HIGH-ORDER AND MOTIF-BASED GRAPH EMBEDDING IN SERVICE RECOMMENDATION

A THESIS SUBMITTED TO AUCKLAND UNIVERSITY OF TECHNOLOGY
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Supervisor

Associate Prof. Jian Yu

Associate Prof. Quan Bai

December 2021

By

Thi Thuy Mo Nguyen

School of Engineering, Computer and Mathematical Sciences

Abstract

With the rapid increase of the number of diverse services and APIs on the Internet and the Web, recommender systems have become an indispensable tool for service discovery and selection. Existing probabilistic matrix factorization (PMF) recommender systems can effectively exploit the latent features of the invocations with the same weight but not all features are equally significant and predictive, and the useless features may bring noises to the model. Currently, Deep Neural Networks (DNN) based collaborative filtering is a popular method for service recommendation. However, many such approaches treat each mashup-API invocation as separate instances and overlook the intrinsic relationships among data. We have finished three major pieces of work as follows.

First, we propose the Attentional PMF Model (AMF), which leverages a neural attentional network to learn the significance of latent features. We then inject the attentional scores and the mashup-API context similarity, and API co-invocation into the matrix factorization structure for training. Particularly, the we first apply an attentional mechanism with the neural network to the PMF-based framework. Later on, we then use the Doc2Vec technique to explore the latent features from document context of the mashups and APIs' description and do some statistical distribution fitting to draw out the APIs co-invocation pattern. Such auxiliary information significantly regularize the learning model for better prediction.

Second, inspired by the discovery that the autoencoder architecture can force the hidden representation to capture information on the structure of networked data, we

propose a novel framework called *Data Augmented High-order Graph Autoencoder* (DHGA) that learns the latent high-order connectivity signals in the mashup-API invocation graph for service recommendation. Specifically, each mashup-API pair is augmented with their higher-order neighbourhood data, and such data is input into two sets of autoencoders, one set for the mashups and the other for the APIs. All the autoencoders in one set shared parameters, so increasing the number of autoencoders does not increase the model size.

Finally, we propose a Motif-based Graph Convolution Layers with self-attention using attention motif-based neighbour mechanism to capture the high-order structure , and a Motif-based Graph Attention Collaborative Filtering model with MLP to generate the recommendation prediction. Specifically, we analysis some most common used sub-graphs or motifs in the mashup-API bipartite and apply the motif-based attentional mechanism for each motif type and put them in a convolution neural layers. Then, a dot product is used to calculate the invocation prediction value between a pair of mashup and API.

In summary, we have explored three recommendation models for service composition including AMF, DHGA, and MGAT. All of them successfully explore the latent features of mashups and APIs in different sides of the data to support for the Collaborative Filtering Neural Network recommendation model. While the AMF focus on the context and co-invocation auxiliary information in a shallow framework, DHGA and MGAT pay attention on the intrinsic relation of the data structure through using high-order and motif-based connectivity with deep learning neural network layers. All of the proposed models obtain superior performance on service recommendation accuracy compared with existing defined baselines in the thesis.

Contents

List of Tables

List of Figures

Attestation of Authorship

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

Signature of candidate

Publications

Delete and replace the text in this list

Nguyen, M., Yu, J., Nguyen, T., Han, Y. (2021). Attentional matrix factorization with context and co-invocation for service recommendation. Expert Systems with Applications, 186, 115698.

Nguyen, M., Yu, J., Bai, Q., Yongchareon, S., Han, Y. (2020, February). Attentional matrix factorization with document-context awareness and implicit api relationship for service recommendation. In Proceedings of the Australasian Computer Science Week Multiconference (pp. 1-10).

Nguyen, M., Yu, J., Nguyen, T., Yongchareon, S. (2021). High-order autoencoder with data augmentation for collaborative filtering. Knowledge-Based Systems, 107773.

Nguyen, M., Yu, J., Nguyen, T., Han, Y. (2021). AMF: Attentional Matrix Factorization with Context and Co-invocation for service recommendation [Source Code]. https://doi.org/10.24433/CO.3968216.v1

Acknowledgements

- I would like to express my thankful to my primary supervisor Associate Professor Jian Yu, who has guided, encouraged, and motivated my research with excessive enthusiasm over the last three years. It was an notable experience to have such a enthusiastic supervisor. He has enabled me to reach this point of my PhD journey with his constant instruction and tireless support.
- This work would not be possible without the support of AUT Scholarship who provided me the entire Doctoral Fee during my PhD study.
- I am thankful to my secondary supervisor Associate Professor Quan Bai for the always-available support. I am also thankful to my co-authors, Doctor Tung Nguyen for his contribution in various portions of my research.
- I thank all my colleagues in WT 406 Research Lab and the Software Engineering Research Lab, especially Khavee Agustus Botangen, Olayinka Adeleye, and Alan Zhang for the friendship and snap chats.
- Finally, I thank my family for supporting me throughout my PhD journey. I am especially grateful to my husband Duong Vu for being always there as a source of inspiration, and to my parents, Tuyen and Thuong, for their incredible support and sacrifice.

Dedication

I dedicate this thesis to my wonderful children An and Dan. You have made me stronger and sharpener to achieve accomplishments on my way to PhD. I love both of you to the moon and back.

Chapter 1

Introduction

Service computing and its popular application called Web services or Web API help software engineers to enhanced the way of designing, developing and maintaining enterprise applications (Maamar, Hacid & Huhns, 2011). With service computing, software development had a paradigm shift and it revolutionized the information system development processes, which are transformed from component-based to service-based (W. Xu, Cao, Hu, Wang & Li, 2013). Such service-based architecture reduces requirement of time, cost and effort to create service-based systems and enhance re-usability (Q. He, Yan, Jin & Yang, 2014). The key evolution of this improvement is the Web API, which is identified as loosely-coupled, self-contained, self-describing, Web accessible, modular programming functions that can be published, discovered and invoked across the Web (Rostami, Kheirkhah & Jalali, 2013; Papazoglou & Van Den Heuvel, 2007).

Specifically, Web APIs or API (in short) have been increasingly developed and widely used for Web and mobile applications over the past years (Bouguettaya et al., 2017; Sheng et al., 2014). For instance, programmers in different application domains are able to integrate maps into their products in several coding lines with Google Maps API. ProgrammableWeb.com has become the largest online Web API registry with extensively-cited sources of data such as mashup and API descriptions

and invocations. However, the accelerated growth in the quantity of APIs has brought difficulties for programmers to choose the right APIs from a large number of candidates. Moreover, sparse API invocation data also brought challenges to API recommendation systems (Kim, Park, Oh & Yu, 2017). Hence, it is highly demanded to develop better recommendation techniques for programmers to pinpoint the proper APIs for mashup development.

Such state-of-art software development technique applies the service-oriented computing (SOC) archetype (Dustdar & Schreiner, 2005; Papazoglou & Van Den Heuvel, 2007; Bouguettaya, Sheng & Daniel, 2014; Lemos, Daniel & Benatallah, 2015) to build the service-based software applications. The key idea of service computing is recursively integrating current (atomic) APIs to make a value-added API, which is called composite service or business process, to obtain a business objective that cannot be done by a single API. Current service computing approaches are developed on the conventional Web service techniques such as an enterprise system composed of Web APIs in which each constituent Web API enclosed a entire business functionality.

1.1 Concepts in service computing

The basic concepts and issues surrounding the typical Web service composition or mashup are described as below (Sheng et al., 2014):

- Orchestration and choreography: (Peltz, 2003) such concepts consider the control of integration and coordination of Web services that make up a business process. The orchestration uses a centralised prospect where there is a single API coordinating the interaction among the involved services. WS-BPEL1 (Web Services Business Process Execution Language) is a common standard for Web services orchestration. The choreography uses the decentralised framework allows each component Web API to demonstrate its interaction role more collaboratively.

WS-CDL2 (Web Services Choreography Description Language) is a standard specification for Web services choreography.

- Static and dynamic composition: static (composed APIs are determined at design time) and dynamic (composed APIs are determined at runtime) composition. (Dustdar & Schreiner, 2005) considers the time when the Web APIs are integrated. In a static composition, component APIs are selected and bound at design time. This would be fine when business stakeholders and service functionalities or composition requirements are fixed (Sheng et al., 2014). Reversely, dynamic composition implements the selection and binding of component APIs at runtime. This is convenient for a volatile operational environment with frequent runtime changes in requirements or services, that requires an adaptable composite service.

- Manual and automatic composition: manual with user-driven and automatic with no-user intervention composition determine the manner of aggregating services. In the manual structure, an abstract composite process is created and manually chosen and secured with Web services. WS-BPEL is used to designated manual compositions. However, Web service composition includes a complex task and takes lot of human capability. One of such limitations of this complexity is the impresive increase in the quantity of available Web APIs across the Web and the demand to design on the fly based on the current situation, and the heterogeneity of Web services. The automatic composition generally targets for the invocation, automated discover, composition, selection and interoperation of Web APIs. Automatic API compositions make use of Artificial Intelligence (AI) planning (Rao & Su, 2004), semantic Web (McIlraith, Son & Zeng, 2001), and dynamic workflow (Casati & Shan, 2001) approaches. OWL-S4 is a standardisation example to facilitate automatic service composition. It is a Web Ontology Language (OWL)-based Web service ontology which determines a core set of constructs to describe

the properties and capabilities of Web APIs in clear, computer-based interpretive form. The hybrid semi-automatic composition embellishes the manual method with some automatic approaches. A conventional example is the model-driven service composition (Gronmo, Skogan, Solheim & Oldevik, 2004), which automatically build a service composition specification from the abstract composition models that demonstrate the workflow, requirements, and component services.

The growth of technology (Perera, Zaslavsky, Christen & Georgakopoulos, 2013) expands the scope of service to concepts like "everything as a service (Duan et al., 2015), which connect daily activities, people, and devices to the internet. The internet of things provide us a lot of smart applications such as smart-appliances, smart-TVs, smartphones, robots, smart public displays, and traffic sensors. To be accessible from the Internet or an ad-hoc network, such applications are connected through standard protocols. Because every "thing" in the world could be encased and identified as a service to distribute their usage, for instance, attaching heterogeneous physical objects on the conventional Web APIs (Meyer, Ruppen & Magerkurth, 2013; Guinard, Trifa & Wilde, 2010; S. N. Han & Crespi, 2017), a composition of diverse smart applications and Web APIs can be executed to satisfy even the most complicated user objective (Ko, Ko, Molina & Kwon, 2016).

