component Analysis (PCA), Probabilistic Matrix Factorization (PMF) and Singular Value Decomposition (SVD).

Matrix factorization methods have received significant exposure in recent years, especially as an unsupervised learning approach for dimensionality reduction, latent variable decomposition, and for realizations of various latent factor models (Bokde et al., 2015; Koren et al., 2009). MF methods have been used majorly in data-mining, information retrieval and machine-learning domains. In most cases, MF models are usually realized based on the latent factor model. MF characterizes users and items using the vectors of factors extracted from item rating patterns. MF methods that are



Recommendation Approach	Recommendation Technique	
	Heuristic-based	Model-based
Content-based	Commonly used techniques: • TF-IDF (information retrieval) • Clustering Representative research examples: • Lang 1995 • Balabanovic & Shoham 1997 • Pazzani & Billsus 1997	Commonly used techniques: • Bayesian classifiers • Clustering • Decision trees • Artificial neural networks Representative research examples: • Pazzani & Billsus 1997 • Mooney et al. 1998 • Mooney & Roy 1999 • Billsus & Pazzani 1999, 2000 • Zhang et al. 2002
Collaborative	Commonly used techniques: • Nearest neighbor (cosine, correlation) • Clustering • Graph theory Representative research examples: • Resnick et al. 1994 • Hill et al. 1995 • Shardanand & Maes 1995 • Breese et al. 1998 • Nakamura & Abe 1998 • Aggarwal et al. 1999 • Delgado & Ishii 1999 • Pennock & Horwitz 1999 • Sarwar et al. 2001	Commonly used techniques: • Bayesian networks • Clustering • Artificial neural networks • Linear regression • Probablistic models Representative research examples: • Billsus & Pazzani 1998 • Breese et al. 1998 • Ungar & Foster 1998 • Chien & George 1999 • Getoor & Sahami 1999 • Pennock & Horwitz 1999 • Goldberg et al. 2001 • Kumar et al. 2001 • Pavlov & Pennock 2002 • Shani et al. 2002 • Yu et al. 2002, 2004 • Hofmann 2003, 2004 • Marlin 2003 • Si & Jin 2003
Hybrid	Combining content-based and collaborative components using: • Linear combination of predicted ratings • Various voting schemes • Incorporating one component as a part of the heuristic for the other Representative research examples: • Balabanovic & Shoham 1997 • Claypool et al. 1999 • Good et al. 1999 • Pazzani 1999 • Billsus & Pazzani 2000 • Tran & Cohen 2000 • Melville et al. 2002	Combining content-based and collaborative components by: • Incorporating one component as a part of the model for the other • Building one unifying model Representative research examples: • Basu et al. 1998 • Condliff et al. 1999 • Soboroff & Nicholas 1999 • Ansari et al. 2000 • Popescul et al. 2001 • Schein et al. 2002

Figure 2.4: Summary Classification of Recommender Systems Research

based on latent factor model usually model recommendation ratings as dot product of item factor matrix and user factor rating matrix. MF model map the factors to a merged latent factor space of dimensionality f, where user-item relationships are realized as inner products in the dimensional space. High similarity between each item i and user u results to a recommendation (Koren et al., 2009). Each user u is map to a vector $p_u \in \mathbb{R}^f$ and likewise each *item* i is associated with a vector $q_i \in \mathbb{R}^f$. For a given item i, the elements of q_i estimate the degree to which the item possesses the factors, positive or negative. For a given user u, the elements of p_u measure the extent of interest the user has in items that are high on the corresponding factors, again, positive or negative.The result of the dot product is ${q_i}^T p_u$. The captures the interconnection between user u and item i, the user's universal interest in the item's properties. This estimate fairly accurately the user $u's$ rating of item i, which is denoted by r_{ui}, resulting to the estimate:

$$r_{ui} = {q_i}^T p_u \tag{2.4}$$

The main task here is estimating the mapping of each item and users to factor vectors q_i,$p_u \in \mathbb{R}^f$. After completion of the mapping, the RS can easily compute the rating a particular user will allocate to a particular item using equation 2.4. This method is considered effective performance-wise in reducing problems related to high-level sparsity in recommender system databases. Some recent research efforts employ dimensionality reduction techniques to tackle similar problem. MF methods have also proven very efficient and flexible in handling large recommender system databases and enhancing scalability. Another major advantage of MF is that it enables incorporation of additional information, especially when there is no sufficient explicit information to make recommendation. Recommender systems can use implicit information such as historical information about user behavioural patterns, browsing history, web activities etc. to infer user preferences. Implicit information or feedbacks usually represent the presence or absence of events and are normally represented as a densely filled matrix.

2.5 Matrix Decomposition Models

Generally, recommendation tasks and many other real-world tasks are usually represented with initial high-dimensional matrices that required *decomposition* or *factorization* into two or more smaller matrices. The resulting matrices from the decomposition (factors of the initial matrix) have several advantages due smaller dimensions. For instance, the smaller dimension would result to reduced processing time and minimize the amount of memory requirement needed for storing the matrices. Hence, improving the overall computational efficiency of the algorithms that would have performed less on the initial matrix.

This section presents different methods that could be used to deal with high-dimensional original dataset, and matrix decomposition models including the popular decomposition models such as Eigenvalue decomposition, Singular Value Decomposition (SVD), Principal Component Analysis (PCA), Latent Semantic Indexing (LSI), Probabilistic Matrix Factorization (PMF), Non-Negative Matrix Factorization (NMF). Each of this model can be reformulated as an optimization problem with a *loss function* and data-dependent constraints.

2.5.1 Eigenvalue Decomposition Method

Eigenvalue matrix decomposition approach focuses on decomposing initial matrix into a *canonical form*. This is usually done in linear algebra, where a matrix A is factored into a canonical form, such that matrix A'^s *eigenvalues* and *eigenvectors* are used to represent the matrix. Eigenvalue matrix decomposition method is only applicable for *square* and *diagonalizable* initial matrix (Bensmail & Celeux, 1996; Golub & Van Loan, 2012; Watkins, 2004; Moler & Stewart, 1973).

Given matrix A, which is a square and diagonalizable matrix, and a constant λ and a column vector e that is non-zero and with same number of rows as the given matrix A . Then, column vector e is an eigenvector of matrix A, while λ is the corresponding eigenvalue provided:

$$Ae = \lambda e \tag{2.5}$$

If matrix A is with rank r , r nonzero eigenvalues can be grouped into an $r \times r$ diagonal matrix Λ and the corresponding eigenvectors in an $n \times r$ matrix E. Then, the resulting equation is :

$$AE = E\Lambda \tag{2.6}$$

If rank r of the matrix A is equal to its dimension n, then, matrix A can be decomposed as follows:

$$A = E\Lambda E^{-1} \tag{2.7}$$

Equation 2.7 diagonalization is very similar to Singular Value Decomposition (discussed in next section).

2.5.2 Singular Value Decomposition

In linear algebra, the Singular Value Decomposition (SVD) is an important tool that is use to solve mathematical problems including factorization of a real or complex matrix (Berry, Dumais & O'Brien, 1995; Symeonidis & Zioupos, 2016). SVD is one of the common techniques used for matrix dimension reduction, and can be considered as the generalization of the eigen-decomposition of a positive semi-definite normal matrix to any $m \times n$ through an extension of *polar decomposition*. It has many useful applications in recommender systems, statistics and signal processing. The major issue in a decomposition based on SVD is to find a lower dimensional feature space (Isinkaye et al., 2015). Formally, the SVD of an mn real or complex matrix A is a decomposed form of the :

$$SVD(A) = U\Sigma V^T \tag{2.8}$$

Where U and V are $m \times m$ and $n \times n$ real or complex unitary matrices (orthogonal matrices) respectively. Σ is an $m \times n$ rectangular diagonal matrix with non-negative elements (real numbers) on the diagonal. Matrix U w ith dimension $m \times m$ is called orthogonal if $U^T U$ is equivalent to an $m \times m$ identity matrix. The diagonal elements σ_i of Σ are called *singular values* of initial matrix A. Normally, the singular values of are ordered in descending order in Σ. The columns of matrix U and that of V are called the left-singular vectors and right-singular vectors of A, respectively.

As discussed in section 2.5.1, eigenvalue decomposition shares some similarity with SVD. If and on if matrix A is positive definite that is $A \iff A = A^T \wedge \forall e \in E, e > 0$ and symmetric, then the single value decomposition and eigenvalue decomposition are both coinciding as follows :

$$A = USU^T = E\Lambda E^{-1} \tag{2.9}$$

Where $U = E$ and $S = \Lambda$. If matrix A is a *non-square* matrix and its decomposition can be written in form $A = USV^T$, thereafter, two matrices $A_1 = M^T M$ and $A_2 = MM^T$ exist with their factorization, which can be simplified as follows (Symeonidis & Zioupos, 2016):

$A_1 = M^T M \iff$

A_1 = (USV^T)^T (USV^T) $\iff$

A_1 = (VS^TU^T) (USV^T) $\iff$

A_1 = VS^T ISV^T $\iff$

A_1 = VS^T SV^T $\iff$

$$A_1 = M^T M = VS^2V^T \tag{2.10}$$

$A_2 = MM^T$ is simplified in similar fashion:

$A_2 = MM^T \iff$

A_2 = (USV^T) (USV^T)^T $\iff$

A_2 = (USV^T) (VS^TU^T) $\iff$

A_2 = USIS^TU^T $\iff$

A_2 = USS^TU^T $\iff$

$$A_2 = MM^T = US^2U^T \tag{2.11}$$

For both A_1 and A_2 matrices , similar computations to equation 2.10 and equation 2.11 can be performed . That is , the application of SVD on the initial original matrix A can be followed to calculate matrices A_1 and A_2 SVD factorization.

To decide when to apply matrices A_1 and A_2 , minimum dimension of the matrix $A$is selected. If the dimension of matrix A is chosen to be $n \times m$ and $m << n$, then A_1 is chosen. A_2 is chosen when $n << m$.

2.5.3 Principal Component Analysis

Principal component Analysis (PCA) is also one of the common statistical techniques for data analysis and processing. It is a well-established technique for dimensionality reduction use to extract dominant patterns from high-dimensionality dataset by transforming large sets of variables into smaller one .

2.5.4 Probability Matrix Factorization

Probability Matrix Factorization (PMF) is a probabilistic model with Gaussian observation noise which scale linearly with the number of observations , and performs more efficiently on large-scale , sparse and imbalance recommendation datasets (Mnih & Salakhutdinov, 2008). Given set of item (e.g. APIs) M, and N users (like mashups) with integer rating values ranging from $1\ to\ K^1$. If Rij denotes users i rating for item j,and let $U \in R^{D \times N}$ represents the latent feature matrix for user, and $V \in R^{D \times M}$ denotes the latent feature matrix for item. The column vector U_i denoting user-specific latent feature vector and V_j represents the item-specific latent feature vector. With the rating R_{ij}, the aim of model is to find the factors of the rating matrix by minimizing the error on the test set. Therefore, the performance of the model is measured by estimating the $RMSE$ (Root mean squared error) on the test set. The first attempt is to adopt the probabilistic linear model with Gaussian noise in the data as shown in left pane of figure

2.5 (?, ?). Let I_{ij} be equal to 1 if R_{ij}- (that is user i rated item j), and I_{ij} equals 0 if otherwise. In addition, let $\mathcal{N}(x|\mu,\sigma^2) = fX(x)$ where $X \sim \mathcal{N}(\mu,\sigma^2)$. The conditional distribution of the corresponding observed ratings for user and items can be defined with as follows:

$$p(R|U,V,\sigma^2) = \prod_{i=1}^{N}\prod_{j=1}^{M}\left[\mathcal{N}(R_{ij}|U_i{}^TV_j,\sigma^2\right]^{I_{ij}} \tag{2.12}$$

Where $\mathcal{N}(x|\mu,\sigma^2)$ is the PDF (probability density function) of the Gaussian distribution with μ as the mean and variance σ^2. By placing *zero mean spherical Gaussian* priors on U and V with *hyperparameters* σ_U^2, σ_V^2: $p(U|\sigma_U^2) = \prod_{i=1}^{N}\mathcal{N}(U_i|0,\sigma_U^2\mathbf{I})$, $p(V|\sigma_V^2) = \prod_{i=1}^{M}\mathcal{N}(V_i|0,\sigma_V^2\mathbf{I})$

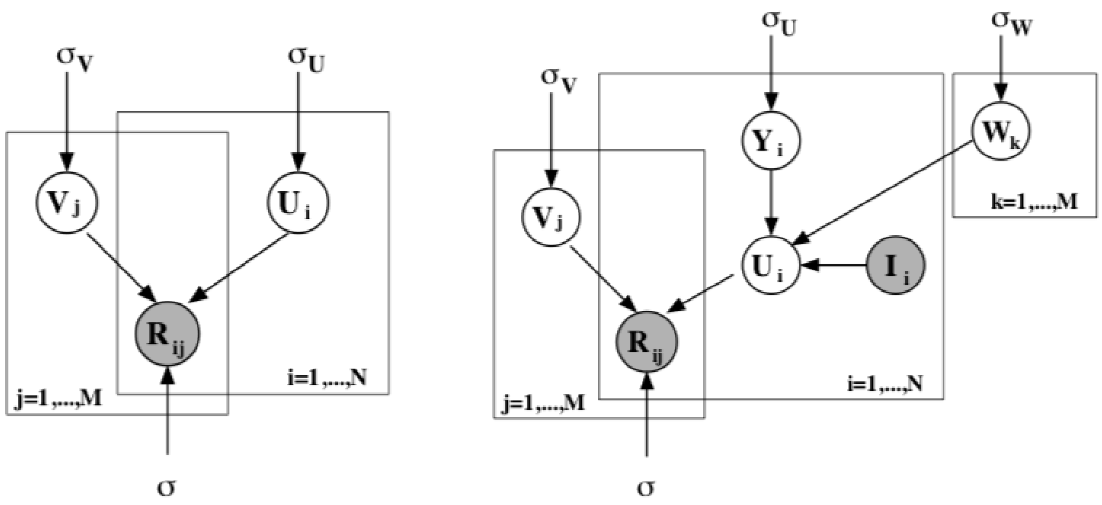
To maximize the log posterior over U and V, N definition is substituted and log is taken:

$$\begin{aligned}\ln p(U,V|R,\sigma^2,{\sigma^2}_U,{\sigma^2}_V) = &-\frac{1}{2\sigma^2}\sum_{i=1}^{N}\sum_{j=1}^{M}I_{ij}(R_{ij}-U_i{}^TV_j^2) - \frac{1}{2\sigma_U^2}\sum_{i=1}^{N}U_i{}^TU_i - \frac{1}{2\sigma_V^2}\sum_{j=1}^{M}V_j{}^TV_j \\ &-\frac{1}{2}\left(\left(\sum_{i=1}^{N}\sum_{j=1}^{M}I_{ij}\right)\ln\sigma^2 + ND\ln{\sigma^2}_U + MD\ln{\sigma^2}_V\right) + C\end{aligned} \tag{2.13}$$

where C is a constant that does not depend on the parameters. Maximizing the log-posterior over item and user features with the observation noise variance and prior variances (hyper-parameters) as constant reduces the optimization to minimization of the sum-of-squared-errors objective function with quadratic regularization terms:

$$\mathcal{E} = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}I_{ij}\left(R_{ij}-U_i^TV_j\right)^2 + \frac{\lambda_U}{2}\sum_{i=1}^{N}\parallel U_i \parallel_{Fro}^2 + \frac{\lambda_V}{2}\sum_{j=1}^{M}\parallel V_j \parallel_{Fro}^2 \tag{2.14}$$

Where $\lambda_V = \frac{\sigma^2}{{\sigma^2}_V}$, $\lambda_U = \frac{\sigma^2}{{\sigma^2}_U}$, and $||\ .\ ||_F^2$ represents the the Frobenious norm of a



Source: (Mnih & Salakhutdinov, 2008)
Figure 2.5: Left pane: Graphical Model of PMF. Right Pane: Constrained PMF

matrix. Equation 2.14 gives the local minimum of the objective function through the computation of gradient descent in U and V. The PMF model can also be considered as an a probabilistic extension of SVD model discussed in previous section (Mnih & Salakhutdinov, 2008). Constrained PMF is further discussed in (Mnih & Salakhutdinov, 2008).