Service composition is propulsive applications. Both hardware and software resources attached components of software systems to become dynamically, interoperable, loosely coupled, standardized, and reusable. Service composition releases into air the online, service-enable business, which create motivation for seamless delivery of APIs through the internet with lowest cost but high potential market. Similarly, service composition launches channels for alliances, allows functionality outsources, and supplies a one-stop shop for business clients. The enormous growth in the number of Web APIs and the rising of IoT and cloud services (Miorandi, Sicari, De Pellegrini

& Chlamtac, 2012; Wei & Blake, 2010; Tsai, Sun & Balasooriya, 2010) has strongly driven the service composition to become a promising development resolution for contemporary software systems. For instance, a survey in 2008 explored 5,077 conventional (WSDL-described) Web APIs (Al-Masri & Mahmoud, 2008), a survey in 2013 explored 6,686 cloud APIs (Noor, Sheng, Ngu & Dustdar, 2014) and, by the April of 2019, the number of Web APIs exceeded 21,600 in one of the largest service repository. An emerging service Web (Tselentis, Domingue & Galis, 2009) – "a web that allows billions of parties to expose and consume billions of services", drives service composition to be a potential efficient, on-demand up-to-date development methods. It is because a component service can be carried out as soon as it is demanded.

Web service discovery is defined as the process of choosing right set of APIs which promisingly fulfil particular user's requirements. The key significance of this process is to capture the API textual description that matches the API requester query from central or a database record of service description. Following the discovery of set of candidate APIs is the selection process, which chooses the most appropriate APIs for the service consumer's request for recommendation list. Figure 1.1 demonstrate the key components of a Web API discovery process. Service consumer launches the API discovery process by systematically specifying a API request in term of query to identify appropriate candidate APIs in the Web API repository. If the chosen API is already known earlier, the requester directly secures the API, otherwise, a service broker (usually arranges the API registry where the APIs are published) might be engaged in the invocation step. After discovering, securing, invoking, and executing, the consumer uses the API request under a keyword-based description of service requirements which might contain functional and non-functional capabilities of Web API. Service producers issue the APIs' functional descriptions on the advertisement via a service broker on the repository. This process also includes a matching engines attached on the Web service repository to secure the service request of consumers and update them with the available

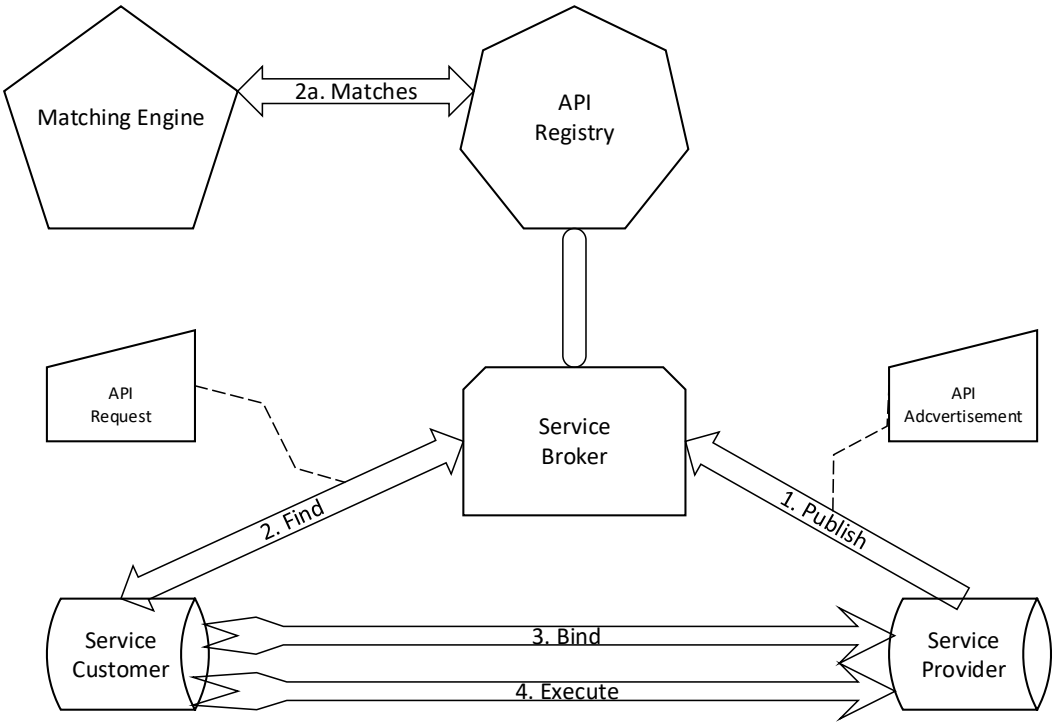

Figure (1.1) Service Discovery Process

API advertisement records in the repository.

1.2 Service recommendation

Service discovery is an important prerequisite to apply service technologies. It uses service discovery algorithms to search services from the service registry according to the users' functional and nonfunctional needs and constraints. Service recommendation is a proactive service-discovery technique, which proactively recommends services to users according to users' preferences.

Service computing build blocks from APIs to create the low-cost and appropriate software applications. The rapid development of cloud computing and SOC has increased the quantity of Web APIs published on the internet. Specifically, there are over 16000 Web APIs have been published at Programmableweb before the first of January 2017. This number generally rises triple for three years. Big IT companies such as Amazon and Microsoft have found many Web API marketplaces with lots of published Web APIs. Because many of users often are not satisfied with a unique API, an API composition or mashup is built from couples of such single APIs.

The rapid growth of web API brings both advantages and difficulties to SOC. Service recommendation has been widely applied in academic environment and industry. Currently, many proposed works have been presented to services recommendation which use collaborative filtering-based framework (Adomavicius & Tuzhilin, 2005; M.-H. Park, Hong & Cho, 2007; Roh, Oh & Han, 2003; L. Gao & Li, 2008; J. Zhong & Li, 2010), context-aware approaches (Y. Xu, Yin, Deng, Xiong & Huang, 2016; Finkelstein & Savigni, 2001), the graph-based approaches (T. N. Kipf & Welling, 2016a; X. Wang, He & Chua, 2020a), etc. In the area of service recommendation, the process of determining and building a combination of available APIs to propose an optimized mashup that satisfy the user's interest given by previous service composition is called composite service-oriented recommendation (CSOR).

Figure 1.2 demonstrates an overview concept of CSOR. Particularly, the composite

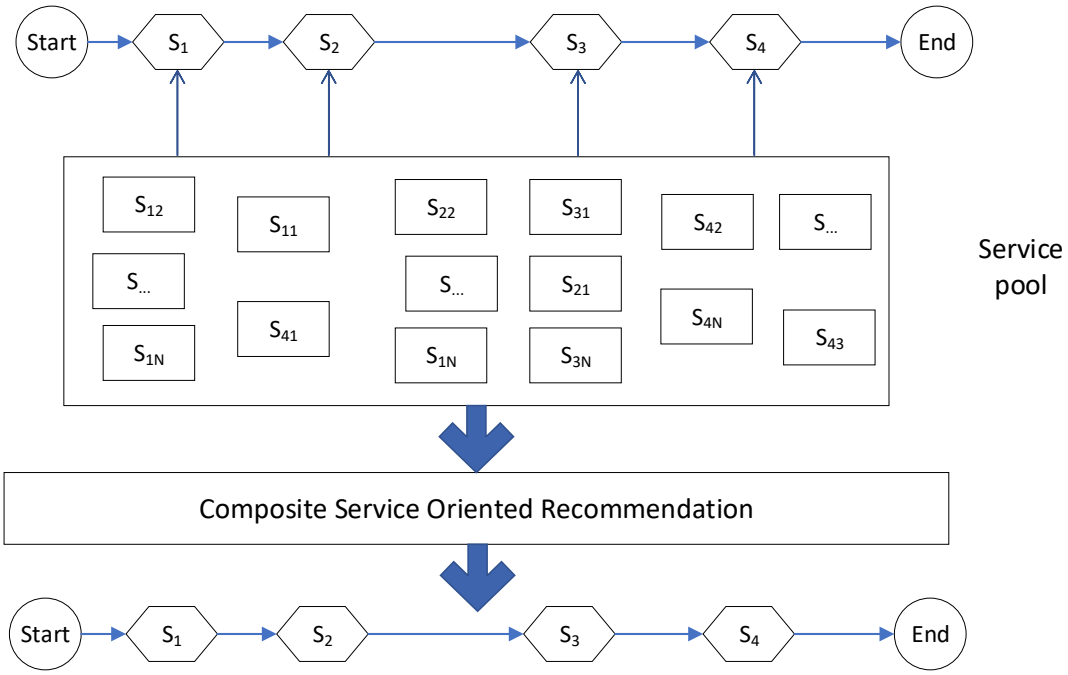

Figure (1.2) Overview of composite service-oriented recommendation.

service or mashup is built for several services or APIs which satisfy sub-functionalities in the service pool. Obviously, the aim of CSRO is to compose some candidates APIs which are similar or equal to the atomic services in the mashup for optimized composite service recommendation.

Currently, matrix factorization based collaborative filtering models have been widely employed in Web service recommendation (Jain, Liu & Yu, 2015; J. Liu, Tang, Zheng, Liu & Lyu, 2015; Samanta & Liu, 2017; Tian, Wang, He, Sun & Tian, 2017; Zheng, Ma, Lyu & King, 2012). However, the large number of Web APIs makes bring difficulties to manually choose relevant APIs to satisfy complicated user demand. Therefore, it is essential to select appropriate Web APIs that is suitable for user requirement. The process of discovering relevant APIs is defined in Web API recommender system. Recently, matrix factorization is popular framework applied in Collaborative Filtering based models in Web API recommender system (Jain et al., 2015; J. Liu et al., 2015; Samanta & Liu, 2017; Tian et al., 2017; Zheng et al., 2012), which can predict the list

of Web APIs for mashup construction by exploiting existing invocation data.

Nevertheless, matrix factorization learns the latent factors by a linear model and is thus difficult to contain the intrinsic features of the mashups and APIs invocation when the invocation matrix is highly sparse (X. He et al., 2017a). The statistical data of the largest Web API repository ProgrammableWeb shows the extreme sparsity of the invocation matrix at about 99.83%. Such extreme sparsity brings more intractable issue to accurately learn the complicated mashup-API relations. Existing deep learning approaches have been successfully applied in recommender systems (S. Zhang, Yao, Sun & Tay, 2019) and show their powerful abilities to learn the complex representation. They learn hidden structures from the invocation of mashups and APIs.