2.5.5 Non-Negative Matrix Factorization

Non-Negative Matrix Factorization (NMF) is also a widely used tool for processing and analyzing high-dimensional data because it automatically extracts sparse and easily interpretable features from a set of non-negative data vectors (Gillis, 2014). NMF algorithm is one of the multivariate analysis and linear algebra algorithms which can factorize a matrix A into two matrices P and Q, with property that the three matrices do not have negative elements (Guan, Tao, Luo & Yuan, 2012). Non-negative feature enables the resulting matrices more suitable for objects clustering application (Symeonidis & Zioupos, 2016).

2.5.6 Matrix Factorization in Web service Recommendation

To achieve higher recommendation accuracy, researcher usually combine different kind of additional information using Matrix factorisation. Cao et al. (Cao, Tang, Huang, 2014) utilize content similarity between user history records and web services to recommend services for compositions using matric factorization method. In other to improve recommendation accuracy and diversity, Rahman et al. (Rahman, Liu Cao, 2017) proposed a matrix factorisation-based recommendation approach that recommend suitable web services for composition development. The authors build their work on existing solution in (Cao, Liu, Li, Liu, Tang, Zhang, Shi, 2016), which is based on integrated content and network-based service clustering. They applied MF algorithm to predict the recommendation values of missing services and then integrate the result with the popular services. Xu et al. (Xu, Cao, Hu, Wang Li, 2013), exploits the multi-dimensional social relationships among potential users, topics, service compositions and web services, which are described in a coupled matrix model. The authors designed a factorization algorithm to predict unobserved relationships in the model to facilitate effective and more accurate service recommendations. In Yao et al., (Yao, Wang, Zheng, Benatallah Huang, 2018) use a probabilistic matrix factorization method coupled with implicit correlation regularization to recommend services for composition development. The authors use latent variable model to uncover the latent correlations between services by analyzing the co-invocation pattern. The major challenges of CF-based approaches are:

- They are strongly dependent on training data and predict ranking value based on the opinions of past users' rating feedbacks
- Web services have unique attributes including functional, non-functional and social influence, which depends on and affect each other. Unlike conventional recommendation of items like movies or products in Amazon, web service behave

differently, they, interact, socialize, compete and can be composed together to create a new value-added service. CF-based approach does not capture this.

- CF-based approach is limited in terms of providing recommendation for novice consumers or developers who have vague requirements.
- Cold start problem for services and
- Loss of information in dimensionality reduction technique such PCA and SVD MF-models

The MF model and its extensions achieve good performance in recommender systems, and have been employed to predict QoS values in recent years (Zheng et al., 2013; Lo et al., 2012a; Xu et al., 2013). Zheng et al. (2013) proposed a user neighbourhood extended MF model named NIMF, which identified user neighbours through similarity computation of each pair of users. The predicted value was learned from the latent factors of the target user and his (or her) similar neighbours. Lo et al. (2012a) selected similar neighbours for each user and each service respectively through similarity calculation. They constructed two regularization terms, which tried to minimize the difference between latent factors of the target service (or user) and the neighbours. Finally, they built a model named EMF_F. Apparently, due to the similarity computation, the time complexity of the two models $(NIMFandEMF_F)$ is also quadratic to the data size. Xu et al. (2013) proposed a location-based model. For each user, their model selected the similar neighbours according to the geographic distance. The similarity between each user and each his neighbour was computed based on the distance with a predefined function. The predicted value was learned by both the latent factors of the target user and his neighbours. Although this model achieves good 6 Y. Xu, J. Yin and Y. Li prediction accuracy, it suffers from bad efficiency when the data size is large. Also, it is hard to define a suitable similarity function to measure the similarity according to the geographic information. Recommender systems mainly base their suggestions on rating data of two entities (users and items), which are often placed in a matrix with one representing users and the other representing items of interest. For example, Netflix collects ratings for movies using the five-star selection schema, and TiVo users indicate their preferences for TV shows by pressing thumbs-up and thumbs-down buttons. These ratings are given explicitly by users creating a sparse user–item rating matrix, because an individual user is likely to rate only a small fraction of the items that belong to the item set. Another challenging issue with this user–item rating matrix is scalability of data (i.e., the large number of possible registered users or inserted items), which may affect the time performance of a recommendation algorithm. We can deal with all aforementioned challenges by applying matrix decomposition methods (also known as factorization methods). Matrix factorization denotes a process, where a matrix is factorized into a product of matrices. A matrix factorization method is useful for solving plenty of problems, both analytical and numerical an example of a numerical problem is the solution of linear equations and eigenvalue problems. Its importance relies on the exploitation of latent associations that exist in the data among participating entities (e.g., between users and items). In a trivial form, the matrix factorization method uses two matrices, which hold the information of correlation between the user-feature and item-feature factors, respectively.

2.6 Tensors Decomposition Techniques

This section provides the preliminary knowledge of *Tensors*, which is one of the key techniques used in this research work. First, an overview of tensors factorization is presented and then various related tensors decomposition methods *Tuckers Decomposition*(TD) – as the underlying tensor decomposition technique for *Higher Order SVD* (HOSVD), *PARAllel FACtor analysis* (PARAFAC), *Low-Order Tensor Decomposition*

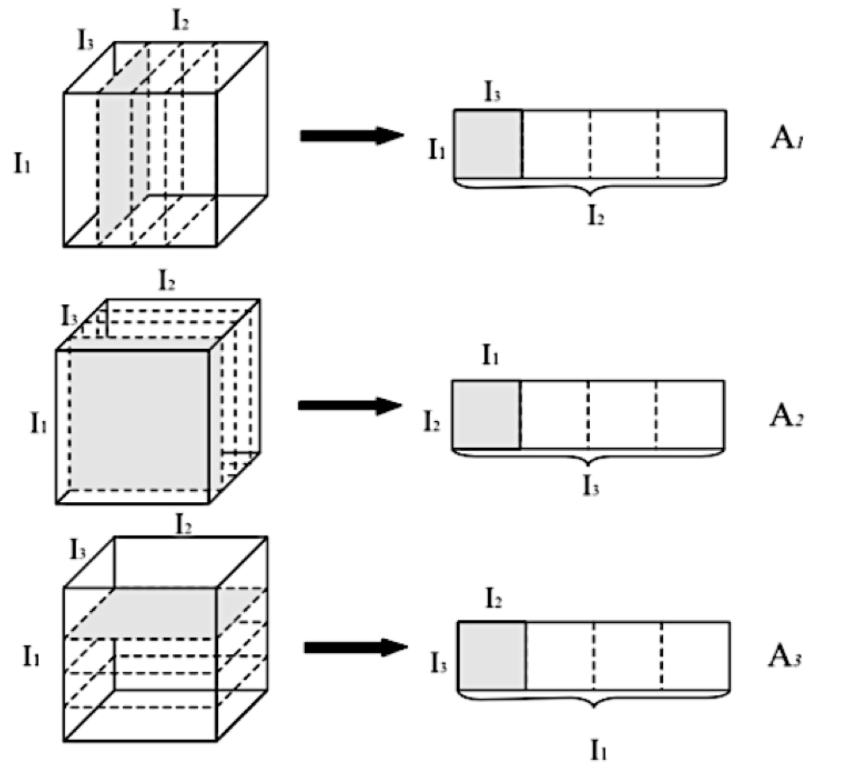
(LOTD) and *Pairwise Interaction Tensor Decomposition* methods are discussed.

2.6.1 Tensors Overview

Generally, in mathematics, geometric objects, which maps in a multi-linear manner geometric vectors, scalars, and other tensors to a resulting tensor is called *Tensor*. Vectors and scalars are the simplest form of *Tensors*. More formally, *Tensors* are multidimensional matrices. For instance, an $N - order$ tensor A is represented as $A \in \mathbb{R}^{I_1 \dots I_N}$, with elements $ai_1, ..., i_N$. Tensor factorization can be considered as the generalized form of MF, which enables flexibility in processing and integrating multi-dimension data by modelling the data as an *N-dimensional* tensor instead of the conventional 2D matrix approach, thereby enabling inclusion of any number of variables into recommendation solution (Karatzoglou, Amatriain, Baltrunas & Oliver, 2010). This approach could be employed to hybridize content and collaborative filtering approaches.

2.6.2 Tucker Decomposition And Higher Order Singular Value Decomposition

Tucker decomposition (Tucker, 1966) has few variants. Higher order singular value decomposition (HOSVD) is a specific variant of Tucker decomposition that factorizes a tensor into a set of matrices with one small core tensor. Often regarded as Tucker I decomposition (Symeonidis & Zioupos, 2016; De Lathauwer, De Moor & Vandewalle, 2000) . In order to use HOSVD technique on a 3rd-order tensor A, three *matrix unfolding* functions that are the matrix representations of tensor A having all column (row,...) vectors stacked over each other successively is constructed. Figure 2.6 illustrates a typical unfolding of a 3-order tensor A, where A_1, A_2, A_3 are known as *mode-1*, *mode-2*, *mode-3* matrix unfolding of A respectively.



Source: (Symeonidis & Zioupos, 2016)
Figure 2.6: The unfolding of 3rd-order tensor A, in the three modes

$$A_1 \in \mathbb{R}^{I_1 \times (I_2 I_3)}, A_2 \in \mathbb{R}^{I_2 \times (I_1 I_3)}, A_3 \in \mathbb{R}^{I_1 I_2 \times (I_3)} \tag{2.15}$$

HOSVD has been employed by various systems (Symeonidis, Papadimitriou, Manolopoulos, Senkul & Toroslu, 2011; S. Chen, Wang & Zhang, 2007; Symeonidis, 2009) especially recommender systems for tensor factorization (for generating low-rank tensor approximations) to enable exploitation of multi-dimensional relationships that exists between recommendation objects and the underlying latent semantic structure of the objects. For instance, an application of HOSVD in tensor decomposition can found in Social Tagging Systems (STSs) (Symeonidis, 2009), where the ternary interaction of users, items, and tags in the system can be captured as a third- order tensor A. These elements are represented by a 3-order tensor, on which latent semantic analysis and dimensionality reduction is performed using the HOSVD approach. Generally, STSs can recommend users with similar social interest based on common tags on similar items. Users could have diverse interests for an item and similarly, items could have multiple facets. The key motivation here is to use the data triplet (users U, items I, tags T) in the system and present the recommendation problem as a third-order tensor, which requires unfolding the initial third-order tensor A and completing its non-observed entries.

Relational structure $\mathbb{F} := (U, I, T, Y)$ is used to formally represent the STSs, where U,T and I are non-empty, finite sets. Y is the set of observed ternary relation between the triplets (U,I,T). That is, $Y \subseteq U \times I \times T$. Normally, a post has a tag assignments of a particular user for a given item , that is , for the triplet $(u, i, T_{u,i})$ where $u \in U, i \in I$, and a nonempty set $T_{u,i} := \{t \in T \mid (u, i, t) \in Y\}$. Y can be represented by the binary tensor $A = (a_u, i, t) \in \mathbb{R}^{|U| \times |I| \times |T|}$. Here digit 1 denotes observed tag assignments and digit 0 denotes missing values. The relationship is presented below:

$$a_u, i, t := \begin{cases} 1, & (u, i, t) \in Y \\ 0, & else \end{cases}$$

Hence, the tensor $\hat{A}$ is built as product of the core tensor $\hat{C}$ and the mode products of the three matrices $\hat{U}$, $\hat{I}$, and $\hat{T}$ as expressed below :

$$\hat{A} := \hat{C} \times_u \hat{U} \times_i \hat{I} \times_t \hat{T} \tag{2.16}$$

$\hat{U}$, $\hat{I}$, and $\hat{T}$ are all low-rank feature matrices denoting a mode that is user, items, and tags, respectively in terms of its small number of latent dimensions k_U , k_I , k_T , and $\hat{C} \in \mathbb{R}^{k_U \times k_I \times k_T}$ is the core tensor which governs the relation between the latent semantic factors. Model parameters to be optimized are represented by the quadruple

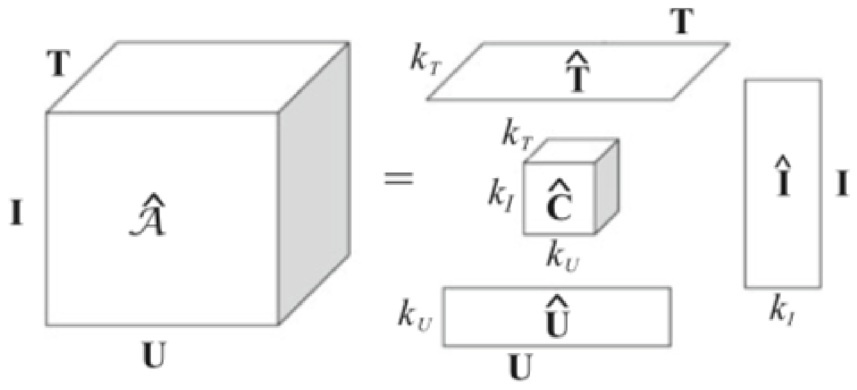


Figure 2.7: Tensor decomposition in STS of A in the three modes –

$\hat{\theta} \coloneqq (\hat{C}, \hat{U}, \hat{I}, \hat{T})$ as shown in Figure 2.7.

HOSVD algorithm fundamental idea is to reduce an element-wise loss on the components of $\hat{A}$ by optimizing the square loss. That is,

$$\arg\min_{\hat{\theta}} \sum_{(u,i,t)\in Y} (\hat{a}_{u,i,t} - a_{u,i,t})^2 \tag{2.17}$$

Following the optimization of the parameters, prediction can be realized as follows:

$$\hat{a}(u,i,t) \coloneqq \sum_{\tilde{u}=1}^{k_U} \sum_{\tilde{i}=1}^{k_I} \sum_{\tilde{t}=1}^{k_T} \hat{c}_{\tilde{u},\tilde{i},\tilde{t}} \cdot \hat{u}_{u,\tilde{u}} \cdot \hat{i}_{i,\tilde{i}} \cdot \hat{t}_{t,\tilde{t}} \tag{2.18}$$

where user $\hat{U} = [\hat{u}_{u,\tilde{u}}]^{u=1,\ldots,U}_{\tilde{u}=1,\ldots,k_U}$, $\hat{I} = [\hat{i}_{i,\tilde{i}}]^{i=1,\ldots,I}_{\tilde{i}=1,\ldots,k_I}$, $\hat{T} = [\hat{t}_{t,\tilde{t}}]^{t=1,\ldots,T}_{\tilde{t}=1,\ldots,k_T}$ and indices over the feature dimension of a feature matrix are marked with a tilde, and elements of a feature matrix are marked with hat like ($\hat{t}_{t,\tilde{t}}$) (Symeonidis & Zioupos, 2016).