Existing work on Deep learning based collaborative filtering technologies are (S. Zhang et al., 2019) (H. Guo, Tang, Ye, Li & He, 2017) (X. He et al., 2017a) (Paradarami, Bastian & Wightman, 2017) (Xue, Dai, Zhang, Huang & Chen, 2017). Specifically, (Xiong, Wang, Zhang & Ma, 2018) proposes a novel deep hybrid collaborative filtering approach for service recommender system (DHSR) to capture the complex invocation relation between mashups and services. However, these work stop at the direct relation and hence ignore the intrinsic relation in the graph structure, which would be potential to contain rich of useful information.

1.3 Recommender system based on graph embedding

By applying graph embedding on graph representations of services, we can learn the embeddings of services and use them to calculate the similarly between pairs of mashup and API. Later on, their probability of invocation can be obtained by their according similarity.

A graph embedding-based recommendation have superior performance compared with traditional recommender systems. Particularly, after allocating information into

graph representations, traditional recommender system learn the parameters by analyzing the graph topological features such as users' co-interactions with frequent used items (Sarwar, Karypis, Konstan & Riedl, 2001a) or global topological diffusion (Y.-C. Zhang, Blattner & Yu, 2007; Y.-C. Zhang, Medo et al., 2007). Differently, the graph embedding-based recommendation learn the embedding vectors and secure graph topological features by embedding techniques (Goyal & Ferrara, 2018). To do the analysis of graph topological features, some work utilize subgraphs (Zhao et al., 2021), motifs (Lim & Lee, 2016; S. Zhang, Hu, Subramonian & Sun, 2020), and neighborhood [68–70] to extract the features embedding [39] and perform the recommendation. Recent works on graph embedding for recommendation (Sun et al., 2019; Y. Gao, Li, Lin, Gao & Khan, 2020; Q. Guo et al., 2020) have demonstrated the success on such area. However, such model has not been applied in service computing.

1.4 Research questions

Social Web service recommender system requires techniques and recommender models which can capture the attributes of Web APIs and effectively predict the suitable list of Web APIs for mashup developing. This thesis focuses on solving three primary challenges, which influence the prediction performance of Web service recommender systems for creating mashups.

1. How to take advantages of the document context awareness and co-invocation in an Attentional PMF model for service recommendation. The question aims to discover the methods to increase the prediction for mashup-API invocation prediction from the information in the document context and historical co-invocation data. Some detail concerns can be considered consisting of (1) What NLP techniques can be used for learning the latent features from document context; (2) What is the attention mechanism used and how to attach it to the PMF model; (3) How can the co-invocation data be built from the dataset?

2. How to work with the High-order connectivity and embed such information to the service recommender system. The direct invocation is sparse and hence not sufficient for a model to generate good prediction. High-order connectivity is considered as potential source of data in which more relevant information can be extracted for learning the service embedding features. Autoencoder is known as a good technique to learn the hidden features in graph structure. How to apply autoencoder with high-order connectivity is a significant question needed for the improvement of the service recommendation.

3. How do the Motif-based structures in bipartite network contribute to the service recommender system performance. Motifs or subgraphs present for pattern relationship among mashups and Web APIs, which is believed to contain rich information for a future invocation prediction. The question is how to discover such features

and apply it with a deep graph neural network. Particularly, (1) How to figure out relevant types of motifs which can significantly contribute to the model performance?; (2) How to exploit the graph structure from a deep neural network?; (3) How to weight the motifs' types in such model?; and (4) How much significance is the motifs based features add to service recommendation?

1.5 Methodology and Research objective

We address the research questions in Section 1.4 and present in chapters of this thesis. More specifically, Chapter 3 solves the RQ1, Chapter 4 deals with RQ2, and Chapter 5 demonstrate the proposal for RQ3. Following the instruction of engineering research (Wieringa, 2005), we address the research questions in three steps including problem analysis, solution analysis, and implementation analysis. In the problem analysis phase, we explore and figures out problem's details which have been clarified in the research question. This step is implemented by investigating related works in the literature review and setting of research objectives. After that, in the solution analysis step, we propose solution to the defined problem. We performs the attributes investigation, solution sensitivity analysis, and the evaluation of the ability of the proposal approaches for the problem. Finally, in the implementation analysis step, we implement the proposed approaches and do further experiments with case studies. Table 1.1 shows the flow of principle-based method of systems analysis approach.

1.6 Contributions

This thesis intend to contribute towards the feature embeddings for the service recommendation Collaborative Filtering framework introduced in Section 1.2. Addressing each research question has created techniques that can fill the existing limitation of the

existing approaches. Specifically, the thesis has major contributions as below:

- An Attentional Matrix Factorization model with the document context awareness and API co-invocation.
- A data augmentation technique with high-order connectivity in autoencoder frameworks to explore the intrinsic relation of the bipartite network.
- A motif-based graph convolution self-attention in a CF-based framework to attach the high order connectivity with variant patterns of motif-based neighbour sub-graphs.

1.7 Thesis Structures

The remainder of the thesis is structured as follows:

- Chapter 2 demonstrates a comprehensive survey of the state of art. The first section presents fundamental concepts in service computing including mashup and Web service discussion. After that, existing work on context awareness and graph representation learning are studied and reviewed, which would be used as auxiliary information for service recommendation. Service recommendation approaches such as functionality-based, social network-based, Collaborative Filtering, and hybrid methods are investigated with the current studies in this area. The limitation of mentioned work in this chapter is the motivation for the research questions which are addressed in the following chapters.
- Chapter 3 is the first proposed approach using Attentional Matrix Factorization with Context and Co-invocation for service recommendation. The chapter presents some related works and studies the significance of document context and the compatibility of Web APIs invoked by the same mashups. The chapter

also presents the latest baselines related to the approach's techniques, which use attention factors, context awareness, and co-invocation. Some evaluation metrics are defined for the comparison. The chapter is based on the journal " Attentional matrix factorization with context and co-invocation for service recommendation" published in Mo Nguyen (2021) and the conference ASW2020. The code of this work is published in CodeOcean. However, the proposed approach in this chapter is limited in the first order interactions of the network and has not explore the potential values of the high-order connectivity which will be investigated in Chapter 4.

- Chapter 4 presents another approach for service recommendation using Data Augmented High-order connectivity with the graph autoencoder technique. The chapter reviews some recent work on CF-based methods and neural networks for learning the feature embedding within the data augmentation framework. Then, the chapter defines some definitions on high-order mashup-API invocation graph, and the data augmentation framework. The proposed detail model is demonstrated with the combination of autoencoder embedding layers and the high-order relation framework. The model is evaluated by extensive experiments with related baselines. The chapter is written based on the journal published in Mo Nguyen, Jian Yu (2021). Even though the chapter demonstrates the success of high-order connectivity and auto-encoder embedding technique, such approach does not aware the power of subgraphs which are believed to contain lots of valuable latent features for prediction model. We shed light on this challenge in Chapter 5.

- Chapter 5 presents the proposed model for Web service recommendation using Motif-based Graph Attentional Neural Network. The chapter introduce definitions about the types of motifs in the bipartite graph. Later, the chapter presents

the framework of self-attention for convolution layer with motif-based graph, and the integration of such method into the Collaborative Filtering based recommender system. The chapter also reviews related papers using network motifs for recommendation models.

- Chapter 6 concludes the thesis and demonstrates some challenges that figure out future research directions.

Table (1.1) Principle-Based Method of Systems Analysis Method

Research questions	**Objectives**
Research question 1	Investigate the recommender systems approaches applied in service recommendation - Investigate the existing repositories of Web services - Examine the impact of context (i.e., geographic location) on both invocation behaviour and QoS of real-world services - Develop a service recommendation model that considers the context of both composition and potential component service - Evaluate the performance of the recommendation model over a real-world service invocation data
Research question 2	Use autoencoder in high-order connectivity to learn the embedding of mashups and APIs. - Study the autoencoder framework on existing approaches for features embedding. - Study the data augmentation technique in service recommendation. - Design an autoencoder architecture for bipartite network . - Develop a data augmentation with high-order connectivity Collaborative Filtering model for service recommendation. - Evaluate the performance of the proposed model over a real Web API dataset.
Research question 3	Explore the graph structural data through Motif-based Graph Convolution Attention framework to learn the embedding for a MLP Collaborative Filtering recommender system. - Examine the variants of motifs of sub-graph in the bipartite network. - Build a Motif-based Graph Convolution Self-attention layers for embedding learning. - Develop the Motif-based Graph Attention Collaborative Filtering model. - Evaluate the performance of the proposed model.

Chapter 2

Literature Review

This chapter presents all-inclusive review of concepts, algorithm, techniques, and frameworks which create the fundamental of this thesis. The thesis is the combination of service computing, context variability, graph embedding, and recommender system. In the service computing, the chapter presents the general concepts, definitions, and related frameworks in the four mentioned research streams, which provide the context for later chapters of the thesis. In addition, the chapter also reviews the existing works relating to the research questions defined in Chapter 1. Particularly, Section 2.1 introduces of mashup and API in service computing. Section 2.2 presents the context awareness requirement and existing relevant work in service computing. Section 2.3 provides detail discussion about the graph representation learning in graph data structure. Section 2.4 presents the current recommender system which have been implemented in service computing.

2.1 Fundamental concepts in service computing

2.1.1 Software-as-a-service

SOC develops the idea of providing software as-a-service to end-users. The concept of central management and offering a standard application package occurred in the 1960s, when the goal was to address the increasing needs for IT software but lack of person capacity (J.-N. Lee, Huynh, Kwok & Pi, 2003). Programming contract is the most common form of outsource during that time. Since 1990, the internet has exploded and the Application Service Provider (ASP) model built up software-as-a-service. The ASP is a vendor or third party who owns a software application offered to the clients or end-users. The application in the vendor's system is deployed, managed, and hosted by the vendor who is consider as a central provider. The clients within the network use a subscription or rental basis to access the application remotely. Generally, companies might outsource their information technology demands, while the applications, the connected infrastructure, and the data of clients are operated and maintained by the ASPs. However, the built-in drawbacks of the ASP model come from applications with monolithic framework, and with extremely fragile, customer-specific, and tightly-coupled components (Papazoglou, 2003). Some different acronyms and related business models were built such as AIP (Application Infrastructure Provider), IBS (Internet Business Service), BSP (Business Service Provider), and SSP (Solution Service Provider). In 2000, the Software Information Industry Association created the term SaaS (Software-as-a-Service) in order to consolidate the variant terminologies used in software services. Big companies such as Salesforce.com, Oracle, and SAP have attempted to launch the Saas model. At the same time, the SOC paradigm has been changed the SaaS to a loosely-coupled interaction among Internet applications.