2.6.3 Parallel Factor Analysis (PARAFAC)

The Parallel Factor Analysis (PARAFAC) (Bro, 1997) is a special case Tucker decomposition method (also known as *canonical decomposition*), which minimizes the complexity of tensor decomposition by assuming only a diagonal and core tensor. The

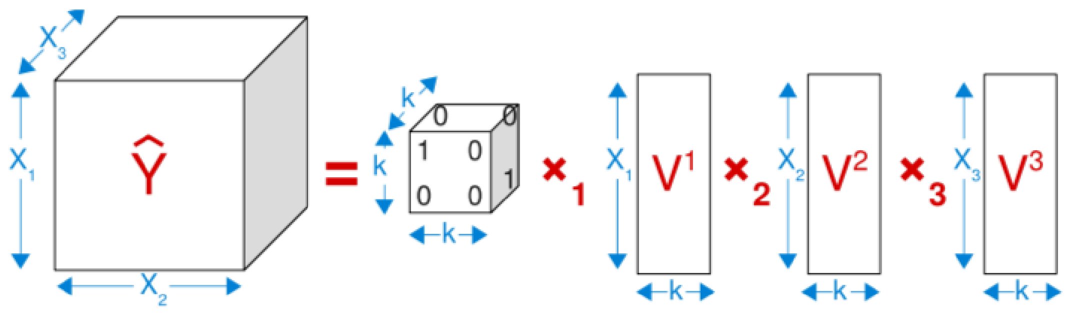


Figure 2.8: The graphical representation of PARAFAC with diagonal core tensor and the factorization dimensionality (equal for the three modes)

method can be considered as a generalized bi-linear Principal Component Analysis (PCA). In PARAFAC, a decomposition of data is made into a tri-linear components with each component consisting one score vector and two loading vectors. The score and loadings for all the components are treated equally numerically . An typical example of PARAFAC is shown in figure 2.8 for 3-modes.

PARAFAC employs similar decomposition and parametrization approach as Tuckers (as in equation 2.15. However, dimensionalities of the factor matrices in PARAFAC are thesame for all the components and the core tensor is diagonal:

$$c_{\hat{u},\hat{i},\hat{t}} \overset{!}{=} \begin{cases} 1, & if = \hat{u} = \hat{i} = \hat{t} \\ 0, & else \end{cases}$$

For instance, given a 3-way array with 3-loading matrices A, B, C for a PARAFAC model . The 3-loading matrices have elements a_{if}, b_{jf}, c_{kf} respectively. The tri-linear method results to minimizing the sum of squares of the residuals, e_{ijk} in the model.

$$x_{ijk} = \sum_{f=1}^{F} a_{ij} b_{jf} c_{kf} + e_{ijk} \tag{2.19}$$

The graphical representation of equation 2.19 for 2-compoents is shown in figure

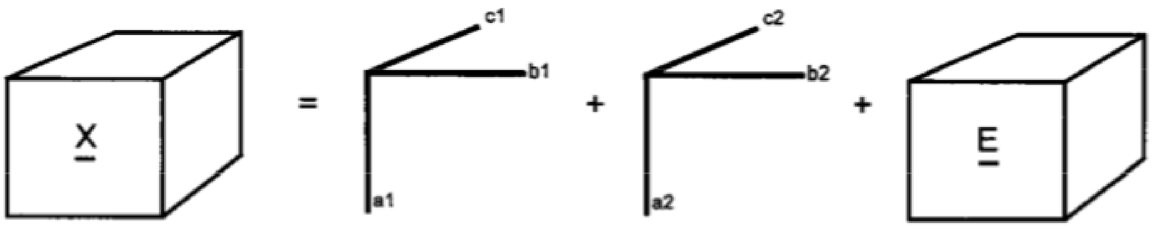


Figure 2.9: The graphical representation of 2-components PARAFAC of data array X
Source: (Bro, 1997)

2.9. The resulting model can be written as:

$$x_{ijk} = \sum_{f=1}^{F} a_f \bigotimes b_f \bigotimes c_{kf} \tag{2.20}$$

where a_f is the $f_t h$ column of loading matrix A, while b_f and c_f hold thesame values for loading matrices B and C respectively (Bro, 1997).

2.7 Pairwise Interaction Tensor Factorization

Pairwise Interaction Tensor Factorization (PITF) is a special case of Tucker decomposition model with linear runtime both for learning, recommendation and prediction. PITF is discussed in (Rendle & Schmidt-Thieme, 2010). Unlike PARAFAC that models a $m - ary$ relation directly with one $m - ary$ product, PITF explicitly models many pairwise relations (Rendle, n.d.). While Tucker decomposition and PARAFAC directly model a ternary relation, the main idea of PITF is express or model pairwise interactions.

Chapter 3

Research Method

This chapter provides the details of the construction of the proposed Tensor factorization models used for Web-API recommendation in this research, This research work adopts both exploratory and quantitative approaches to understudy the application of 3-order Matrix factorization technique in exploiting the latent feature that exists among multi-dimensional relations in Web service domain. Generally, the research method can be divided into four key components: (i) First, the construction and description of the data-set used in the research(ii) the representation and construction of regularization-based Tensors factorization optimization model for the Web-API recommendation application. (iii) the prediction algorithm based on HOSVD decomposition for the proposed Web-API recommendation framework. (iv) Comparative procedure for Traditional 2-orders Probabilistic Matrix Factorization and 3-order Tensors Factorization using Web-APIs dataset.

3.1 Overview

This research intends to explore both Tensor factorization (TF) technique and Probabilistic Matrix factorization (PMF) to tackle various issues related to the both large multi-dimensional datasets and sparse Web-APIs dataset.

Generally, typical recommender system is based on CF, which relies on historical users' information such as ratings given by past users. CF automatically predicts the interest of user based on the rating information from other similar user or items (Ma, Yang, Lyu & King, 2008). Ratings are usually explicitly given by the users resulting to a sparse user-item rating matrix. Hence, data sparsity is one of the key challenges of CF. Matrix Factorization performs well on sparse data (Symeonidis, Nanopoulos & Manolopoulos, 2008). MF method uses two matrices that hold information of the correlation between the item-features (such as Web-API features) and user-features (such as the mashup feature) factors respectively (Symeonidis & Zioupos, 2016). Another very key advantage of MF is that it enables incorporation of-of side information. For instance, in the absence of explicit feedback, RS can utilize implicit feedback to infer user preference. However, MF can only operate on two-dimensional data, which may not be applicable to real-world application that involves more than 3-entities interaction. For instant in Web-APIs recommendation, apart from Mashup-API interaction matrix, other contextual information or entities such as the Location, QoS of services, Time and Age could influence the quality of Web-API rating prediction. Recently, Tensor Factorization (TF) approach became popular for handling multi-faceted data (Huang et al., 2018; Yao, Sheng, Wang, Zhang & Qin, 2018). TF can be employed to add any number of variables to an RS solution. This work presents a generic Collaborative Filtering model that is based on extended concept of matrix factorization to that of tensor factorization to address recommendation problems. The research used raw data set, which includes the data related to mashups and Web APIs from ProgrammableWeb:com (ProgrammableWeb, 2017), since it is the largest Web API directory on the Web. Then the data is processed according to the requirement of network modelling from a network science perspective to facilitate us to investigate the relationship between mashups and Web APIs.

3.2 Data Acquisition and Processing

This research work explores publicly available dataset from *ProgrammableWeb* repository, the current largest Web-API, mashup repository. The time-stamped, raw dataset craw from the repository consists of textual descriptions of 17829 APIs and 5691 mashups, and their historical invocation from June 2005 to January 2019. Considering that the ProgrammableWeb backend database is not publicly accessible, only its web pages can be employed for collecting the data. Hence, data scraping technique is employed to crawl data from the repository web pages. After that, the web pages are apportioned into two categories: Web-APIs and mashups, Each Web-API has features including ***tags, name, description, users (as mashup), publication date, URL, end-point, portal and category***. Likewise, every mashup also has the above metadata plus the set of Web-APIs invoked. Table 3.1 describes some basic statistics of the acquired ProgrmmableWeb dataset.

Due to the fact that the ProgrammableWeb dataset does not include ratings between mashups and APIs, Mashup-Web-API invocation data is adopted has the ratings here. Then several cleaning and preprocessing follow. First, redundant APIs and mashups are removed and then obtain 5691 mashups, with only 1,170 Web-APIs included. A very sparse mashup-API mapping was achieved with density $1.6 \times 10^{-}3$. All the APIs and mashups descriptions were put into two lists respectively, and assign a tag for each description.

3.3 Mashup-Oriented MF-Based Recommendation

Two-dimensional Matrix Factorization approaches have been greatly successful in latent factor models, and it has been employed in several MF-based collaborative filtering Web-API recommendation solutions.

Data Type	**Statistics**
Number of Web APIs acquired	17,829
Number of Mashups acquired	5691
Average number of Web APIs invoked by Mashups	2.1
Number of Web APIs invoked in at least one Mashup	1,170
Number of Mashups with less than 2 services	241
Number of Mashup-API interaction	10,737
Mashup-APIs Affiliation Matrix Density	1.6×10^{-3}

Table 3.1: Statistics of the ProgrammableWeb Dataset

Web-API: ***Twillo API***		**Mashup:** ***Nostalgia***	
Attribute	**Value**	**Attribute**	**Value**
Web-API Name	Twillo	Mashup Name	Nostalgia
Category/Tags	Telephony Text-to-Speech, Voice...	Category/Tags	Music, Charts
API Portal	http://www.twilio.com	API Portals	http://www.nostal.se
Profile	*Twilio provides a simple hosted API...*		
Endpoints	Affiliation Matrix Density		

Table 3.2: Sample Specification of Web-API (a) and Mashup Profile (b) in PW dataset

Generally, MF models map both users and items to a joint latent factor space of dimensionality, such that user-item interactions are modelled as inner products in that space (Koren et al., 2009). This success and adoption can be mainly attributed to its exceptional scalability and accuracy.

For instant, suppose two entities $u\&v$, where $u = \{u_1, u_2, \dots u_m\}$ is a set of users, and $v = \{v_1, v_2, \dots v_n\}$ be the set of items; the primary idea here is to decompose a 2-dimensional user-item matrix $\mathbf{R} \in \mathbb{R}^{m \times n}$ into two low-order matrices representing user latent subspace matrix $\mathbf{U} \in \mathbb{R}^{m \times d}$ and item latent subspace matrix $\mathbf{V} \in \mathbb{R}^{n \times d}$ respectively. Dimensional shared latent space $d \ll min(m, n)$. The likelihood of a particular user u_i interacting with item v_j will be approximated through computation of the following optimization problem (Yao, Sheng et al., 2018):

$$\mathcal{L}(\mathcal{U}, \mathcal{V}) = \min_{U,V} \sum_{i=1}^{m} \sum_{j=1}^{n} (R_{ij} - U_i V_j^T)^2 \tag{3.1}$$

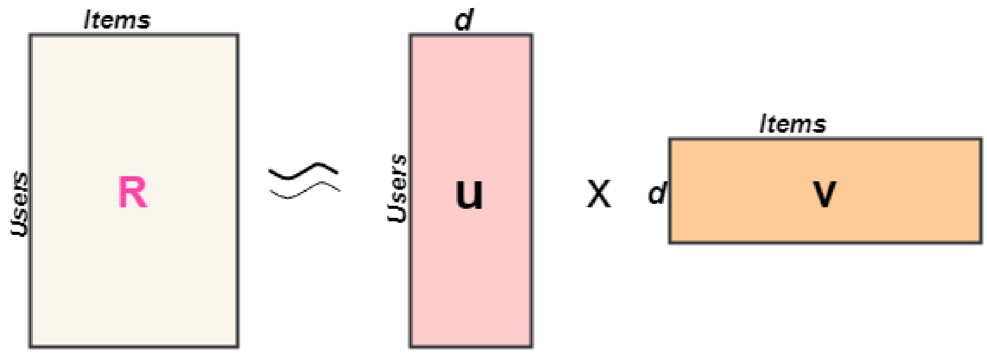


Figure 3.1: Matrix Factorization for recommendation

Equation 3.1 is a typical Probabilistic Matrix Factorization solution , which is described and simplified by authors in (Mnih & Salakhutdinov, 2008) as follows:

Since Root Mean Squared Error (RMSE) is the metric usually adopted to measure such model performance or a likelihood of such occurrence, RMSE is computed on the test set based on probabilistic linear model with Gaussian observation noise. To simplify the rating prediction problem, a conditional distribution over the observed ratings is defined as :

$$p(R|U,V,\sigma) = \prod_{i=1}^{m}\prod_{j=1}^{n}\left[\mathcal{N}(R_{ij}|U_i{}^TV_j,\sigma^2\right]^{I_{ij}} \tag{3.2}$$

Where $\mathcal{N}(x|\mu,\sigma^2)$ is the probability density function (PDF) of the Gaussian Distribution with variance σ^2 and μ as the mean. I_{ij} denotes the indicator function that is equivalent to 1 provided the user i rated or interact with item j and would be 0 if otherwise.

With *zero-mean spherical Gaussian priors* placed in range [1,11] on both the user and item feature vector, the result equation is as follow:

$$p\ (U|\sigma^2{}_U) = \prod_{i=1}^{m}\mathcal{N}(U_i|0,\sigma^2{}_U\mathbf{I}),\quad p\ (V|\sigma^2{}_V) = \prod_{j=1}^{n}\mathcal{N}(V_j|0,\sigma^2{}_V\mathbf{I}) \tag{3.3}$$

For both user U and item V latent features, the log of the *posterior distribution* over both entities is computed as follows based on elements of equation 3.3:

$$\ln p(U,V|R,\sigma^2,{\sigma^2}_U,{\sigma^2}_V) = -\frac{1}{2\sigma^2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}(R_{ij} - {U_i}^T V_j^2) - \frac{1}{2\sigma_U^2}\sum_{i=1}^{m} {U_i}^T U_i - \frac{1}{2\sigma_V^2}\sum_{j=1}^{n} {V_j}^T V_j \\ -\frac{1}{2}\left(\left(\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}\right)\ln\sigma^2 + md\ln{\sigma^2}_U + nd\ln{\sigma^2}_V\right) + C \tag{3.4}$$

C in equation 3.3 is a constant that is not parametric dependent. In order to maximize the *log posterior* over the feature of both entities' features (user,item features), while keeping the *hyper-parameters*, (that is, the observation noise variance and prior variances) fixed, the sum-of-squared-errors *Objective function* is minimized with *quadratic regularization terms* as follows:

$$E = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}\Big(R_{ij} - {U_i}^T V_j^2\Big)^2 + \frac{\lambda U}{2}\sum_{i=1}^{m} \| U_i \|_F^2 + \frac{\lambda V}{2}\sum_{j=1}^{n} \| V_j \|_F^2 \tag{3.5}$$

Where $\lambda_U = \frac{\sigma^2}{{\sigma^2}_U}$, $\lambda_V = \frac{\sigma^2}{{\sigma^2}_V}$, and $\| \ . \ \|_F^2$ represents the the Frobenious norm[1]. By performing gradient descent on both U and V in Equation 3.7, a *Local Minimum* of equation 3.7's *objective function* can be realized as in equation 3.1.

Applying the approach described above to ***mashup-oriented Web-API recommendation problem***:
Given entity M as target mashup (same as user is above scenario), how do we discover most suitable entity A – candidate Web-APIs components to compose M?

Given the invocation information of n APIs in k mashups, Let the invocation interaction between APIs and Mashups be represented as 2-dimensional matrix $\mathcal{R} \in$

[1]https://en.wikipedia.org/wiki/Matrix_normFrobenius_norm

$\mathbb{R}^{n \times k}$, where each element r_{ij} implies if or if not an API represented by a_i is consumed by a mashup (or a users) denoted by m_j (that is true if r_{ij} = 1 and false if r_{ij} = 0). The main objective of PMF in this case is to map the mashups with respect to component APIs into a shared a dimensional shared latent space $d \ll min\{n, k\}$. The resulting latent subspace matrices for mashup and Web-APIs are arranged in $k \times d$ for matrix M and $n \times d$ for matrix A respectively,where API $a_i \in \mathbb{R}^d$ and mashup $m_j \in \mathbb{R}^d$. Therefore, the probability-based prediction that a_i will be consumed by m_j is calculated by:

$$\hat{r}_{ij} = a_i^T m_j \tag{3.6}$$

If the latent factors of both mashups and Web-APIs are both represented as matrices **M** and **A**, such that **A** $\in \mathbb{R}^{n \times d}$ and **M** $\in \mathbb{R}^{k \times d}$ respectively, these factors can be learned from minimizing the sum-of-squared-errors Objective function with quadratic regularization terms as follow:

$$\mathcal{L} = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{k} I_{ij} \Big(r_{ij} - a_i^T m_j \Big)^2 + \frac{\lambda A}{2} \sum_{i=1}^{n} \| A_i \|_F^2 + \frac{\lambda M}{2} \sum_{j=1}^{k} \| M_j \|_F^2 \tag{3.7}$$

Similar to equation in 3.7, $\lambda_A = \frac{\sigma^2}{\sigma^2{}_A}$, $\lambda_M = \frac{\sigma^2}{\sigma^2{}_M}$, and $\| \, . \, \|_F^2$ represents the the Frobenious norm of a matrix. ($\frac{\lambda A}{2} \sum_{i=1}^{n} \| A_i \|_F^2 + \frac{\lambda M}{2} \sum_{j=1}^{k} \| M_j \|_F^2$) are the regularization terms introduced to reduce *over-fitting* the training data. The value of λ in this case is data-dependent.where I_{ij} equals 1 if API a_i is invoked by mashup m_j , and 0 otherwise. The aim of the optimization is to minimize the sum-of-squared-errors loss function with quadratic regularization terms, and gradient-decent approaches can be applied to find a local minimum (Yao, Sheng et al., 2018)

3.4 Tensors Factorization

For ternary relations, where standard matrix factorization cannot be applied *tensors* are used. Tensor is regarded as a multi-dimensional matrix- a generalization of a matrix, for a *order N* tensor means N-dimensional tensor.

3.4.1 Need for Tensors in Recommender Systems

Why do we need tensors in recommendation applications? Even-though conventional two-dimensional recommendation models can be used successfully in many cases, it is common to find real-world settings where auxiliary contextual information such as *location, proximity, time, taste, company of others, mood* can influence user's decision making. Therefore, such contextual information can be captured and incorporated into the recommendation algorithm to enhance the rating accuracy.

Let the set of contextual dimension of each contextual information in the recommendation space be represented as $\{C_1, C_2, \ldots, C_k\}$, where k is the total number of contextual information in the recommendation space and each dimension C_i is the set of features that captured a specific kind of context.
Suppose dimensions $U, V, C_1, C_2, \ldots, C_k$ are given, a recommendation space can be defined for these dimensions as a Cartesian product $Q = U \times V \times C_1 \times C_2 \times \cdots \times C_k$. The contextual dimensions are modelled similar to the users-items model in MF. Similarly, observing ratings on this space can be modeled as a sparse tensor. Hence, the recommendation problem is now interpreted as a Tensor Completion problem, which infers missing ratings from a partially specified tensor of observations by fitting a tensor factorization model to the data (Zou, Li, Tan & Chen, 2015).

3.4.2 Notations and Operations

An N-way Tensors is generalization of matrices or vectors represented as $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, where the elements of the N-way array are indexed by $i_n \in \{i_1, i_2, \ldots, i_N\}$ for $1 \leq n \leq N$. Various terms, concepts and procedures in linear algebra are important to understanding tensor factorization. Some key concepts adopted in this work are defined below:

Square Matrix and orthogonality

A square matrix $A \in \mathbb{R}N \times N$ is a matrix with the same number of rows N and columns N . A is called *non-singular* if there is another matrix $B \in \mathbb{R}N \times N$ such that $AB = I$ and $BA = I$, where I is an identity matrix $I \in \mathbb{R}N \times N$. If A is not invertible , then it is *singular*.

A square matrix A is called *orthogonal*, if the column vectors of A form an orthonormal sets in $\in \mathbb{R}^N$. That is, A is matrix with real numbers entries, whose columns and rows are orthogonal unit vectors. $AA^T = A^T A = I$, where A^T is the transpose of matrix A.

Frobenius Norm || . ||

The Frobenius norm [2], also known as the Euclidean norm (also used for the vector $L^2 - norm$)is matrix norm of an $M \times N$ matrix A defined as the square root of the sum of the *absolute squares* of its elements, define in equation 3.8 below:

$$\parallel A \parallel_F = \sqrt{\sum_{i=1}^{N} \sum_{j=1}^{M} |a_{ij}|^2} \tag{3.8}$$

[2]http://mathworld.wolfram.com/FrobeniusNorm.html

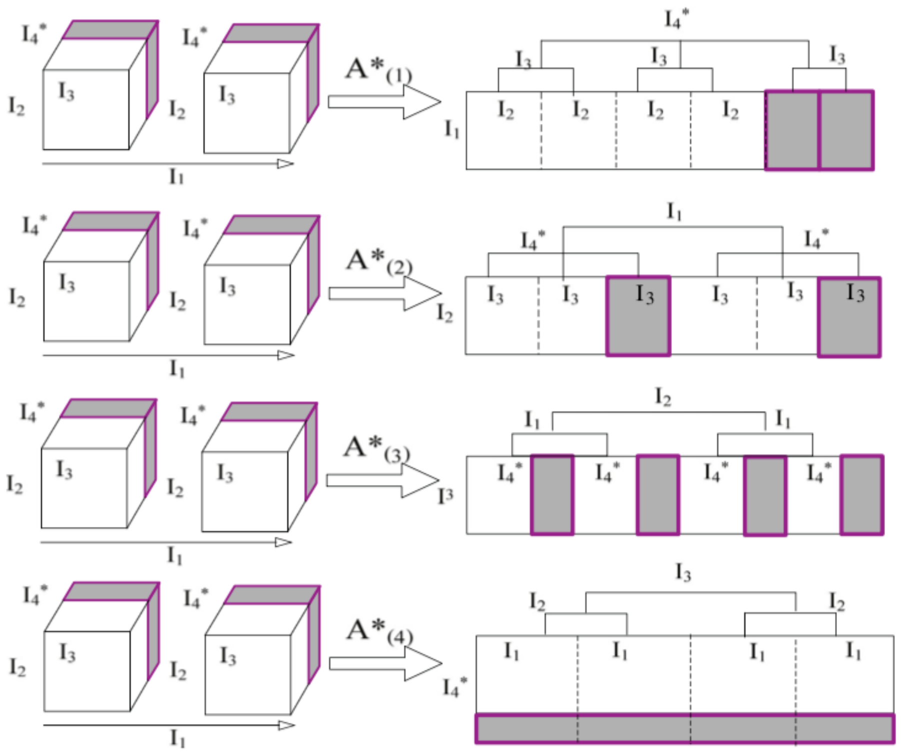


Figure 3.2: Illustration of four-order tensor Unfolding
Source: (Zou et al., 2015)

Matrix Unfolding

Matrix unfolding is simply a mapping operation from a tensor to a matrix. A tensor $\mathcal{A}$ can be $matricized$ (i.e. construct matrix representations) in which all the column (or row) vectors are stacked one after the other. A tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \ldots I_n \ldots I_N}$ which can be unfold along the n_{th} mode represented as $uf(A, n)$. The resulting matrix $\mathcal{A}_{(n)} \in \mathbb{R}^{I_1 I_2 \times \ldots I_{n-1} I_{n+1} \ldots I_N}$ has its column vectors in the $n - mode$ of vectors of tensor $\mathcal{A}$. Figure 3.2 illustrates an example of 4-order tensor of dimensions $2 \times 2 \times 2 \times 2 \times 3$ and its respective 4 unfolding matrices in different modes as presented in (Zou et al., 2015). Different color are used in the figure to describe the later incremental updates and can now be ignored.

3.5 High-Order Singular Value Decomposition on Tensors

This section describes tensor factorization method employ in this work called HOSVD, which is an extension of Singular Value Decomposition (SVD) method described in the literature review. The section will show algorithm and step-by-step implementation of HOSVD, and how the method can be employ to exploit the underlying latent semantic structure in three-dimension data model. Then an illustration of Web-API or service oriented tensor model implementation is presented with algorithms.

3.5.1 HOSVD Algorithm Description

The algorithm description presented here shows the operation of HOSVD and how it is performed based on inferred latent associations in 3-dimensional plane. Tensor decomposition technique initially build a tensor, depending on the usage data triplet u, i, t of user (mashup), item (API), and proximity. The idea is to employ the three entities that relate in a location-based context-aware recommender system. Consequently, a tensor $\mathcal{A}$ will be unfolded into three new matrices. Thereafter, SVD approach will be apply to each of the new matrix. The following 6-steps summarizes the SVD procedures (Symeonidis et al., 2008):

- *Step-1 :* The construction of the initial tensor $\mathcal{A}$ based on the integration data triplet
- *Step-2 :*The matrix unfolding of tensor $\mathcal{A}$, where three mode matrix representation of tensor $\mathcal{A}$ is constructed , resulting to the creation of three new matrices (one for each mode) as shown in equation 2.15
- *Step-3 :*After unfolding the matrix , SVD is then applied on all the three new matrices.

- *Step-4 :* Core tensor $\mathcal{S}$, which reduces the dimensionality is then constructed.

$$\mathcal{F} = S \times_1 U^1 \times_2 U^2 \tag{3.9}$$

where $U^1 = u_1^{(1)} u_2^{(1)} \dots u_{I_1}^{(1)})$ is a unitary $(I_1 \times I_1)$ matrix and $U^2 = u_1^{(2)} u_2^{(2)} \dots u_{I_1}^{(2)})$ is a unitary $(I_2 \times I_2)$ matrix. $\mathcal{S}$ is an $(I_1 \times I_1)$-matrix with the following properties:

i.) *Pseudodiagonality*: $\mathcal{S} = diag(\sigma_1, \sigma_2, \dots, \sigma_{min}\{I_1, I_2\})$ and

ii.) *Ordering*: $\sigma_1 \geq \sigma_2, \geq \dots \geq \sigma_{min}\{I_1, I_2\} \geq 0$.

- *Step-5 :* The construction of the $\hat{\mathcal{A}}$ tensor, which is an approximation of tensor $\mathcal{A}$.

$$\mathcal{A} = S \times_1 U^1 \times_2 U^2 \times_3 U^3 \tag{3.10}$$

Note that the tensor-matrix multiplication operator $\times_u$ indicates the direction on the tensor on which to multiply the matrix using the superscript.

$$Mashup \times API \times Proximity \longrightarrow Ratings \tag{3.11}$$

3.5.2 HOSVD Decomposition with Single Contextual Variable

Since this research employs *proximity* as the only contextual information used in HOSVD tensor data model, an illustration of a typical HOSVD decomposition with single contextual variable P is described in this section, hence, **Y**, which is the tensor holding the ratings, will be three-dimensional. While multiple contextual variables could be used, the generalization number of dimensions coupled with number of context variables is trivial (Karatzoglou et al., 2010). Assuming five star rating scale $\mathcal{Y}$ is given with 3-dimensions $\{0, \dots, 5\}^{m \times a \times p}$ for a sparse tensor $\mathbf{Y} \in \mathcal{Y}^{m \times a \times p}$, where m are the number of users (mashups in our case), a number of items (APIs in out case) and p

is the contextual variable (s) (i.e $p_i \in \{1, \ldots, p\}$ - in our case only *proximity* is used as a single contextual variable). For the rating $\mathcal{Y}$, the value 0 represents that a user (mashup) did not rate or consume an item (an API). It is worth to note that 0 in this case specially indicates *missing data* and not synonymous to *dislike*. Figure 3.3 shows the 3-dimensional tensor decomposed into 3-matrices $M \in \mathbb{R}^{m \times d_M}$, $A \in \mathbb{R}^{a \times d_A}$ and $P \in \mathbb{R}^{p \times d_P}$ and a core tensor $S \in \mathbb{R}^{d_M \times d_A \times d_P}$. With respect to this representation, the *decision function* for a single *user i* (mashup) , *item j* (like API) and *Context k* (e.g. proximity) fusion becomes :

$$F_{ijk} = S \times_m M_{i*} \times_a A_{j*} \times_p P_{k*} \tag{3.12}$$

M_{i*} represents the entries of the i_{th} row of matrix M . This factorization model enables full control over the dimensionality of the factors retrieved for the users, items and proximity (or any other context) by tuning the d_M, d_A , d_P parameters . This feature is very valuable especially when dealing with large-scale real-world datasets where both user's matrix and item matrix can grow size and cause storage problem (Karatzoglou et al., 2010).

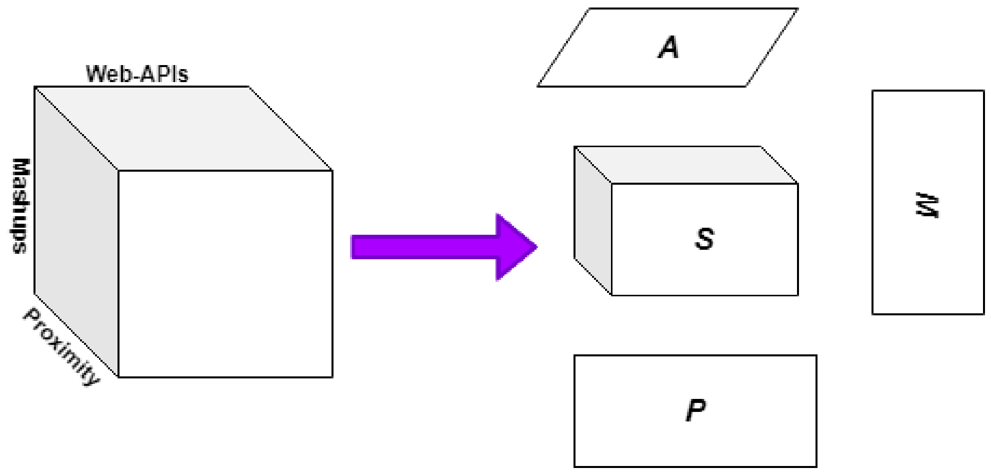


Figure 3.3: An illustration of 3-order HOSVD tensor factorization model of Web service data

3.5.3 Loss Function

In comparison with 2-D Matrix factorization approach described in section 2.4, the *loss function* for 3-order tensor is described as follows:

$$L(F,Y) := \frac{1}{||S||_1} \sum_{i,j,k} D_{ijk} l(F_{ijk}, Y_{ijk}) \tag{3.13}$$

Where $D \in \{0;1\}^{m \times a \times p}$ is a binary tensor which has non-zero entries D_{ijk} wherever Y_{ijk} is observed. $l : \mathbb{R} \times \mathcal{Y} \longrightarrow \mathbb{R}$ represents a *point-wise loss function* penalizing distance between the observation and the estimate . F_{ijk} already described in equation 3.12. It worth nothing that the overall loss L is meant to captured only the observed values in the sparse tensor Y and not the missing ones.