2.1.2 Web service oriented architecture

Service-Oriented Architecture (SOA) (MacKenzie et al., 2006) (Papazoglou & Van Den Heuvel, 2007) is the key implementing model of SOC to demonstrate and manage distributed capabilities of software applications, resources, and infrastructure in a set of loosely-coupled, standard-based, inter-operable, and location-transparent services. Building multiple applications in variant techniques, protocols, and platforms allows such application to communicate and interconnect, and become integration-ready services (Papazoglou, 2003). For example, a programmer uses any device with any operating system, any computing platform, or any programming language, to work with a SOA service to build her own application. To implement functionalities of both end-user applications and other services, SOA defines a logical method to provide the service to authorize a cost-effective design, development, and management of software systems (Papazoglou & Van Den Heuvel, 2007). It allows to explore and use up the remote and distributed services over the network via services published interfaces. Figure 2.1 demonstrates a typical SOA, which defines the roles of and interaction among the three key partners: the Web Service provider, the Web API registry, and the service customer. The interactions includes the publish, find, bind, and execute actions. The requester and the provider are both software agents who represent for entities such as people or organisations. The former requests an API while the latter provides a API. A service provider hosts as API and publishes the description of API into the API registry or repository. Then the service customer searches for an API description and bind it with the service provider to invoke an API.

The function of a service broker is defined by the SOA service registry (Papazoglou & Van Den Heuvel, 2007). Trusted software agents who motivate service providers to abide by laws and regulations or industrial governance, are service brokers. A service broker controls a groups of available service providers, whose additional information

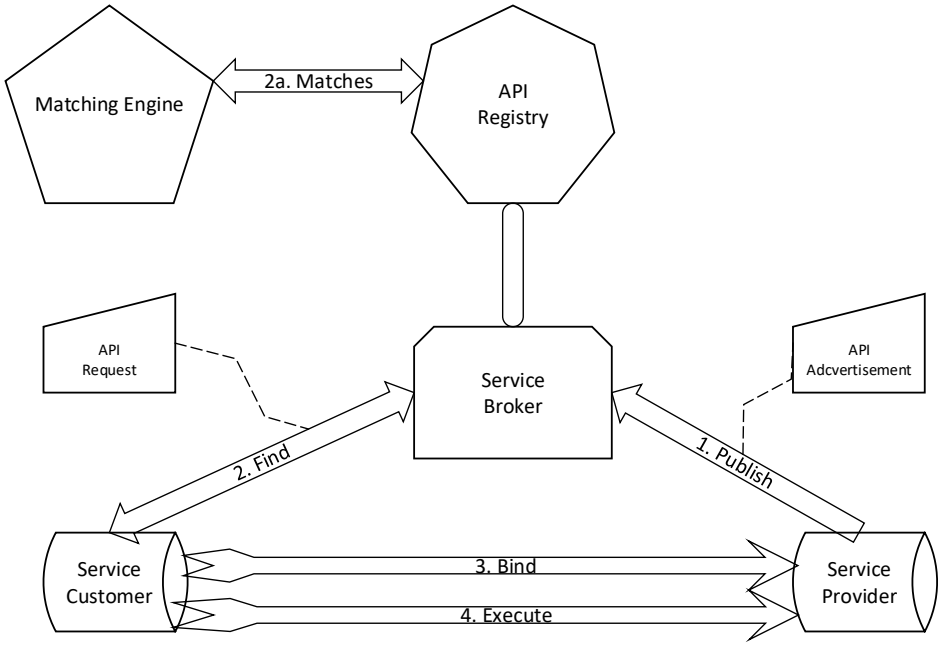

Figure (2.1) SOA

such as description, quality and service ratings can be added to the registry. However, the elementary SOA might not deal with some overall concerns for example such as service composition, service transaction, management, coordination, security, and management, which can be applied to the service architecture's components.

2.1.3 Mashup

A service is defined to have the working capability for another, or the offer to execute job for another (MacKenzie et al., 2006). It is known as a non-material equivalent of a good. Similar definition of SOA presented by (Sheng et al., 2014) is a "semantically well-defined abstraction of a set of computational or physical activities involving a number of resources, intended to fulfil a client need or a business requirement". Futhermore, a SOA service is basically a software in which a complete business functionality is encapsulated and can be invoked to satisfy a required functionality by another software system. In SOA, any business function from basic requirement to complicated business processes

are published as software resources and are packaged as services. Such services have below characteristics (Papazoglou & Van Den Heuvel, 2007; Channabasavaiah, Holley & Tuggle, 2003; Papazoglou, 2003).

- Self-contained – a service is well-defined, unique state, and context independent on other services. Similarly, a service in the client's system is designed irrespective of the type, the objective, and the usage circumstances.
- Platform-independent – a service is technology-neutral. It can be used without regarding the technical platform of the customer. Therefore, all elements including protocols, descriptions, and discovery mechanisms should follow the general established standards.
- Loosely-coupled – a service might not know the deployment details of the customers. Different customers can use for variant purposes. There are three main aspects of loosecoupling (Pautasso, Zimmermann & Leymann, 2008), including: (1) time/availability of the service consumers to collaborate with a service provider, (2) location aspect offers customers to explore the real location of service providers at run time, and (3) evolution aspect is the ability to make adjustments to a service without influencing its customers.
- Autonomous – a service is considered as a black box by exterior entities. For example, a customer hardly predicts the expected invocation result of a service, but it is needless to consider how the service executes its function.

2.1.4 Web service or Web API

The technologies and standards of Web service or Web API, have become a main facilitator for the SOA. Web API has been the key technology to carrying out the

conventional SOA objectives, for example sharing API, integrating just-in-time or on-demand, and interoperating among modern and legacy software resources (Bouguettaya et al., 2017; Sheng et al., 2014; Papazoglou & Van Den Heuvel, 2007). Standardisation organizations such as the World Wide Web Consortium (W3C) and the Organisation for the Advancement of Structured Information Standards (OASIS), have first implemented Web services by using specification and standardisation to smooth the interoperations. There have been various definitions in literature proposed for the term Web service from general to details. In the work (Sheng et al., 2014; Manes, 2001), Web service is loosely defined as "any unit of business, application, or system functionality accessible over the Web". This means that any applications containing a Uniform Resource Locator (URL) is a Web service. Further definitions on more technical and detailed are the following:

- (1) The Dagstuhl SOC working group (Ludwig & Petrie, 2006): defines Web serive as "a possibly remote procedure with an invocation that is described in a standard (preferably XML-based) machine-readable syntax reachable via standard Internet protocols with a description, including at a minimum the allowed input and output messages, as well as a possible semantic annotation of the service function and data meaning".

- (2) W3C (World Wide Web Consortium) defines Web service as "a software system identified by a Universal Resource Identifier (URI) designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialisation in conjunction with other Web-related standards".

Both definitions use the standard Internet protocols and Web service technologies to adjust towards the representation and integration of services. The conventional standard

Web services interaction protocols and technologies consists of SOAP3 (Simple Object Access Protocol) – an XML-based messaging protocol, REST (Representational State Transfer) (Fielding, 2000) – an architectural style for large-scale distributed hypermedia systems, OSGi4 (Open Services Gateway Initiative) – a specification to develop, deploy, and manage services for Java, WSDL5 (Web Services Description Language) – an XML-based specification for a Web service description, UDDI6 (Universal Description Discovery and Integration) – a standard universal business registry for indexing Web services, and ebXML7 (Electronic Business XML) – an initiative for a B2B XML framework of Web-based business services.

2.1.5 Discussion

Service computing is a archetype of service-oriented consideration and demanded co-operation, which obviously brings a new prospect for software engineering systems. Applying Web APIs technology, programmers can enclose whatever into a service or mashup. Web APIs are pervasive in nowadays livings and it makes conventional available APIs in society and economy such as education, healthcare, entertainment, and finance to be provided as online service. Such enhancements in online service technology bring the internet into a world wide workplace, which manages the individual and social events, a platform of an entertainment, and a API business platform (Bouguettaya et al., 2017). Normally, a single API might not satisfy with a request of functionality. Therefore, recommending a list of suitable APIs can reach user's goal better. Community of both service computing and software engineering consider service composition and service recommending as big challenges in such area (Sheng et al., 2014).

2.2 Context awareness variability

2.2.1 Context based requirement

The auxiliary dynamic information of service-based systems such as changes related with descriptions, bandwidth, available interfaces, locations, ambient conditions, heterogeneity of mashups, temporal and spatial conditions, and user activities, which are assigned to the context awareness should must be considered in requirements specification. Context is the reform of the environment which consists of anything that provides surroundings for a system to operate (Finkelstein & Savigni, 2001). Different definitions of context have delivered various interpretations. Differences might bring the lack of consensus in the domains and such limitation has being considered by researchers. This thesis emphasises on the definition of (Abowd et al., 1999), which defined "Context is any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves". Instead, (Henricksen, 2003) presents a service-oriented perspective for context: “The context of a task is the set of circumstances surrounding it that are potentially of relevance to its completion”. Henricksen differently contends that context is a vague concept which is hard to identify. Therefore, Henricksen proposes two related concepts including context models and context information, which are stated as: “A context model is a specification of identified concrete subset of the context that is realistically attainable from sensors, applications, and users and able to be exploited in the execution of the task”; and “A context information is a set of data, gathered from sensors and users, that conforms to a context model. This provides a snapshot that approximates the state, at a given point and time, of the subset of the context encompassed by the model”.

Furthermore, there are different taxonomies of context in literature including both

the general comprehensive classifications and the more detail ones. That demonstrates how different researchers identiy the context. (Perera et al., 2013) introduces and combines a few of such taxonomies, and advises to combine context taxonomies together to build a more all-inclusive taxonomy. In this thesis, we define the "context" as in (Krogstie, 2001), which contains the following types of context: information, spatio-temporal, task, personal, environment, and social. Such above contexts are appropriate for delighting, namely the flexibility and personalised tendencies of service-based systems. Identifying contexts in the requirements is important for variability of requirement. Mainly, context modelling focuses on representing context awareness in an infrastructure-centred view. Such perspective defines that the complexity of engineering context-aware applications can be mainly cut down simply by using an appropriate infrastructure to gather and manage the context information (Henricksen & Indulska, 2004). However, it is significant to specify the usage of applications which influences the variability in the requirement standard by grasping the context awareness.

Existing specialised RE methodologies integrating context-awareness have been presented. (Kolos-Mazuryk, Poulisse & van Eck, 2005) find that traditional RE methods have not explored the context-awareness, mobility, and personalisation, and hence propose an RE approach for service-based systems, which uses the attributes of Pervasive services. (Munoz, Valderas, Pelechano & Pastor, 2006) deal a similar definition that pervasive services require appropriate approaches to bring out and identify requirements. They leverage the ConcurTaskTree (CTT) method to form the functional requirements of intensive systems, specifically the action required. Their requirements framework forcefully hold the following elements: (1) physical environment's attributes such as a location model, (2) descriptions of actions such as a CTT model augmented with the geographical position where the invocation occurred, prerequisite to complete an action, interaction interface, and time-aware reliance), and (3) characterization of the

system's attitude. Such requirements model could be afterward connected to a domain-specific language for the pervasive systems' specification named Pervasive Modelling Language (PervML) (Muñoz & Pelechano, 2006) in a technology-independent conduct.

2.2.2 Context awareness in service computing

How to discover and select appropriate component APIs have become the most long-term challenge in service recommendation (Bouguettaya et al., 2017). The conventional discovery process mainly relies on matching the mashup's request to the description of available APIs in the repository. In the recommendation list, APIs with the best QoS are chosen. (Rong & Liu, 2010) shows that the success of such discovery process depends on both mashup's request and the matchmaking model, which can be classified into context-aware, agent-based, semantic, syntactic, and recommender system (Sheng et al., 2014; Rong & Liu, 2010; Afify, Moawad, Badr & Tolba, 2014). The conventional approach using keyword-based similarity is syntactic matching. This method matches the real words in the query with the description of APIs, which uses term analysis technique for example the Term Frequency-Inverse Document Frequency (TF-IDF) (Salton & Buckley, 1988). Semantic approach explores the intention of the queries and API' descriptions and form the semantic description from the domain concepts' ontology. After that, the semantic descriptions are attached in the matchmaking. To define the semantic descriptions, there are different standards such as OWLS (Martin et al., 2004), WSDL-S (Akkiraju et al., 2005), WSML (De Bruijn, Lausen, Polleres & Fensel, 2006), and DAML-S (Ankolekar et al., 2002). The context-aware approach studies the context of the services. In (Rong & Liu, 2010), the explicit context is provided directly by the users, and the implicit context is automatically or partly automatically drawn from the matchmaking mechanism. The other methods execute the discovery and recommend APIs' list which satisfy the functional requirements, technical requirements, and fiscal

need of users (L. Guo, Wang, Kang & Cao, 2015; Sim, 2011). For example, (Maximilien & Singh, 2004) broadens the typical agent-based SOA framework to an agent-based service model. In this work, using a QoS ontology and XML policy language in matchmaking, the agent performs as a broker of service consumers and agencies such as the augmented API repository. Inter-organisational systems use software agents to do the functional integration and improve business intelligence, and decide the collaborative (Shen, Hao, Wang, Li & Ghenniwa, 2007).

2.3 Graph representation learning

One of common used data structure is graph, which is applied widely in many fields relating to computer science such as social media, biological protein-protein networks, molecular graph architecture, and their recommender systems. Graphs grasp the relation or edges among nodes. In (Angles & Gutierrez, 2008), graphs are defined as the backbone of the countless systems that contain the accessible relational information of all interactions. Furthermore, graph contributes a pivot role in modern machine learning. There are many existing application in different fields using graph data structure to learn the embedding features to forecast and make recommendation such as forecasting people' roles in collaborative network, providing recommendation in social media, classifying the role of protein in the graph of biological relationship, and so on. However, how to embed the graph structure into the model using machine learning is still an undiscovered question. For instance, to predict the interactions in a social network, the pairwise properties of pairs of nodes are encoded. Another example is the classification work which attaches the global position information of the node such as the local neighbour graph. However, it is difficult to have a clear method to transform the high-dimensional, non-Euclidean values of the graph architecture into a latent feature vector. Conventional machine learning models use the graph statistical summary

such as clustering coefficients (Bhagat, Cormode & Muthukrishnan, 2011), kernel functions (Vishwanathan, Schraudolph, Kondor & Borgwardt, 2010) to extract the graph-based information, and embed the engineered features to calculate the neighbor's architecture (Liben-Nowell & Kleinberg, 2007). However, such engineered features are not flexible, specifically they might not adjust for the training and designing process of these features which causes high time-consuming and costly process.

Therefore, a number of approaches which learn the representation through encoding the graph structural information has increased. Such approaches use a mapping which embeds nodes, subgraph, or the whole graph into a lower dimensional vector space. The objective is to optimize the mapping and reflect the original graph-based structure. After that, they use the embeddings as the input for the later machine learning steps. The main difference between the previous approaches and the representation learning is the way of learning the graph-based embedding. While previous approaches consider it as a reprocessing process which uses hand-engineered statistics to draw out the structural information, the representation method conducts this problem as machine learning task that applies a data-driven framework to encode the graph-based information.

2.3.1 Generalized encoder-decoder architectures

Such above embedding approaches that we have reviewed in previous section are shallow models, which use an embedding lookup (Hamilton, Ying & Leskovec, 2017b). The limitation of shallow models is that the singular embedding vector for each node is trained independently. Therefore such models have some following disadvantages:

- 1. The parameters of the encoder are not shared among nodes, specifically a simple embedding lookup is used as the encoder. While using the shared parameters is a powerful regularization form, un-shared parameters framework increase to $\mathcal{O}(|V|)$ that is inefficient computation.

- 2. It is also difficult to use shallow embedding to extract the node attributes through the encoder especially on many large graphs which consist of attribute information such as user profiles on social media network.

- 3. The shallow embedding approaches can only apply for the nodes in the train set. They cannot obtain the embeddings for masked nodes if we do not execute more iterations of optimization. Therefore, embeddings from such approaches are consider as transductive (Hamilton, Ying & Leskovec, 2017a). This might bring disadvantages to evolving graphs, massive graphs which cannot be completed stored in places requiring new graph generalization after training phase such as memory and domains.

Existing modern approaches have shed light on these problems. Such methods still are in the encoder-decoder architecture but they apply more complex encoders under deep neural networks, that significantly exploit more attributes on the structure-based graph.

2.3.2 Neighborhood autoencoder methods

To address the above problems, remarkable approaches using deep neural network to learn the embedding are Deep Neural Graph Representations (DNGR) (S. Cao, Lu & Xu, 2016) and Structural Deep Network Embeddings (SDNE) (D. Wang, Cui & Zhu, 2016a). Particularly, these approaches use the deep neural network straight attach the graph structure into the encoder framework through a famous deep learning technique autoencoder (Hinton & Salakhutdinov, 2006) where the neighbour information of nodes is compressed and attached in.

Different from previous methods, instead of a pairwise decoder, DNGR and SDNE use a unary one. Accordingly, a neighbourhood vector $s_i \in R^{\mathcal{V}}$ is assigned to each node v_i, which is the row of v_i in the pairwise similarities matrix S. The s_i includes the

similarity between v_i and other nodes and it is calculated as a high-dimension embedding vector of v_i's neighbour. The approaches DNGR and SDNE aim to learn the node embedding by using s_i vectors which can be rebuilt later from such embeddings. DNGR and SDNE use the autoencoder with s_i vectors to reconstruct from such embeddings. The objective is to use a low dimension vector to compact the neighbour's information of the node. Both DNGR and SDNE have the autoencoder parts consisting of multiple neural network layers in which each encoder layer diminishes it input dimension while each decoder layer enlarges it dimension (Hinton & Salakhutdinov, 2006).

The difference between SDNE and DNGR is the similarity formulas used to construct the neighborhood vectors s_i and the optimization function of the autoencoder. Similar to DeepWalk and node2vec, DNGR calculate s_i by using the pointwise mutual information of two nodes co-existing on random walks. While SDNE uses the adjacency vector and using Laplacian eigenmaps to create the autoencoder objective (D. Wang et al., 2016a). Such encoder depends on the input s_i which consists of the local graph information so it allows SDNE and DNGR to integrate the structural features.

2.3.3 Neighborhood aggregation and convolution encoders

The purpose of existing approaches on node embedding is to address the main disadvantage of the shallow models and autoencoder methods by using the encoders which exploit the node's local neighborhood but not necessarily the whole graph. The insight of such approaches is that they create embeddings for a node by attaching the local neighborhood information. Different from the conventional methods, these neighborhood aggregation models depend on the features or attributes of nodes to learn the embeddings. For instance, a textual description in a social network or molecular markers in a protein-protein interaction network might have some relationship with each node. The neighborhood aggregation methods exploit such attribute information

to build their embeddings. If there is no such attribute data, a simple statistical graph is used instead such as node degrees (Hamilton et al., 2017a) and a one-hot indicator vector as an attribute (T. Kipf & Welling, 2016) (Schlichtkrull et al., 2018). Such type of approaches are convolutional because it uses the surrounding neighbor nodes to calculate the node embedding which is equivalent to the receptive field of a center-surround convolutional kernel in computer vision (T. N. Kipf & Welling, 2016b). The neighborhood based approaches learn the node representation in an iterative or recursive in the encoding phase. Firstly, the input node attributes are used to initialize the node embeddings. After that, at each iteration, nodes are attached with their neighbours' embedding by an aggregation function of a set of vectors. All nodes later are assigned with new embeddings which are equal to their neighbourhood vectors integrated with their previous embeddings in the latest iteration. In the final step, these embeddings are fed into a dense neural network layer. When repeating this process, the information reached from the graph is aggregated into the node embedding. However, the embedding dimension remains constrained when the process repeats. Therefore, the encoder encloses all neighborhood features into a vector with low dimension. After a certain loops, the iteration is terminated and output the final embedding vectors as the node representation.

2.3.4 Graph representation approaches

Early research (Chamberlain, Clough & Deisenroth, 2017) (Y. Li, Tarlow, Brockschmidt & Zemel, 2015) that apply neural network approaches on the graph data began with recursive frameworks which consider data as the directed acyclic graphs. Currently, deep learning has been successfully used to exploit sequence data in recursive neural network (RNN) (Chung & Graham, 1997; Perozzi, Kulkarni & Skiena, 2016) and grid-shaped data in CNNs (Atwood & Towsley, 2016). Other new approaches in a more

generic structural data such as NeuralFPS (S. Cao, Lu & Xu, 2015; Fortunato, 2010) which present tailored framework to a particular problem domain and used as an end-to-end differential deep model. Such approach produces the famous Weisfeiler-Lehman algorithm applied in molecular graphs.