Generally, there different possible choices of approaches for estimating the *loss function* l. Some the common approaches are described below:

Squared Error

The squared error provides an estimate of the squared difference between the estimated values and what is estimated. It gives a computation of the conditional mean as follows:

$$l(f,y) = \frac{1}{2}(f-y)^2 \tag{3.14}$$

By taking the partial derivation of equation 3.14, it becomes $\partial_f l\ (f,y) = f - y$.

Absolute Loss

The absolute loss gives an estimate of the conditional median

$$l(f,y) = |f-y| \tag{3.15}$$

$$\partial_f l(f, y) = sgn[f - y] \tag{3.16}$$

While, the are other loss function possible (Karatzoglou et al., 2010) , this work focuses on the two described above.

3.5.4 Regularization

Usually, minimizing the loss function tends to lead to overfitting (i.e. model is adapting itself too much to the training data and thus, not generalizing well to the test data) the training data. Therefore, in order to reduce the overfitting effect on the test data, a regularization term is usually introduced. From equation 3.12, where S, M, A, P constitute the data model, we can optimize the complexity of the model and ensure it does not grow without bound. Hence, a L_2 *regularization* term or $l_2\ norm$ of the factors is added. For matrix , the norm can also be referred to as the *Frobenius norm*.

$$\Omega[M, A, P] := \frac{1}{2}[\lambda_M \, ||M||_F^2 + \lambda_A \, ||A||_F^2 + \lambda_P \, ||P||_F^2] \tag{3.17}$$

Similarly, the core tensor S complexity can also be restricted by imposing the $l_2\ norm$ penalty:

$$\Omega[S] := \frac{1}{2}[\lambda_S \, ||S||_F^2] \tag{3.18}$$

λ here is the regularization parameter that is data-dependent and determined by cross-validation - usually non-negative values that basically trade-off between the *"training error "* and the *"length"*.

3.5.5 Basic Optimization

To this extent, this framework will aim to optimize a regularized risk functional which is an aggregate of equation 3.13 and 3.17, that is $L(F,Y)$ and $\Omega[M,A,P]$. Hence, the objective function for the optimization problem becomes:

$$R[M,A,P,S] := L(F,Y) + \Omega[M,A,P] + \Omega[S] \tag{3.19}$$

To minimize the objective function, various approaches can be employed. For instance, in MF, *subspace descent* is a common approach for this type of problem and could also be applied to tensor model. In this approach, individual components of the above model are optimized iteratively, while other components are kept fixed. This approach will eventually rich convergence. Even-though the approach is quick to converge, it usually requires the optimization procedure to be run in a batch setting. Increase in data size usually makes the computation in-feasible by batch optimization Alternative algorithm can be applied to find the optimal solutions with computing gradient with respect to each factor with M, A and P . The algorithm will keep updating the variables until convergence or reaching the maximum iterations. For this work, an algorithm for performing *Stochastic Gradient Descent* (SDG) (Symeonidis et al., 2008) on the factors M_{i*}, A_{j*}, P_{k*} and S for a particular rating Y_{ijk} at thesame time employed. To calculate the updates for the SDG algorithm, the gradients of the loss function are calculated and then the objective function with reference to individual components in the model:

$$\partial_{M_{i*}} l(F_{ijk}, Y_{ijk}) = \partial_{Fijk} l(F_{ijk}, Y_{ijk}) S \times_a A_{j*} \times_p P_{k*}$$

$$\partial_{A_{j*}} l(F_{ijk}, Y_{ijk}) = \partial_{Fijk} l(F_{ijk}, Y_{ijk}) S \times_m M_{i*} \times_p P_{k*}$$

$$\partial_{P_{k*}} l(F_{ijk}, Y_{ijk}) = \partial_{Fijk} l(F_{ijk}, Y_{ijk}) S \times_a A_{j*} \times_m M_{i*}$$

$$\partial_{S} l(F_{ijk}, Y_{ijk}) = \partial_{Fijk} l(F_{ijk}, Y_{ijk}) M_{i*} \otimes A_{j*} \otimes P_{k*}$$

3.6 Geographical Distance Between Services

There are various factors influencing the network performance between the target user and the target web service. The most critical factors are *network distance* and *network bandwidth* (J. Liu, Tang, Zheng, Liu & Lyu, 2015), which are highly relevant to locations of the target user and the target service. Incorporation of location information between users and candidate services have been found to be an important influence in the consumption and selection of services. The location of service consumer and the service can influence the observed quality of rating in service invocation. Chen et al. (X. Chen, Liu, Huang & Sun, 2010) and (J. Liu et al., 2015) emphasis how critical the location of service with respect to users are in influencing the Quality of Service of the web-service. Location here implies the network environments that could affect the QoS attributes (like response time, throughput etc.) of a service. For instance, if the quality of network performance between a target user and the target Web service is high, the likelihood that the user will experience high QoS on the target service will increase. Moreover, users who reside in the same network region usually observe thesame response time with respect to service. Therefore, we consider geographical proximity between users and services as the single contextual variable in our recommendation framework, which is capable of influencing the invocation preferences of services with respect to mashups. Intuitively, we assume higher preferences for the API-mashups interaction with closer proximity.

3.6.1 Estimating Geographical Proximity Score

In order to capture the *proximity* score between a particular Web-API a_i and mashup m_j, this work employ *Haversine*[3] which is used for determining the great-circle distance between two points on a sphere given their longitudes and latitudes. The formula is

[3]https://en.wikipedia.org/wiki/Haversine_formula

defined below:

$$du, i = 2r.arcsin\Big(\sqrt{sin^2(\frac{\varphi_u - \varphi_i}{2}) + cos(\varphi_i)cos(\varphi_u)sin^2(\frac{\gamma_u - \gamma_i}{2})}\Big) \tag{3.20}$$

where $r = 6371$ denotes the Earth radius, $\varphi_u, \varphi_i \in (-180, 180]$ denote the latitudes in corresponding geolocations of u and i, and $\gamma_u, \gamma_i \in (-180, 180]$ also represent the respective longitudes. Relevant geographical information for the service proximity computations were acquired from the GeoIP database [4] via a lookup of each Web service and Mashup URLs. The resulting proximity score is normalized to be within the range [0.1, 1]. A min-max normalization is performed on d as follows:

$$z_{u,i} = 0.8.\frac{d_{u,i} - min(d)}{max(d) - min(d)} + 0.1 \tag{3.21}$$

$z_{u,i}$ represents the normalized values $d_{u,i}$. Therefore, the geographical proximity values between entities m_u and a_i can be simplified as follows:

$$g_s(m_u, a_i) = 1 - z_{u,i} \tag{3.22}$$

[4]https://www.maxmind.com/en/geoip2-databases

Chapter 4

Analysis

In this chapter, we described the implementation process of higher-order singular value decomposition (HOSVD) and predicted the potential correlation between mashup and APIs based on the three-dimensional tensor of $Mashup \times APIs \times Proximity$. Firstly, we presented raw data processing and three-dimensional tensor construction. Secondly, we show the algorithm of HOSVD and presented the code implemented HOSVD step by step. The key parts of the code were highlighted and discussed. Thirdly, we would show the results and discussed the performance of HOSVD.

4.1 Introduction

At present, all large websites such as Last.fm, Movielens and YouTube use social label recommendation system to classify items and share information among users, and gradually realize label classification and build corresponding user groups. However, there is a problem on the Internet, that is, some users feel that typing tags are very tedious, which results in the sparse data problem of tag system caused by users' unwillingness to provide tags. In addition, there are still some problems in the tag recommendation system, such as lexical differences and semantic ambiguity. Therefore, we need to find a kind of label that users think is the most appropriate and comprehensive interpretation of the item information, so that users can query, share and integrate the item information more effectively. For these reasons, recent studies have been devoted to mining user tags (tag metadata) on specific items to improve tag recommendation algorithms. Traditional recommendation systems generally use collaborative filtering recommendation algorithm based on two-dimensional data ((?, ?); Herlocker et al. 2002; Sun et al. 2006; Karypis 2001), while other algorithms combine labels into standard CD algorithm to form three-dimensional association data (Tso-Sutter et al. 2008), which is helpful to mine the potential semantic association among the three types of entities. For the latter, we need to solve two questions firstly: (i) the construction of three-dimensional relationship among users, items and tags; (ii) the sparsity of meta-data. In order to mine the potential semantic association between mashup and APIs, we first constructed three-dimensional tensor (proximity × Mashup × Apis) based raw data and decomposed it with HOSVD. Finally, we compared its performance with PMF algorithm based two-dimensional data. The results indicated that HOSVD is better than PMF in aspect of strong association prediction.

4.2 Tools

The following function modules were carried out in Python 3.6.3 environment (Oliphant 2007).

4.2.1 Pandas module

Pandas module (Bernard 2016) was used to get the number of column and row of data in *dataframe* format, conduct *dataframe* format conversion and basic operation associated with *dataframe* .

4.2.2 Sktensor and Tensorly modules

Sktensor and *Tensorly* modules (Kossaifi et al. 2019) were used to conduct three-dimensional tensor construction and decomposition.

4.2.3 Numpy module

Numpy module (Van Der Walt et al. 2011) were used to sample the training dataset and test dataset from raw data and conduct singular value decomposition for two-dimensional matrix.

4.2.4 Matplotlib module

Matplotlib module (Hunter 2007) was used to conduct analysis visualization, including heat map and line chart plotting.

4.3 Implementation

The implementation of the Web-service recommender system with HOSVD tensor decomposition approach is discussed as under three subsections. First, the construction of the three-dimensional tensor data model for the framework is described. Secondly, the sampling procedure for both the training dataset and testing set is described. Third, the Web API prediction procedure with tensor decomposition.

4.3.1 Three-dimensional tensor construction

We select the first and fifth columns in $mashup - data.csv$ and the seventh column in $mashup - city.csv$ to combine a new data frame and then construct three-dimensional matrix (Country × Mashup names × Apis).

```
def df2tb(df,name,tag,item):
  Ncol = df.shape[1]
  Nrow = df.shape[0]
  df_name = df.columns.values.tolist()[name]
  df_tag = df.columns.values.tolist()[tag]
  df_item = df.columns.values.tolist()[item]
  Users = []
  Tags = []
  Items = []
  for i in range(Nrow):
    tags_temp = literal_eval(df[df_tag][i])
    Length = len(tags_temp)
    for j in range(Length)
      Users.append(df[df_name][i])
      Tags.append(tags_temp[j])
      Items.append(literal_eval(df[df_item][i])[0])
  Tb = pd.DataFrame({df.columns[0]:Users,df.columns[1]:Tags,df.
   columns[2]:Items})
  return tb

def tb2tc(tc,tb):
  N1 = tc.shape[0]
  N2 = tc.shape[1]
  N3 = tc.shape[2]
  tb_users = tb.ix[:,0]
  tb_tags = tb.ix[:,1]
  tb_items = tb.ix[:,2]
  rn = tb._stat_axis.values.tolist()
  res = np.zeros([N1,N2,N3])
  for i in rn:
    res[tb_items[i]][tb_users[i]][tb_tags[i]]=1
    return(res)
```

To show the sparsity of the dataset, a $Mashup - Apis$ heat-map was plotted. Meanwhile, its sparsity was calculated by following formula:

$$Density = \frac{count(eij)}{ncol \times nrow}, e_{ij} \in A \; and \; e_{ij} \neq 0 \tag{4.1}$$

$$Sparsity = 1 - Density \tag{4.2}$$

The codes were as follow, we (i) set the non-zero elements as one and calculated the sum of this matrix; (ii) the sum of matrix divided by the product of the number of rows and columns of matrix; (iii) the sparsity was 1 – density.

```
def tb2mat(tb,r,c):
    rows = tb.ix[:,r]
    cols = tb.ix[:,c]
    nr = len(np.unique(rows))
    nc = len(np.unique(cols))
    res = np.zeros([nr,nc])
    rn = tb._stat_axis.values.tolist()
    for i in rn:
        res[rows[i]][cols[i]] = 1
    return(res)
#Sparisity
mat1 = tb2mat(d0,0,1)
density=sum(sum(mat1))/(mat1.shape[0]*mat1.shape[1])
#density = 0.0015
sparsity = 1-density
#sparsity = 0.9985
```

```
Then, this matrix was converted into tensor 'A' with 'dtensor'
    function in 'sktensor' module.
def TC(tb):
  Nrow = tb.shape[0]
  Ncol = tb.shape[1]
  tb_users = tb.ix[:,0]
  tb_tags = tb.ix[:,1]
  tb_items = tb.ix[:,2]
  tb_tags_unique = np.unique(tb_tags)
  tb_items_unique = np.unique(tb_items)
  tb_users_unique = np.unique(tb_users)
  Tc = np.zeros((len(tb_items_unique),len(tb_users_unique),len(
    tb_tags_unique)),float)
  for index,values in enumerate(tb_items_unique):
    items_temp = enumerate_fn(tb_items,values)
    for i in items_temp:
      r1 = enumerate_fn(tb_users_unique,tb_users[i])[0]
      r2 = enumerate_fn(tb_tags_unique,tb_tags[i])[0]
      print(index,"/",(len(tb_items_unique)),": [",r1,",",r2,"]")
      Tc[index][r1][r2] = 1
  return Tc
```

4.3.2 Sampling the training set and testing set

After relabeling the element of tensor A, we randomly sampled a certain proportion of tensor elements as training set, while the rest of elements in tensor as testing set. The codes were as following:

```
ratio = 0.6
```

```
train_tb = d0.loc[d0_cn[0:int(ratio*len(d0_cn))]]
test_tb = d0.loc[d0_cn[int(ratio*len(d0_cn)):len(d0_cn)]]
train_tc = tb2tc(otc, train_tb)
test_tc = tb2tc(otc, test_tb)
```

4.3.3 Web API Prediction With Tensor Decomposition

The Web API prediction approach applies tensor factorization algorithm based on HOSVD (as discussed in the method chapter) on the API data. In accordance to the HOSVD approach introduced in Section 3.5, the algorithm uses as input the web service data of tensor $\mathcal{A}$ and output the reconstructed tensor $\hat{\mathcal{A}}$. Tensor $\hat{\mathcal{A}}$ compute the latent association among the API, Mashup (users) and the proximity (or location) For each elements in tensor $\hat{\mathcal{A}}$ can be represented by by $\{m, a, p, r\}$. r measures the likeliness that mashups m will consume API a within proximity p. Hence, $API\ a$ can be predicted /recommended to based o on the weight attributed $\{m, a\}$ pair. The three-dimensional tensor was unfolded into three two-dimensional matrices along with three dimensions of A1, A2 and A3. It was conducted with $unfold$ function in $tensorly$ module. Secondly, these three two-dimensional matrices were perform the *singular value decomposition* (SVD) (Sun et al. 2005) via svd function in $numpy$ module. After SVD, we can get core tensor ($S^1, S^2 and S^3$), left singular matrices $(U^1, U^2\ and\ U^3)$ and transposition matrices of right singular matrix $(V^{(1)T}, V^{(2)T}\ and\ V^{(3)T})$.

Denoising the left singular matrix U

We sorted the diagonal matrix element of in descending order. The elements that their accumulative proportions were not less than 90% the sum of all diagonal matrix element was kept, while the rest were set to zero. Meanwhile, the corresponding column of

U^1, U^2 *and* U^3 also were set to zero and then got similar matrix U1, U2 and U3 form original matrices.

Calculating the similar core tensor

The core tensor $\mathcal{S}$ administers the relationships among the 3-entities (mashup, Api and proximity) From the inital $\mathcal{A}$, we constructed the similar core tensor $\mathcal{S}$ by following codes:

$$\mathcal{S} = A \times_1 U^1 \times_2 U^2 \times_3 U^3 \tag{4.3}$$

Reconstruction of similar tensor

According to formular 4.3, we calculated the similar tensor $\hat{\mathcal{A}}$ by the product of the core tensor $\mathcal{S}$ coupled with the product of the 3-matrices:

$$\hat{\mathcal{A}} = \mathcal{S} \times_1 U^1 \times_2 U^2 \times_3 U^3 \tag{4.4}$$

The processes described above were performed with following codes: The codes were as following:

```
def myHOSVD(train_tc ,denoising):
   tc = dtensor(train_tc)
   a0 = tl.unfold(tc ,0)
   a1 = tl.unfold(tc ,1)
   a2 = tl.unfold(tc ,2)
   U0,S0,V0 = la.svd(a0, full_matrices=False)
   U1,S1,V1 = la.svd(a1, full_matrices=False)
   U2,S2,V2 = la.svd(a2, full_matrices=False)
   best_rank = [minRank(S0,denoising),minRank(S1,denoising),minRank(
     S2,denoising)]
   U0 = np.array(U0)
```

```
    U1 = np.array(U1)
    U2 = np.array(U2)
    U0s = np.array(U0[:,range(best_rank[0]+1)])
    U1s = np.array(U1[:,range(best_rank[1]+1)])
    U2s=np.array(U2[:,range(best_rank[2]+1)])
    uu = [U0s.T,U1s.T,U2s.T]
    s = ttm(tc,uu)
    u = [U0s,U1s,U2s]
    d2_1 = ttm(s,u)
    return(d2_1)
def minRank(S,ther):
    Sr = sorted(S,reverse=True)
    T = sum(Sr)*ther
    res = 0
    for index,value in enumerate(S):
        res = res+value
        if(res >= t):
            res1 = index
            break
    return(res1)
```

4.3.4 The performance of HOSVD

We calculated RMSE and MAE to access the performance of HOSVD prediction based on top predictions(Karypis 2001) according to following two formulas.

$$RMSE = \sqrt{\sum(prediction - true)^2} \tag{4.5}$$

$$MAE = mean(\sum abs(prediction - true)) \tag{4.6}$$

The codes were as following.

```
def myRMSE(preds,top_N):
  T = sorted(preds,reverse=True)[:top_N]
  eui = [np.square(i-1) for i in T]
  return(np.sqrt(np.mean(eui)))

def myMAE(preds,top_N):
  T = sorted(preds,reverse=True)[:top_N]
  eui = [np.abs(i-1) for i in T]
  return(np.mean(eui))
```

4.3.5 Comparison with PMF

We used the following codes provided by teacher get the PMF (Bao et al. 2013; Yang et al. 2013) results.

```
class PMF():
    '''
    a class for this Double Co-occurence Factorization model
    '''
    def __init__(self, R, lambda_alpha=1e-2, lambda_beta=1e-2,
   latent_size=50, momuntum=0.8,
                 lr=0.001, iters=1000, seed=None):
        self.lambda_alpha = lambda_alpha
        self.lambda_beta = lambda_beta

        self.momuntum = momuntum
        self.R = R
        self.random_state = RandomState(seed)
        self.iterations = iters
        self.lr = lr
        self.I = copy.deepcopy(self.R)
        self.I[self.I != 0] = 1
        self.U = 0.1*self.random_state.rand(np.size(R, 0),
    latent_size)
        self.V = 0.1*self.random_state.rand(np.size(R, 1),
    latent_size)
    def loss(self):
        loss = np.sum(self.I*(self.R-np.dot(self.U, self.V.T))**2) +
     self.lambda_alpha*np.sum(np.square(self.U)) + self.lambda_beta*
    np.sum(np.square(self.V))
        return loss
    def predict(self, data):
        index_data = np.array([[int(ele[0]), int(ele[1])] for ele in
     data], dtype=int)
        u_features = self.U.take(index_data.take(0, axis=1), axis=0)
        v_features = self.V.take(index_data.take(1, axis=1), axis=0)
        preds_value_array = np.sum(u_features*v_features, 1)
        return preds_value_array
    def train(self, train_data=None, vali_data=None):
        train_loss_list = []
        vali_rmse_list = []
        last_vali_rmse = 1000
        temp_U = np.zeros(self.U.shape)
        temp_V = np.zeros(self.V.shape)

        for it in range(self.iterations):
            grads_u = np.dot(self.I*(self.R-np.dot(self.U, self.V.T)
    ), -self.V) + self.lambda_alpha*self.U
```

```
            grads_v = np.dot((self.I*(self.R-np.dot(self.U, self.V.T
    ))).T, -self.U) + self.lambda_beta*self.V
            temp_U = (self.momuntum * temp_U) + self.lr * grads_u
            temp_V = (self.momuntum * temp_V) + self.lr * grads_v

            self.U = self.U - temp_U
            self.V = self.V - temp_V
            train_loss = self.loss()
            train_loss_list.append(train_loss)
            vali_preds = self.predict(train_data)
            vali_rmse = RMSE(train_data[:,2], vali_preds)
            vali_rmse_list.append([it+1,vali_rmse])
            print('training iteration:{: d} ,loss:{: f}, vali_rmse
    :{: f}'.format(it+1, train_loss, vali_rmse))
return self.U, self.V, train_loss_list, vali_rmse_list
```

We also calculated $RMSE$ and MAE and compared it with HOSVD as follows:

```
R = np.zeros([train_tc.shape[1], train_tc.shape[2]])
for ele in range(train_Tb.shape[0]):
  R[int(train_Tb.ix[ele,0]),int(train_Tb.ix[ele,1])]=float(train_Tb.
    ix[ele,2])
lambda_alpha = 0.01
lambda_beta = 0.01
latent_size = 15
lr = 0.0005
iters = 0
model = PMF(R=R, lambda_alpha=lambda_alpha, lambda_beta=lambda_beta,
      latent_size=latent_size, momuntum=0.9, lr=lr, iters=iters, seed
    =1)
```

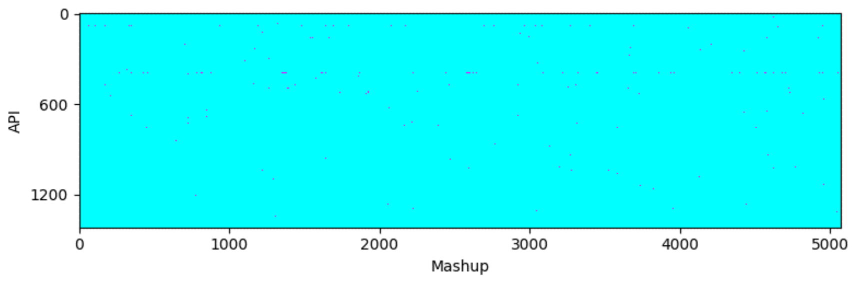


Figure 4.1: The heat-map of Mashup-API matrix

```
preds = model.predict(data=np.array(test_Tb))
```

4.4 Findings

4.4.1 Data sparsity

The three-dimensional tensor of $proximity \times mashup \times apis$ captured 49 countries, 5071 mashups and 1418 Apis. Then, the sparsity of datasets was calculated with custom function. The result showed that the sparsity of two-dimensional matrix of Mashup-Apis was 0.9985, which showed in heat-map Figure 4.1. These results indicated that the data was sparse. Thus, it was appropriated for predicting the potential correlations between mashup and APIs.

4.4.2 The performance of HOSVD and PMF with different size training set

For evaluating the performance of HOSVD algorithm after remove 10% noise of left singular matrices, we selected a certain proportion of top prediction to calculate the $RMSE$ and MAE based training dataset with different size sampled from the original dataset. We randomly resample 10%, 20%, 30%, 40%, 50%, 60% training dataset to fit the PMF model and $HOSVD$ model. The results were listed in Table 1 and showed in Figure 2. Table 1 showed that the $RMSE$ and MAE of PMF-based top 50 predictions slightly increased with the increasing size of the training set and independent on the size of training sets.

Table 4.1: The RMSE and MAE of HOSVD and PMF model with different size training set.

Percentage	RMSE(HOSVD)	RMSE(PMF)	MAE(HOSVD)	MAE(PMF)
10%	0.9703	0.9384	0.9660	0.9383
20%	0.9381	0.9386	0.9328	0.9386
30%	0.8926	0.9393	0.8820	0.9393
40%	0.9015	0.9397	0.8935	0.9397
50%	0.8744	0.9403	0.8671	0.9403
60%	0.8532	0.9408	0.8426	0.9408

As shown in Figures 4.2 the performance of HOSVD was dependent on the size of training dataset that is the performance would increase with increasing size of training data set. When the size of training data set was less than 20%, the performance of HOSVD model (blue line) was more bad than PMF model (red line) indicated by RMSE and MAE based top 50 predictions. When the size of training data set was more than 20%, the performance of HOSVD model (blue line) was better than PMF model. These results revealed that HOSVD model has better robust than PMF model when data sets are highly sparse. Notably, when the size of training set were more than 30% and less than 40%, the performances of HOSVD model was decreased.

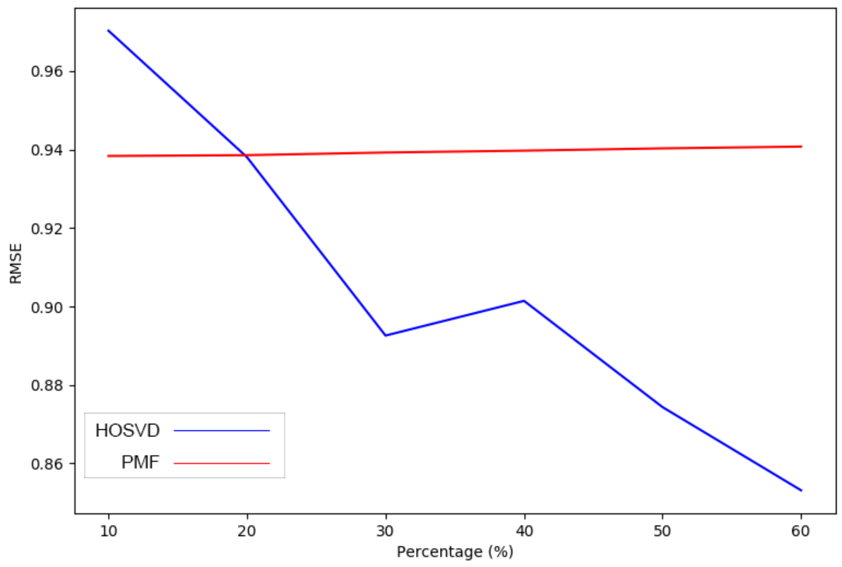


Figure 4.2: The RMSE of HOSVD and PMF based training set with different size

4.4.3 The denoising influence on performance of HOSVD

In HOSVD, new three-dimensional tensor was reconstructed after denoising the left singular matrices. We evaluated the influence of denoising ratio on HOSVD performance based top 50 predictions. Figure 4.3 and 4.4 showed that the RMSE and MAE were linearly decreased with increasing denoising ratio. Increasing denoising ratio would make the loss of original information from raw data set, but the accuracy top prediction representing strong inner association between mashup and APIs still increase. It indicated HOSVD has better performance to predict true relationships between variables.

4.4.4 Prediction accuracy comparison between HOSVD and PMF

Based on the same training set, we trained the $HOSVD$ and PMF model and calculate the $RMSE$ based 60% trained dataset. The results showed that the $RMSE$

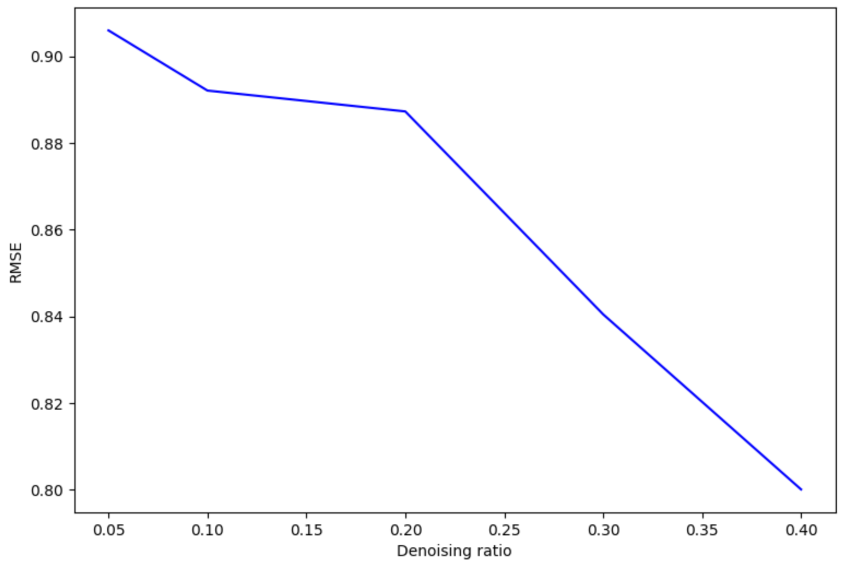


Figure 4.3: The influences of denoising ratio on the performance of HOSVD

of $HOSVD$ was increased with increased top predicted results sorted weight value between mashup and APIs reversely $(from\ 0.6\ to\ 0.99)$, while PMF was stable and around 0.94. Notably, the RMSE of HOSVD was less than PMF results when prediction results occupied less top 100 predicted results, which indicated that HOSVD has better performance than PMF when they predict stronger relevance between mashup and APIs.