On the other hand, some other approaches (De Oliveira & Levkowitz, 2003) use the spectral graph theory to define the graph convolution functions. Other approaches using spectral methods to replace principled-yet expensive graph convolutions such as (Bruna, Zaremba, Szlam & LeCun, 2013) which use Chebyshev polynomials to estimate a smooth filter and GCNs (Donnat, Zitnik, Hallac & Leskovec, 2018) use simple first-order filters to interpret the process. In (Donnat et al., 2018), Kipf and Welling present different types of graph-based tasks (Donnat et al., 2018; Hamilton et al., 2017a; Ou, Cui, Pei, Zhang & Zhu, 2016) and produce some variants such as as (A. Ahmed, Shervashidze, Narayanamurthy, Josifovski & Smola, 2013; Murphy, Weiss & Jordan, 2013). MCN (J. B. Lee et al., 2019) introduce a general version of GCN (Donnat et al., 2018). Such approach firstly use a weighted motif-induced adjacency matrix to enhance the neighbour features of nodes and then propose a novel attention framework to support the neighbours selection for better feature integration.

2.3.5 Higher-order connectivity with Network Motifs

The fundamental building blocks of complex network are defined as network motifs (Gilmer, Schoenholz, Riley, Vinyals & Dahl, 2017). Critical information about the function and structure of complex systems can be investigated from such patterns. A motif study (Hochreiter & Schmidhuber, 1997) in biological networks demonstrate that the vigorous attribute of robustness to perturbations highly correlated to the occurrence of particular motif patterns. Another work (Henderson et al., 2012) applying motifs in temporal networks state that graphs from different domains have very different

sub-graph structures which are considered as type of motifs.

A great number of existing research (Backstrom & Leskovec, 2011; Grover & Leskovec, 2016; Hoff, Raftery & Handcock, 2002; Paranjape, Benson & Leskovec, 2017) show the success of applying higher-order structures in variant graph-based machine learning models. DeepGL (Kearnes, McCloskey, Berndl, Pande & Riley, 2016) define motifs as a fundamental element to deeply learn the inductive relational functions which represent for relational operators' compositions in a base graph function for example triangle counts. In (Hoff et al., 2002), a notion of higher-order network embeddings is proposed, which show that attaching different motif-based matrix formulations can improve the accuracy of embeddings learning. Another work (Paranjape et al., 2017) propose a hierarchical motif convolution to the subgraph classification. Recently, (Grover & Leskovec, 2016) has demonstrated that standard GNN architectures and the 1-dimensional WL graph isomorphism heuristic have the same expressiveness. Therefore, such approaches have some similar disadvantages. They propose a generalization using higher-order structures to address the problem of graph classification.

2.4 Service recommendation systems

To address the challenges of selecting and discovering services, recommender system has been a common technique. Conventional Web API discovery from basic repositories are mainly used in early service recommender system and UDDI registries (Pastore, 2008) is a popular example. However, UDDI has been obsoleted because many large technology companies stopped using it such as Google, IBM, and SAP. Another techniques such as keywords-based and ontology-based query techniques are also used for Web APIs dicovery. However, they are still limited for the rapid developed Web API repositories (Broens, Pokraev, Van Sinderen, Koolwaaij & Costa, 2004; Yao, Sheng, Ngu, Yu & Segev, 2014). Therefore, researchers continue to investigate better solutions

for service recommender system.

2.4.1 Functionality-based Web service recommendation

Functionality-based Web service recommender systems user APIs' description to match request of users to recommend APIs. Some initial approaches use keyword-based API profile to match normally undergo low retrieval performance. Hence, researchers have used explicit semantics to enhance the matching service performance. Such models enrich semantics of APIs' descriptions by exploiting dictionaries and domain ontology, and then apply logic-based reasoning to calculate the semantic similarity. However, the disadvantages of defining ontology manually and annotating descriptions bring more difficulties when applying for a large scale service data.

Therefore, other approaches attach machine learning and data mining technique to functionality based service recommendation. For example, (Meng, Dou, Zhang & Chen, 2014) applied keywords to signify user preferences and recommended APIs corresponding to their semantic compatibility what match with user preferences. (N. Zhang, Wang & Ma, 2017) presents to draw out domain service objectives from context descriptions to satisfy users' intentional requirement. (Yao, Wang, Sheng, Ruan & Zhang, 2015) propose an functional-based service recommendation using functional features and the co-invocation of APIs.

2.4.2 Social network-based Web service recommendation

Using relationship of programmers in the social network to recommend a suitable list of APIs is a another popular approach in service recommendation. Some of such social network-based Web service recommender systems are (B. Cao, Liu, Tang, Zheng & Wang, 2013; W. Chen, Paik & Hung, 2013; W. Xu et al., 2013; W. Gao, Chen,

Wu & Bouguettaya, 2016; T. Liang, Chen, Wu, Dong & Bouguettaya, 2016). Particularly, (B. Cao et al., 2013) attach interest of users and their social relationship to recommender system to program mashup. (W. Chen et al., 2013) combine different type of user-item relations to create a social network to recommend service. (W. Xu et al., 2013) build a global network of social service relying on complex networks and recommend an approach of API discovery. (W. Gao et al., 2016) use the preference history of users to propose a service recommender system, functionalities of services and mashups, and mashup-API invocation relationship. (T. Liang et al., 2016) apply heterogeneous information network to demonstrate heterogeneous objects of APIs, mashups, provider, and tags. This approach also investigate the relations among such objects and design a meta-path based Web service recommender system. The analytical comprehension and multiple embedded factors which contribute to the mashup-API invocation information and therefore such model achieve good recommendation performance.

2.4.3 Collaborative Filtering RS

Currently, Collaborative Filtering (CF) is a popular method for web service recommender systems that utilize the implicit mashup-API invocation information. Existing CF methods can be generally classified into memory-based (Herlocker, Konstan, Borchers & Riedl, 2017; Sarwar et al., 2001a; Linden, Smith & York, 2003; Sarwar et al., 2001a) and model-based (Bobadilla, Ortega, Hernando & Gutiérrez, 2013; Adomavicius & Tuzhilin, 2005; M.-H. Park et al., 2007; Roh et al., 2003; L. Gao & Li, 2008; J. Zhong & Li, 2010). The latter class in general has better prediction accuracy and PMF (Mnih & Salakhutdinov, 2008) is one of the popular models. However, current PMF-based models (P. He, Zhu, Zheng, Xu & Lyu, 2014; S. Li, Wen, Luo, Cheng & Xiong, 2017; Lo, Yin, Deng, Li & Wu, 2012; Sandvig, Mobasher & Burke, 2008; H. Wu, Yue, Li,

Zhang & Hsu, 2018a) are short of the ability to distinguish the importance of invocations or feature interactions, which leads to substandard prediction. In reality, different historical invocations usually have different predictive values and not all latent features contribute relevant information to predict future invocations (Bobadilla et al., 2013). Hence, less relevant features should be assigned lower weights while more relevant features be assigned higher weights. In this thesis, we call these weights *attentional scores*. Furthermore, when the number of mashups and APIs grow, the sparsity of the invocation data is also going up, which influences the accuracy of the model (Kim et al., 2017). There is strong evidence that exploiting the auxiliary information can improve the prediction performance. For example, auxiliary information such as descriptions, reviews, and trustworthiness has been used in (Martinez-Cruz, Porcel, Bernabé-Moreno & Herrera-Viedma, 2015; C. Park, Kim, Oh & Yu, 2016; Y. Zhong, Fan, Tan & Zhang, 2016). Therefore, integrating auxiliary information to the attentional model is promising to model performance. The CF-based models leverage the historical data of similar users or mashups to predict. Some works are a neighbourhood integrated matrix factorization approach, a neighbour-based Region KNN model utilizing the same region historical data (X. Chen, Liu, Huang & Sun, 2010), and a location-aware CF model using location information for QoS prediction (J. Liu et al., 2015).

In addition, there are some other types of CF-based models in service recommendation such as a hybrid random walk model was used to calculate the similarity between mashups (Hu, Peng, Hu & Yang, 2014), a reinforced CF model with mashup-intensive and API-intensive which eliminates the interference of the mashup (Zou et al., 2018). Other works use different types of auxiliary information of mashups and APIs to construct a heterogeneous information network (HIN) and calculate the their similarity scores for invocation prediction (Xie, Wang et al., 2019; T. Liang et al., 2016; Xie, Chen, Lin, Zheng & Lin, 2019).

To tackle the limitation of using a prediction model from single mashup-API invocations, many other hybrid models exploit the context information and invocation history in higher order. Specifically, the correlation scores between mashups and APIs are calculated on the neighbour invocation under CF framework and then obtain the prediction value by using the product function (Jain et al., 2015; Samanta & Liu, 2017). Such approaches only use the linear and product operator which has limitation on capturing the high-order and complex invocation between mashups and APIs.

2.4.4 Hybrid Service Recommendation

Inheriting from above mentioned methods, hybrid approaches incorporate different factors such as history of mashup-API invocation and functionalities to service recommender systems. For instance, (Yao et al., 2015) combine CF and content based recommendation to form a hybrid approach, which actively recommend Web APIs matching the interest of users. (Jain et al., 2015) attached three factors consisting of APIs' functionalities, mashup-API history of existing mashups, and APIs' popularity's into the service recommender systems. They apply matrix factorization based collaborative filtering, probabilistic topic models, and Bayes' theorem to predict the list of APIs for mashup programming. Later on, (Samanta & Liu, 2017) further exploit the Hierarchical Dirichlet Process (HDP) to explore functional relevant APIs relying on their specifications. Then, this work applies the Probabilistic Matrix Factorization (PMF) to predict APIs recommendation list according to invocation history and address the cold start situation to create new mashups among their most adjacency neighbors.

2.5 Summary

This chapter reviews the fundamental concepts including frameworks, definitions, components, methods, and technologies of mashup and Web API in service computing. The

chapter also presents the requirement of context awareness and how it is applied as a type of auxiliary information in existing service computing research. The chapter also reviews the relevant work on graph representation learning which leverages the autoencoder architectures with neighborhood aggregation and convolution layers. For graph structure, higher order of connectivity is also explored with network motifs definitions and concepts in recent works. Finally, the chapter investigates the most popular recommender systems used in service computing consisting of functionality-base, social network-based Web service recommender systems, collaborative filtering recommender systems, and hybrid service recommender systems. Even though extensive existing works on the graph-based collaborative filtering in service recommendation, there are many limitations and challenges need to be addressed. All the sections of this chapter have specifically focused on research to solve the challenges being identified in the research questions of the thesis. Specifically, we explore the works in Section 2.2 and figure out limitation for RQ1. The limitation in Section 2.3.1 and 2.3.2 is used for RQ2. The Section 2.3.4 and 2.3.4 is motivated for RQ3. All challenges identified in Section 2.4 are used to all the three research questions.