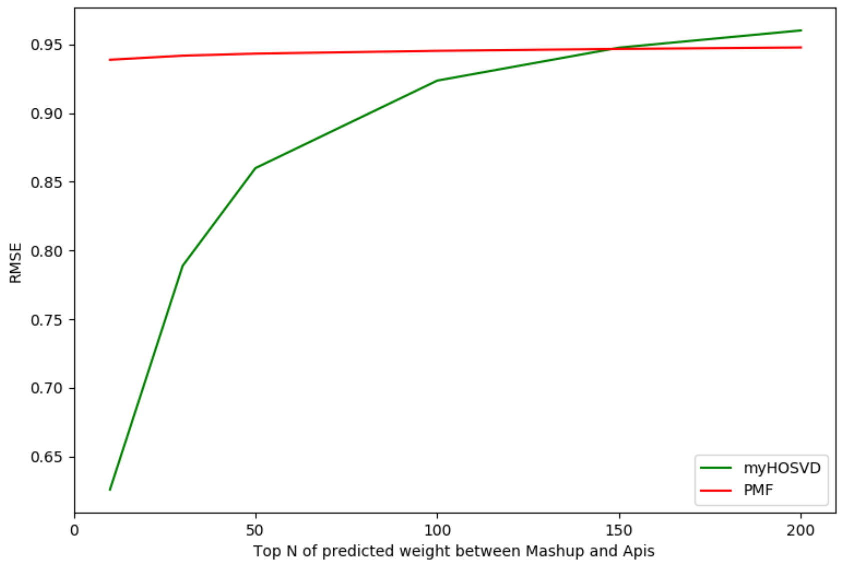


Figure 4.4: The comparison between HOSVD and MAE performance

Chapter 5

Discussion

In this Chapter, we discuss the results from the implementation with respect to the two research questions discussed in section 1.2. First, we discuss how we are able to represent the Web-API recommendation problem as a 3-order data representation task by completing the unobserved entries in the rating matrix. Then we compare the impact of context information integration into the rating using HOSVD with the conventional 2-D Probabilistic Matrix Factorization (PMF). We discuss the impact of the dimensionality .

5.1 Results Assessment

For recommend the APIs to mashup, we constructed the recommend system based HOSVD algorithm with the three-dimensional tensor of proximity $\times$ Mashup $\times$ Apis and compared its performance with PMF algorithm. The algorithm had the task of predicting the API with respect to proximity of the mashup (user) location in the testing dataset.

5.1.1 Impact of Dimensionality

The results shown confirm the effectiveness of HOSVD-based rating data model, that is incorporating multiple dimension of contextual (in this case proximity) information into the three-dimensional tensor was better than two-dimensional PMF matrix because of the former include more information. Even-though, both HOSVD and PMF solve the sparsity issue associated with the rating matrix; the RMSE results of both methods indicate that the HOSVD approach was able to achieve less error compare with PMF.

According to RMSE of both algorithms, we realized that HOSVD has the advantage and robust of predicting closed relationships between variables than PMF. However, its performance was similar to PMF algorithm when they predicted weaken correlations even weaken. Thus, we think that it was better to select HOSVD for mining potential correlations based sparse data rather than PMF. However, notably, the robust of HOSVD was significantly affected by the size of train dataset, while it was not affected by the denoising coefficient. Particularly, the HOSVD performance would worse than PMF when the size of train dataset less than 20 percentage of the whole dataset. Therefore, HOSVD model was more applicable to predict small samples according to large and sparse samples.

5.1.2 Impact of HOSVD Tensor Density

In order to have a clear insight into the impact of different tensors densities on the rating prediction, we consider the accuracy of our approach under different densities and compare the prediction results. The density of the training set used in the experiment was varied between the range of 10% to 90% with 10% steps at a time. Then we record the prediction values under each different matrix density. As shown in figures 4.2, 4.3, we can observe that the accuracy of our method is improved gradually with the increase in the density of the training dataset. This is an indication that with an even more dense data, improved accuracy can be achieved.

5.1.3 Impact of Regularization Parameter

As discussed in 3.5.4, a standard procedure for reducing the problem of overfitting usually faced by the test data is introduction of regularization term λ. We consider the impact of tuning the regularization parameter by varying its value within the following values$\{10^{-4}, 10^{-3}, 10^{-2}, \ldots 10^{2}\}$. By doing this, we observed that the accuracy of the prediction increases until λ reaches 10^{-2} and then decreases with larger values.

5.2 Contributions

This research work proposes, proximity-aware, collaborative filtering method using tensor factorization, a generalization of the Matrix factorization for web service recommendation. The web service prediction model described in this work attempted to predict the potential associations between mashups and APIs with respect proximity (location) information. Unlike previous approaches which are based on 2-dimensional data models, specifically the standard Matrix Factorization model that utilized the sparse mashup-API interaction matrix to compute a low-rank approximation rating matrix, this research shows how Matrix factorization can be extended by increasing the dimensionality of concern, which enables the exploitation of relationship that exists among Web-API, mashups and proximity (as contextual variable) information when recommending Web-APIs but also.

In relation to the two research questions defined for this research work, we summarize the contributions as follows:

1. This work shows how to construct potential interactions between mashups and APIs with respect to their proximity using three-dimensional tensor representation

to solve Web-API recommendation problem.

2. The work provides a template on how to exploits a high-order tensor to integrate contextual information relating to Web-API consumption instead of the traditional two-dimensional mashup-API matrix. The decomposition of this tensor results to a more compact data model that is naturally suitable for fusing contextual information to support Web-API recommendations.
3. Using a real-world dataset, specifically ProgrammableWeb Web-API dataset, this work shows the effectiveness of the tensor-based data model approach for API recommendation. Using two popular prediction metrics (RMSE and MAE), we demonstrated the superiority of our proposed tensor-model with Probabilistic Matrix Factorization model.

Chapter 6

Conclusion

6.1 Challenges

One of the challenges of this study was to get valid training datasets and decide the top of effective predictions. Since the HOSVD model would generate many invalid predictions and its robust and accuracy of prediction affected the size and sparsity of training dataset. In reality, the training sets would more sparsity and more fragmented than the datasets used in this study when we sample this data from the internet. Another was memory management when the HOSVD model run. Before the decomposition of high-order tensor, it needs to construct multiple matrices first. This process has slow efficiency and needs huge storage space so that HOSVD model did not apply to huge dataset. When dealing with large-scale data, especially when the data is non-linear, it can greatly simplify the calculation by mapping the input space to the high-dimensional space through the non-linear change of the kernel function. Xiao (Xiao et al. 2016) found that the kernel-based method was applied to the representation of low-rank matrices. RKLRR, RKLRS and RKNLRS algorithms were proposed. The performance of clustering was improved several times, but the error of clustering was much smaller than that of traditional methods. The representation of a low-rank matrix based on kernel function has only made some progress in low-rank matrices at present, and its application to high-order tensor data remains to be studied. This will inevitably become one of the research directions in the fields of community discovery, recommendation system, signal processing and so on. It has great research value and significance for Li (Tao et al. 2009). In this paper, a non-negative tensor decomposition algorithm (GNTF) based on graph and low-rank representation is proposed. Compared with the existing classification algorithm, the classification effect of the image is improved. However, in this method, the constraints are not selected to compare the results, and the optimization of the performance of the algorithm by kernel function is not considered. In this classifier, Choosing appropriate constraints and applying the kernel function to them will further improve the performance of the classifier and have a good effect on the improvement of the algorithm.

6.2 Future Work

In tensor decomposition, it is very important to make full and reasonable use of the structure of high-dimensional data for problem modelling. In CP decomposition, it is helpful to find the problem of matrix rank minimization and analyze the internal structure information of the matrix for matrix filling and recovery performance. For large-scale data, based on the existing hardware conditions, explore the effective and parallelization of tensor decomposition algorithm for the solution of the problem has important practical value and practical application, such as in nuclear matrix norm minimization problem, singular value decomposition. The most time consuming is in the solving process, such as the dimension of m * n matrices with time complexity O (m*n2). Putting forward two kinds of block SVD signal processing algorithm and two kinds of segmentation algorithm can greatly reduce the processing time. From the examples described in the previous section, we have some understanding of the functions of the aforementioned methods in large-scale data and high-dimensional signal analysis and processing, and these theories provide theoretical support for practical application. Based on the principle or method of matrix rank minimization, low-rank matrix recovery and kernel function, tensor decomposition is used to fully mine the information in large-scale data, so as to design reasonable mathematical model and algorithm, which has a broad application prospect.

References

Adeleye, O., Yu, J., Yongchareon, S., Sheng, Q. Z. & Yang, L. H. (2019). A fitness-based evolving network for web-apis discovery. In *Proceedings of the australasian computer science week multiconference* (p. 49).

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.

Adomavicius, G. & Zhang, J. (2012). Impact of data characteristics on recommender systems performance. *ACM Transactions on Management Information Systems (TMIS)*, *3*(1), 3.

Bensmail, H. & Celeux, G. (1996). Regularized gaussian discriminant analysis through eigenvalue decomposition. *Journal of the American statistical Association*, *91*(436), 1743–1748.

Berry, M. W., Dumais, S. T. & O'Brien, G. W. (1995). Using linear algebra for intelligent information retrieval. *SIAM review*, *37*(4), 573–595.

Billsus, D. & Pazzani, M. J. (1999). A hybrid user model for news story classification. In *Um99 user modeling* (pp. 99–108). Springer.

Blei, D., Ng, A. & Jordan, M. (2003). Latent dirichlet allocation journal of machine learning research (3).

Bobadilla, J., Hernando, A., Ortega, F. & Bernal, J. (2011). A framework for collaborative filtering recommender systems. *Expert Systems with Applications*, *38*(12), 14609–14623.

Bobadilla, J., Hernando, A., Ortega, F. & Gutiérrez, A. (2012). Collaborative filtering based on significances. *Information Sciences*, *185*(1), 1–17.

Bobadilla, J., Ortega, F., Hernando, A. & Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-based systems*, *46*, 109–132.

Bokde, D., Girase, S. & Mukhopadhyay, D. (2015). Matrix factorization model in collaborative filtering algorithms: A survey. *Procedia Computer Science*, *49*, 136–146.

Bro, R. (1997). Parafac. tutorial and applications. *Chemometrics and intelligent laboratory systems*, *38*(2), 149–171.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, *12*(4), 331–370.

Burke, R. (2007). Hybrid web recommender systems. In *The adaptive web* (pp. 377–408). Springer.

Cao, B., Liu, X., Li, B., Liu, J., Tang, M., Zhang, T. & Shi, M. (2016). Mashup service clustering based on an integration of service content and network via exploiting a two-level topic model. In *2016 ieee international conference on web services (icws)* (pp. 212–219).

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*.

Cao, J., Wu, Z., Wang, Y. & Zhuang, Y. (2013). Hybrid collaborative filtering algorithm for bidirectional web service recommendation. *Knowledge and information systems*, *36*(3), 607–627.

Chen, L., Wang, Y., Yu, Q., Zheng, Z. & Wu, J. (2013). Wt-lda: user tagging augmented lda for web service clustering. In *International conference on service-oriented computing* (pp. 162–176).

Chen, L., Yu, Q., Philip, S. Y. & Wu, J. (2015). Ws-hfs: A heterogeneous feature selection framework for web services mining. In *2015 ieee international conference on web services* (pp. 193–200).

Chen, S., Wang, F. & Zhang, C. (2007). Simultaneous heterogeneous data clustering based on higher order relationships. In *Seventh ieee international conference on data mining workshops (icdmw 2007)* (pp. 387–392).

Chen, W., Paik, I. & Hung, P. C. (2015). Constructing a global social service network for better quality of web service discovery. *IEEE transactions on services computing*, *8*(2), 284–298.

Chen, X., Liu, X., Huang, Z. & Sun, H. (2010). Regionknn: A scalable hybrid collaborative filtering algorithm for personalized web service recommendation. In *2010 ieee international conference on web services* (pp. 9–16).

Chen, X., Zheng, Z. & Lyu, M. R. (2014). Qos-aware web service recommendation via collaborative filtering. In *Web services foundations* (pp. 563–588). Springer.

Chen, X., Zheng, Z., Yu, Q. & Lyu, M. R. (2013). Web service recommendation via exploiting location and qos information. *IEEE Transactions on Parallel and distributed systems*, *25*(7), 1913–1924.

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.

Do, P., Pham, P., Phan, T. & Nguyen, T. (2018). T-mpp: A novel topic-driven meta-path-based approach for co-authorship prediction in large-scale content-based heterogeneous bibliographic network in distributed computing framework by spark. In *International conference on intelligent computing & optimization* (pp. 87–97).

Ekstrand, M. D., Riedl, J. T., Konstan, J. A. et al. (2011). Collaborative filtering recommender systems. *Foundations and Trends® in Human–Computer Interaction*, *4*(2), 81–173.

Elmeleegy, H., Ivan, A., Akkiraju, R. & Goodwin, R. (2008). Mashup advisor: A recommendation tool for mashup development. In *2008 ieee international conference on web services* (pp. 337–344).

Fan, X., Hu, Y., Zheng, Z., Wang, Y., Brezillon, P. & Chen, W. (2017). Casr-tse: context-aware web services recommendation for modeling weighted temporal-spatial effectiveness. *IEEE Transactions on Services Computing*.

Friedman, N., Geiger, D. & Goldszmidt, M. (1997). Bayesian network classifiers. *Machine learning*, *29*(2-3), 131–163.

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).

Gillis, N. (2014). The why and how of nonnegative matrix factorization. *Regularization, Optimization, Kernels, and Support Vector Machines*, *12*(257).

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.

Golub, G. H. & Van Loan, C. F. (2012). *Matrix computations* (Vol. 3). JHU press.

Guan, N., Tao, D., Luo, Z. & Yuan, B. (2012). Nenmf: An optimal gradient method for nonnegative matrix factorization. *IEEE Transactions on Signal Processing*, *60*(6), 2882–2898.

Herlocker, J. L., Konstan, J. A., Terveen, L. G. & Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, *22*(1), 5–53.

Hu, Y., Peng, Q., Hu, X. & Yang, R. (2015). Web service recommendation based on time series forecasting and collaborative filtering. In *2015 ieee international conference on web services* (pp. 233–240).

Huang, C., Yao, L., Wang, X., Benatallah, B., Zhang, S. & Dong, M. (2018). Expert recommendation via tensor factorization with regularizing hierarchical topical relationships. In *International conference on service-oriented computing* (pp. 373–387).

Isinkaye, F., Folajimi, Y. & Ojokoh, B. (2015). Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, *16*(3), 261–273.

Jamali, M. & Lakshmanan, L. (2013). Heteromf: recommendation in heterogeneous information networks using context dependent factor models. In *Proceedings of the 22nd international conference on world wide web* (pp. 643–654).

Kang, G., Tang, M., Liu, J., Liu, X. F. & Cao, B. (2016). Diversifying web service recommendation results via exploring service usage history. *IEEE Transactions on Services Computing*, *9*(4), 566–579.

Karatzoglou, A., Amatriain, X., Baltrunas, L. & Oliver, N. (2010). Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the fourth acm conference on recommender systems* (pp. 79–86).

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

Lam, X. N., Vu, T., Le, T. D. & Duong, A. D. (2008). Addressing cold-start problem in recommendation systems. In *Proceedings of the 2nd international conference on ubiquitous information management and communication* (pp. 208–211).

Lecue, F. (2010). Combining collaborative filtering and semantic content-based approaches to recommend web services. In *2010 ieee fourth international conference on semantic computing* (pp. 200–205).

Lee, Y.-J. & Kim, C.-S. (2011). A learning ontology method for restful semantic web services. In *2011 ieee international conference on web services* (pp. 251–258).

Linden, G., Smith, B. & York, J. (2003). Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*(1), 76–80.

Liu, J., Tang, M., Zheng, Z., Liu, X. F. & Lyu, S. (2015). Location-aware and personalized collaborative filtering for web service recommendation. *IEEE Transactions on Services Computing*, *9*(5), 686–699.

Liu, L., Lecue, F. & Mehandjiev, N. (2013). Semantic content-based recommendation of software services using context. *ACM Transactions on the Web (TWEB)*, *7*(3), 17.

Liu, X., He, Q., Tian, Y., Lee, W.-C., McPherson, J. & Han, J. (2012). Event-based social networks: linking the online and offline social worlds. In *Proceedings of the 18th acm sigkdd international conference on knowledge discovery and data mining* (pp. 1032–1040).

Luo, X., Xia, Y. & Zhu, Q. (2012). Incremental collaborative filtering recommender based on regularized matrix factorization. *Knowledge-Based Systems*, *27*, 271–280.

Ma, H., Yang, H., Lyu, M. R. & King, I. (2008). Sorec: social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th acm conference on information and knowledge management* (pp. 931–940).

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

Moler, C. B. & Stewart, G. W. (1973). An algorithm for generalized matrix eigenvalue problems. *SIAM Journal on Numerical Analysis*, *10*(2), 241–256.

Park, D. H., Kim, H. K., Choi, I. Y. & Kim, J. K. (2012). A literature review and classification of recommender systems research. *Expert systems with applications*, *39*(11), 10059–10072.

Park, M.-H., Hong, J.-H. & Cho, S.-B. (2007). Location-based recommendation system using bayesian user's preference model in mobile devices. In *International conference on ubiquitous intelligence and computing* (pp. 1130–1139).

Platzer, C., Rosenberg, F. & Dustdar, S. (2009). Web service clustering using multidimensional angles as proximity measures. *ACM Transactions on Internet Technology (TOIT)*, *9*(3), 11.

Porcel, C., Tejeda-Lorente, A., Martínez, M. & Herrera-Viedma, E. (2012). A hybrid recommender system for the selective dissemination of research resources in a technology transfer office. *Information Sciences*, *184*(1), 1–19.

Rashid, A. M., Karypis, G. & Riedl, J. (2008). Learning preferences of new users in recommender systems: an information theoretic approach. *Acm Sigkdd Explorations Newsletter*, *10*(2), 90–100.

Rendle, S. (n.d.). *Context-aware ranking with factorization models*. Springer.

Rendle, S. & Schmidt-Thieme, L. (2010). Pairwise interaction tensor factorization for personalized tag recommendation. In *Proceedings of the third acm international* *conference on web search and data mining* (pp. 81–90).

Sarwar, B., Karypis, G., Konstan, J. & Riedl, J. (2000). *Application of dimensionality reduction in recommender system-a case study* (Tech. Rep.). Minnesota Univ Minneapolis Dept of Computer Science.

Schafer, J. B., Frankowski, D., Herlocker, J. & Sen, S. (2007). Collaborative filtering recommender systems. In *The adaptive web* (pp. 291–324). Springer.

Schein, A. I., Popescul, A., Ungar, L. H. & Pennock, D. M. (2002). Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international acm sigir conference on research and development in information retrieval* (pp. 253–260).

Shao, L., Zhang, J., Wei, Y., Zhao, J., Xie, B. & Mei, H. (2007). Personalized qos prediction forweb services via collaborative filtering. In *Ieee international conference on web services (icws 2007)* (pp. 439–446).

Shi, C., Hu, B., Zhao, W. X. & Philip, S. Y. (2019). Heterogeneous information network embedding for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, *31*(2), 357–370.

Smyth, B. & Cotter, P. (2000). A personalized television listings service. *Communications of the ACM*, *43*(8), 107–111.

Su, K., Xiao, B., Liu, B., Zhang, H. & Zhang, Z. (2017). Tap: A personalized trust-aware qos prediction approach for web service recommendation. *Knowledge-Based Systems*, *115*, 55–65.

Sun, P. & Jiang, C. (2008). Using service clustering to facilitate process-oriented semantic web service discovery. *Chinese Journal of Computers*, *31*(8), 1340–1353.

Symeonidis, P. (2009). User recommendations based on tensor dimensionality reduction. In *Ifip international conference on artificial intelligence applications and innovations* (pp. 331–340).

Symeonidis, P., Nanopoulos, A. & Manolopoulos, Y. (2008). Tag recommendations based on tensor dimensionality reduction. In *Proceedings of the 2008 acm conference on recommender systems* (pp. 43–50).

Symeonidis, P., Papadimitriou, A., Manolopoulos, Y., Senkul, P. & Toroslu, I. (2011). Geo-social recommendations based on incremental tensor reduction and local path traversal. In *Proceedings of the 3rd acm sigspatial international workshop on location-based social networks* (pp. 89–96).

Symeonidis, P. & Zioupos, A. (2016). *Matrix and tensor factorization techniques for recommender systems* (Vol. 1). Springer.

Tan, W., Fan, Y., Ghoneim, A., Hossain, M. A. & Dustdar, S. (2016). From the service-oriented architecture to the web api economy. *IEEE Internet Computing*, *20*(4), 64–68.

Tucker, L. R. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika*, *31*(3), 279–311.

Vahedian, F., Burke, R. & Mobasher, B. (2017). Multirelational recommendation in heterogeneous networks. *ACM Transactions on the Web (TWEB)*, *11*(3), 15.

Van Der Maaten, L., Postma, E. & Van den Herik, J. (2009). Dimensionality reduction: a comparative. *J Mach Learn Res*, *10*(66-71), 13.

Vozalis, M. G. & Margaritis, K. G. (2007). Using svd and demographic data for the enhancement of generalized collaborative filtering. *Information Sciences*, *177*(15), 3017–3037.

Wang, G., Xu, D., Qi, Y. & Hou, D. (2008). A semantic match algorithm for web services based on improved semantic distance. In *2008 4th international conference on next generation web services practices* (pp. 101–106).

Watkins, D. S. (2004). *Fundamentals of matrix computations* (Vol. 64). John Wiley & Sons.

Wu, C., Qiu, W., Zheng, Z., Wang, X. & Yang, X. (2015). Qos prediction of web services based on two-phase k-means clustering. In *2015 ieee international conference on web services* (pp. 161–168).

Wu, W., Zhao, J., Zhang, C., Meng, F., Zhang, Z., Zhang, Y. & Sun, Q. (2017). Improving performance of tensor-based context-aware recommenders using bias tensor factorization with context feature auto-encoding. *Knowledge-Based Systems*, *128*, 71–77.

Xia, B., Fan, Y., Tan, W., Huang, K., Zhang, J. & Wu, C. (2015). Category-aware api clustering and distributed recommendation for automatic mashup creation. *IEEE Transactions on Services Computing*, *8*(5), 674–687.

Xia, H. & Yoshida, T. (2007). Web service recommendation with ontology-based similarity measure. In *Second international conference on innovative computing, informatio and control (icicic 2007)* (pp. 412–412).

Xu, Y., Yin, J., Deng, S., Xiong, N. N. & Huang, J. (2016). Context-aware qos prediction for web service recommendation and selection. *Expert Systems with Applications*, *53*, 75–86.

Yang, B., Lei, Y., Liu, J. & Li, W. (2017). Social collaborative filtering by trust. *IEEE transactions on pattern analysis and machine intelligence*, *39*(8), 1633–1647.

Yao, L., Sheng, Q. Z., Ngu, A. H., Yu, J. & Segev, A. (2015). Unified collaborative and content-based web service recommendation. *IEEE Transactions on Services Computing*, *8*(3), 453–466.

Yao, L., Sheng, Q. Z., Wang, X., Zhang, W. E. & Qin, Y. (2018). Collaborative location recommendation by integrating multi-dimensional contextual information. *ACM Transactions on Internet Technology (TOIT)*, *18*(3), 32.

Yao, L., Wang, X., Sheng, Q. Z., Benatallah, B. & Huang, C. (2018). Mashup recommendation by regularizing matrix factorization with api co-invocations. *IEEE Transactions on Services Computing*.

Yu, C. & Huang, L. (2016). A web service qos prediction approach based on time- and location-aware collaborative filtering. *Service Oriented Computing and Applications*, *10*(2), 135–149.

Yu, X., Ren, X., Sun, Y., Gu, Q., Sturt, B., Khandelwal, U., … Han, J. (2014). Personalized entity recommendation: A heterogeneous information network approach. In *Proceedings of the 7th acm international conference on web search and data mining* (pp. 283–292).

Yu, X., Ren, X., Sun, Y., Sturt, B., Khandelwal, U., Gu, Q., ... Han, J. (2013). Recommendation in heterogeneous information networks with implicit user feedback. In *Proceedings of the 7th acm conference on recommender systems* (pp. 347–350).

Zhang, S., Wang, W., Ford, J., Makedon, F. & Pearlman, J. (2005). Using singular value decomposition approximation for collaborative filtering. In *Seventh ieee international conference on e-commerce technology (cec'05)* (pp. 257–264).

Zhang, Y., Chen, M., Mao, S., Hu, L. & Leung, V. C. (2014). Cap: Community activity prediction based on big data analysis. *Ieee Network*, *28*(4), 52–57.

Zhao, Z.-D. & Shang, M.-S. (2010). User-based collaborative-filtering recommendation algorithms on hadoop. In *2010 third international conference on knowledge discovery and data mining* (pp. 478–481).

Zheng, J., Liu, J., Shi, C., Zhuang, F., Li, J. & Wu, B. (2017). Recommendation in heterogeneous information network via dual similarity regularization. *International Journal of Data Science and Analytics*, *3*(1), 35–48.

Zheng, Z., Ma, H., Lyu, M. R. & King, I. (2010). Qos-aware web service recommendation by collaborative filtering. *IEEE Transactions on services computing*, *4*(2), 140–152.

Zheng, Z., Ma, H., Lyu, M. R. & King, I. (2012). Collaborative web service qos prediction via neighborhood integrated matrix factorization. *IEEE Transactions on Services Computing*, *6*(3), 289–299.

Zheng, Z., Ma, H., Lyu, M. R. & King, I. (2013). Collaborative web service qos prediction via neighborhood integrated matrix factorization. *IEEE Transactions on Services Computing*, *6*(3), 289–299.

Zheng, Z., Zhang, Y. & Lyu, M. R. (2012). Investigating qos of real-world web services. *IEEE transactions on services computing*, *7*(1), 32–39.

Zhong, J. & Li, X. (2010). Unified collaborative filtering model based on combination of latent features. *Expert Systems with Applications*, *37*(8), 5666–5672.

Zhong, Y., Fan, Y., Huang, K., Tan, W. & Zhang, J. (2015). Time-aware service recommendation for mashup creation. *IEEE Transactions on Services Computing*, *8*(3), 356–368.

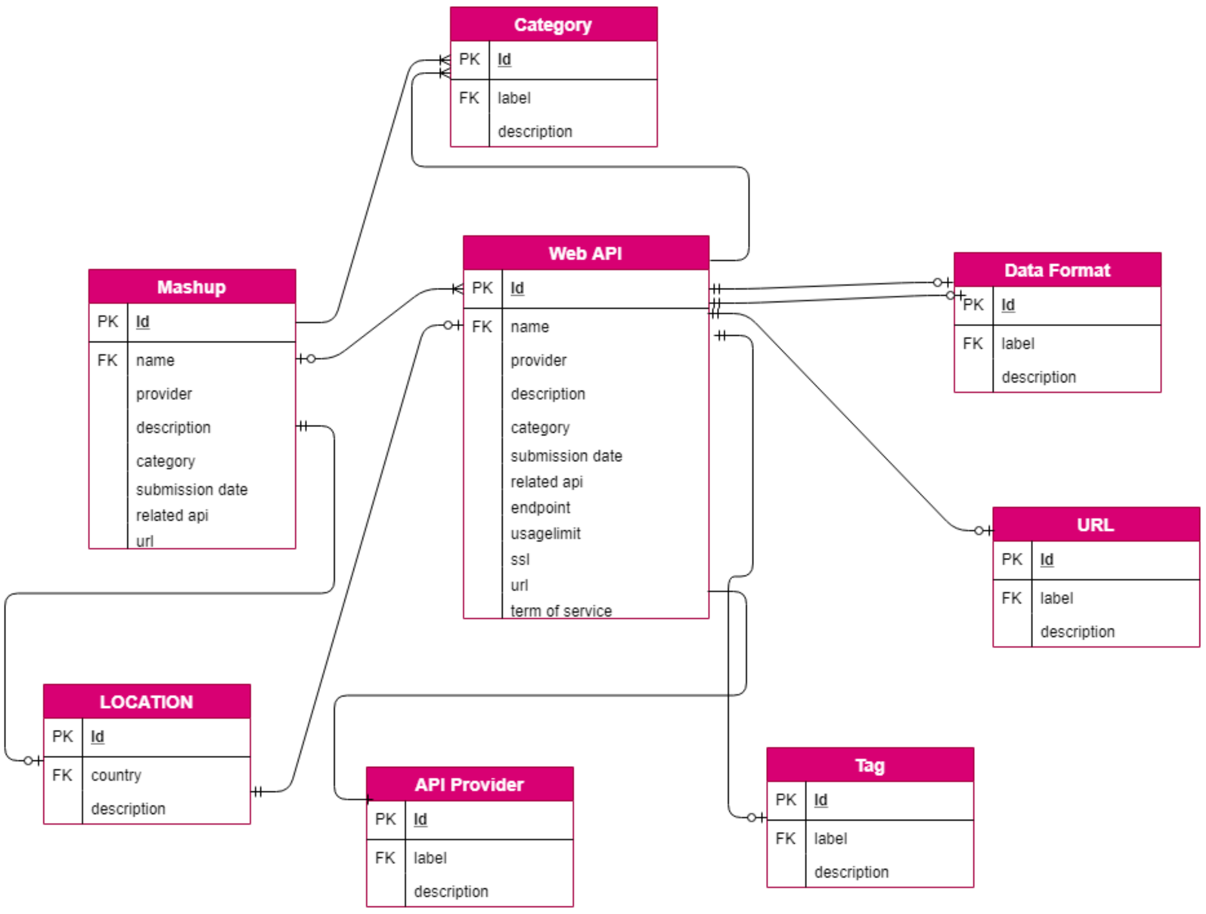
Zhou, Z., Wang, B., Guo, J. & Pan, J. (2015). Qos-aware web service recommendation using collaborative filtering with pgraph. In *2015 ieee international conference on web services* (pp. 392–399).

Zhu, X., Ye, H. & Gong, S. (2009). A personalized recommendation system combining case-based reasoning and user-based collaborative filtering. In *2009 chinese control and decision conference* (pp. 4026–4028).

Zou, B., Li, C., Tan, L. & Chen, H. (2015). Gputensor: Efficient tensor factorization for context-aware recommendations. *Information Sciences*, *299*, 159–177.

Appendix A

Database schema


Category
PK Id
FK label
description
Web API
PK Id
FK name
provider
description
category
submission date
related api
endpoint
usagelimit
ssl
url
term of service
Mashup
PK Id
FK name
provider
description
category
submission date
related api
url
Data Format
PK Id
FK label
description
URL
PK Id
FK label
description
LOCATION
PK Id
FK country
description
API Provider
PK Id
FK label
description
Tag
PK Id
FK label
description

Appendix B

Project Source Code

The database scripts and source codes used in this research are available on this link

https://drive.google.com/drive/folders/1SsEAid5-Zm4X6oXLivJzudlhQPHI1wWa?usp=sharing