Chapter 3

Attentional Matrix Factorization with Context and Co-invocation for service recommendation

3.1 Overview

APIs have been increasingly developed and widely used for Web and mobile applications over the past years ((Bouguettaya et al., 2017; Sheng et al., 2014)). For instance, programmers in different application domains are able to integrate maps into their products in several coding lines with Google Maps API. ProgrammableWeb.com has become the largest online Web API registry with extensively-cited sources of data such as mashup and API descriptions and invocations. However, the accelerated growth in the quantity of APIs has brought difficulties for programmers to choose the right APIs from a large number of candidates. Moreover, sparse API invocation data also brought challenges to API recommendation systems ((Kim et al., 2017)). Hence, it is highly demanded to develop better recommendation techniques for programmers to pinpoint the proper APIs for mashup development.

Currently, Collaborative Filtering (CF) is a popular method for web service recommender systems that utilize the implicit mashup-API invocation information. Existing CF methods can be generally classified into memory-based ((Herlocker et al., 2017; Sarwar et al., 2001a; Linden et al., 2003; Sarwar et al., 2001a)) and model-based ((Bobadilla et al., 2013; Adomavicius & Tuzhilin, 2005; M.-H. Park et al., 2007; Roh et al., 2003; L. Gao & Li, 2008; J. Zhong & Li, 2010)). The latter class in general has better prediction accuracy and PMF ((Mnih & Salakhutdinov, 2008)) is one of the popular models. However, current PMF-based models ((P. He et al., 2014; S. Li et al., 2017; Lo et al., 2012; Sandvig et al., 2008; H. Wu et al., 2018a)) are short of the ability to distinguish the importance of invocations or feature interactions, which leads to substandard prediction. In reality, different historical invocations usually have different predictive values and not all latent features contribute relevant information to predict future invocations ((Bobadilla et al., 2013)). Hence, less relevant features should be assigned lower weights while more relevant features be assigned higher weights. In this chapter, we call these weights *attentional scores*.

Furthermore, when the number of mashups and APIs grow, the sparsity of the invocation data is also going up, which influences the accuracy of the model ((Kim et al., 2017)). There is strong evidence that exploiting the auxiliary information can improve the prediction performance. For example, auxiliary information such as descriptions, reviews, and trustworthiness has been used in ((Martinez-Cruz et al., 2015; C. Park et al., 2016; Y. Zhong et al., 2016)). Therefore, integrating auxiliary information to the attentional model is promising to model performance. In this chapter, we call such auxiliary information *context*.

In this thesis, we improve the conventional PMF model by considering the importance of the binary mashup-API invocations through learning their latent features. We propose an AMF model, which leverages the attentional mechanism in neural network

modelling to weight the latent features interaction of mashup-API invocations. Subsequently, we define an attentional network to parametrize the attentional score and learn them together with the AMF model. To mitigate the sparsity problem, we use Doc2Vec technique (Le & Mikolov, 2014) to exploit the context of APIs and mashups. We also explore the implicit relationships derived from the API co-invocation matrix, in which any pair of APIs is reflected by the number of mashups they invoke together. There is evidence showing that the co-invocation value of two APIs indicates their compatibility in a specific mashup (Yao, Wang, Sheng, Benatallah & Huang, 2018). The main contribution of this work includes:

- We enhance the traditional PMF by setting weights to the latent features of mashups and APIs according to their influence levels, and propose a neural network architecture for the AMF model.
- We propose a method to learn the context of mashups and API by using the Doc2Vec technique. Such method can generate the context similarity between a mashup-API pair and any two APIs pair.
- We implement a statistical analysis for API co-invocation data and learn the distribution of co-invocation frequency for each pair of APIs.
- We conduct comprehensive experiments on ProgrammableWeb dataset with different percentages masked training dataset, and the results show that out model outperforms all the baseline models.

The rest of chapter is organized as follows. Section 3.2 discusses the related work on recommender systems and presents the problem statement. Section 3.3 first gives an overview of our proposed approach, which describes the framework of APIs recommendation, and then defines the AMF model and the method of extracting relevant

information from both context and co-invocation. Section 3.4 presents the experimental evaluation. Finally, Section 3.5 concludes the chapter.

3.2 Preliminaries

3.2.1 Related work

Web service recommendation research was initially focused on service discovery from repositories such as UDDI ((Pastore, 2008)) using keyword and/or ontology based approaches. However, such approaches can only achieve limited accuracy ((Broens et al., 2004; Yao et al., 2014)).

In recent years, with the rapid development of machine learning research, collaborative filtering is widely adopted in service recommendation. In general, CF methods can be classified as memory-based or model-based. Both methods use the the mashup-API interaction matrix. The difference is that the former directly uses the matrix as data in the prediction while the latter extracts information from the data to build a model. In terms of input data, explicit feedback such as QoS ratings is frequently used. For example, ((Zheng, Ma, Lyu & King, 2010)) and ((Shao et al., 2007)) use similarity of QoS ratings of mashups to recommend APIs. ((X. Chen et al., 2010)) augments the memory-based CF approach by using the geographic-based QoS data. ((Tang, Zhang, Liu & Chen, 2015)) uses mashups and APIs' locations to calculate the neighborhood for smoothing the data, then integrates mashup-based and API-based CF for recommendation.

Matrix factorization (MF) ((Mnih & Salakhutdinov, 2008; Koren, 2010)) is one of the most popular model-based CF methods. In general It leverages dot-product as a linear interaction to obtain the latent features for both users and items (or mashups and APIs in the context of Web service recommendation). Recently, Deep Neural Networks

(DNNs) are applied to MF To enhance its capacity. For instance, NeuMF ((X. He et al., 2017b)) applies Multi-Layer Perceptron (MLP) on user-item interactions to extract non-linear information; AFM ((Xiao et al., 2017)) leverages MLP as an attentional mechanism to learn the weights of feature interactions; and HFM ((Tay et al., 2019)) adopts the holographic operator as an augmentation of the inner product. All the above-mentioned approaches treat user-item interactions as isolated pairs instead a connected graph ((X. Wang et al., 2020a)). To further leverage higher-order connectivity among users and items, NGCF ((X. Wang, He, Wang, Feng & Chua, 2019)) and LR-GCCF ((L. Chen, Wu, Hong, Zhang & Wang, 2020)) applies Graph Convolutional Networks (GCN) ((T. N. Kipf & Welling, 2016a)) to learn and reason upon the full user-item interaction graph.

There is also a trend to integrate other sources of information to CF to enhance pr edition accuracy. For example, ((Yao et al., 2014)) consolidates CF and content-based recommendation ((Yao et al., 2014)); ((Naïm, Aznag, Durand & Quafafou, 2016)) leverages pattern mining to present the maximal common context of APIs; ((Buqing, Tang & Huang, 2014)) uses the content similarity between services to recommend APIs; and ((Rahman, Liu & Cao, 2017)) integrates content and network-based service clustering to give recommendation in an API shortlist.

All of the above-mentioned approaches have utilized the relationships between items and APIs. Inspired from these works, our approach learn the relationship among APIs from their invocation history and then attach to the proposed AMF model as a regularization term to enhance recommendation performance.

3.2.2 Problem statement

In a Web API discovery platform like ProgrammableWeb, when a developer wants to build a mashup application, the system generates a list of APIs that are supposed to be

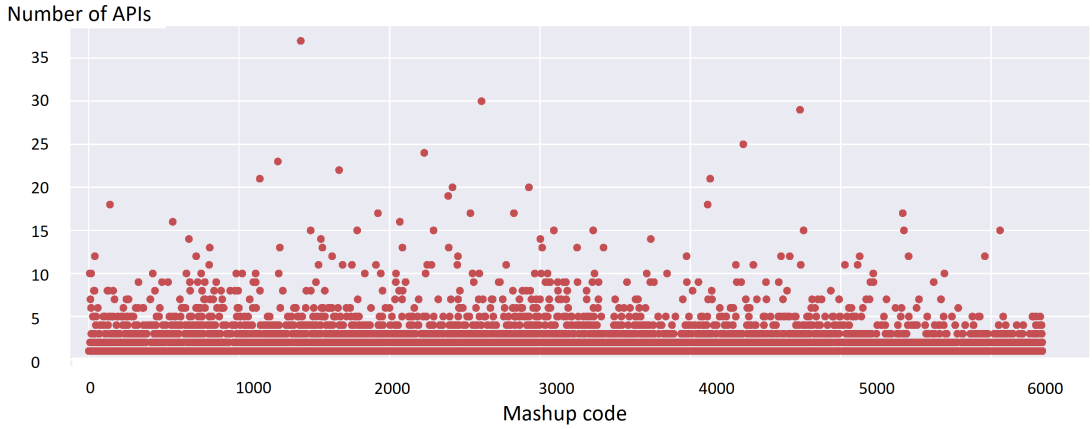

Figure (3.1) Number of APIs in a Mashup

close to the mashup's requirements. We assume that these requirements are descriptions provided by the developer. For example, a food ordering and delivery mashup may be required to provide functions such as restaurant and menu listing, secure payment and customer feedback. Available APIs are abundant and diverse, while a mashup usually only uses a few of them.

Figure 3.1 shows that with over 3500 available APIs, the average number of APIs used in most of the mashups is just around five. Figure 3.2 shows the distributions of APIs over mashups. Each red dot presents an invocation from a mashup to an API. It can be seen that there are many APIs that have never been used by any mashup. Therefore, finding and suggesting a set of most compatible APIs is the main benefit of using recommendation systems.

Like traditional approaches ((Bobadilla et al., 2013)), we focus on the invocation history of APIs and mashups to learn latent features and determine the similarity between APIs and mashups from a matrix factorization model. However, we subtly integrate the attentional score to invocation through the learning process and investigates the context documents of APIs and mashups, which has not been considered by many studies. Also, some APIs are more compatible with the same mashup, and some others are less. Thus, we explore how to derive the implicit relationship of APIs and integrate

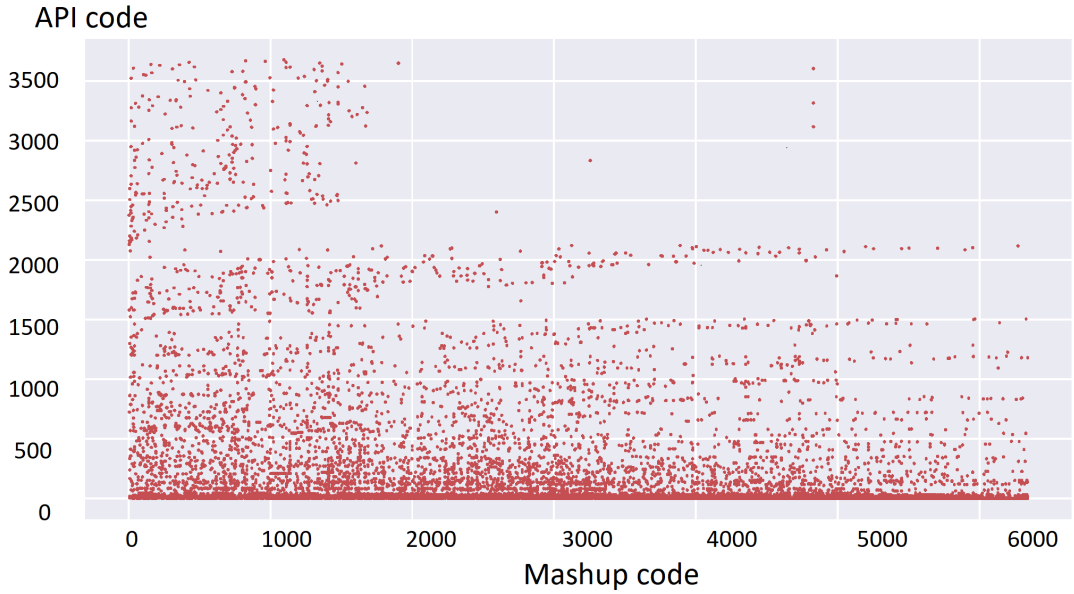

Figure (3.2) Distribution of APIs and mashup

it into our AMF model for better rating prediction.

3.3 Proposed Approaches

3.3.1 Overview

To address the above-mentioned challenge, this chapter proposes a novel framework of APIs recommendation for mashups development. As demonstrated in Figure 3.3, the framework presents the AMF model with Attentional Network and the two main auxiliary information parts including similarity measurement and APIs implicit relationship.

Regarding the similarity measurement part, when an active user wants to make a new mashup, she would type in some descriptions about the mashup. The descriptions from both new mashup and available APIs are fetched into a word embedding model using Doc2vec technique ((Le & Mikolov, 2014)) to generate a document matrix. After that, we use cosine similarity **??** to compute the similarity between the mashup and each APIs, and also between two different APIS using their document matrices.

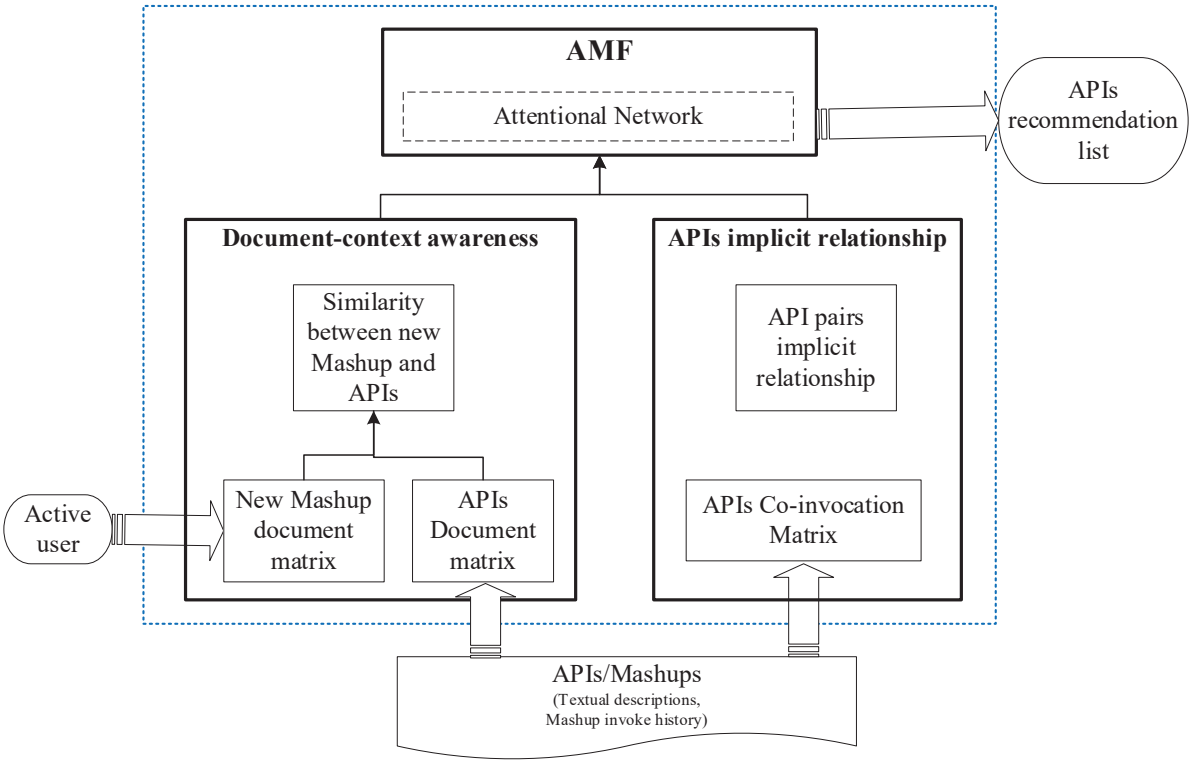

Figure (3.3) The proposed framework for service recommendation.

Regarding the APIs implicit relationship part, API invocation history provides information about their participation in mashups. The implicit relationship of APIs is learned from their prevalence of being in the same mashups.

In the next step, mashup-API similarity and APIs implicit relationship is integrated into an AMF model as regularization terms. A list of APIs with highest recommendation scores will then be recommended for the new mashup development.

The following subsections present further specific aspect of the framework beginning from the general model AMF (Section 3.3.2 and extending to the regularization parts including Document-context awareness and APIs implicit relationship (Section 3.3.3).

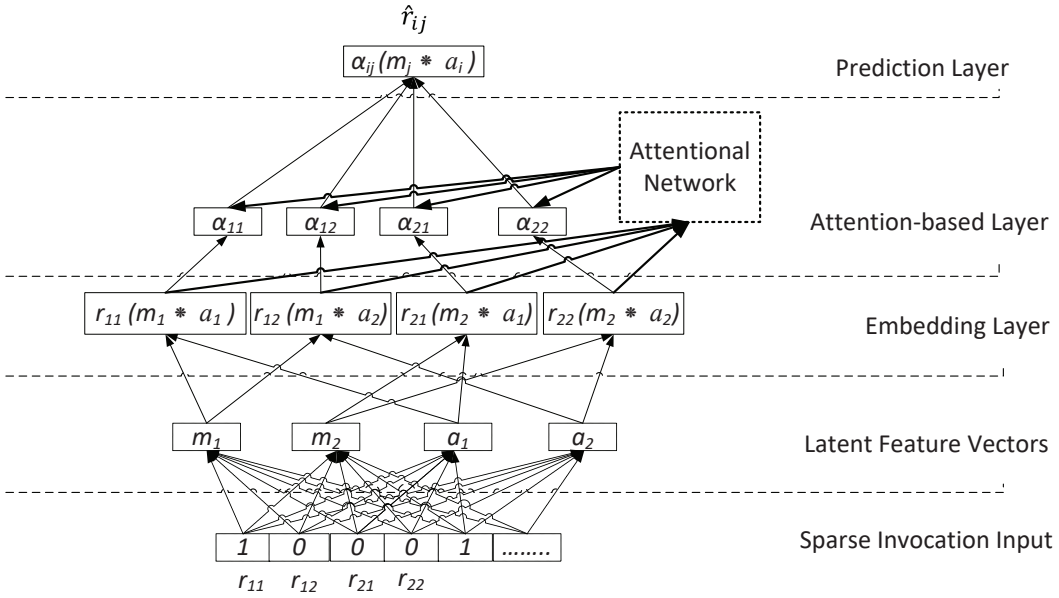

Figure (3.4) The neural network architecture of AMF Model

3.3.2 Attentional PMF Model (AMF)

Supposed that there is an invocation record of n_a APIs and n_m mashups, we denote the invocation relationship between APIs and mashups by a matrix $R \in \mathbb{R}^{n_a \times n_m}$. We use a binary variable r_{ij}, which indicates an invocation pairing of API a_i and mashup m_j. If API a_i is invoked by mashup m_j then r_{ij} is equal to 1, and 0 otherwise. We define $\hat{r_{ij}}$ as the probability that API a_i will be invoked by mashup m_j.

$$r_{ij} = \left\{ \begin{array}{ll} 1 & \text{if } m_j \text{ invoke } a_i \\ 0 & \text{otherwise} \end{array} \right\}. \tag{3.1}$$

Attentional network

Figure 3.4 demonstrates the neural network architecture of our proposed model. The sparse invocation input is the input layer, and it is factorized into two matrices A$\in \mathbb{R}^{n_a \times d}$ and matrix $M \in \mathbb{R}^{n_m \times d}$ which represent latent features of mashups and APIs respectively. In the embedding layer, different from PMF which defines $\hat{r}_{ij}$ as the dot product of two

latent feature vectors a_i and m_j as defined above, we add an attentional score to present the importance of the interaction into the prediction of $\hat{r_{ij}}$ as below:

$$\hat{r_{ij}} = \alpha_{ij} {a_i}^T m_j \tag{3.2}$$

Where α_{ij} is the attentional weight for the invocation between API a_i and mashup m_j, which represents the importance of the dot product ${a_i}^T m_j$ to the predicted value. To learn α_{ij}, we adopt the approach of Xiao et al. ((Xiao et al., 2017)) that constructs an attentional network with a multi-layer perceptron (MLP). By doing so, our model can estimate the attentional scores of these invocations. However, we define attentional score as a scalar, and it is defined as below:

$$\alpha'_{ij} = hReLU\big(W {a_i}^T m_j + b\big) \tag{3.3}$$

$$\alpha_{ij} = \frac{e^{a'_{ij}}}{\sum_{i=1}^{n_a} \sum_{j}^{n_m} e^{a'_{ij}}} \tag{3.4}$$

where $W \in \mathbb{R}^{l \times d}$, $b \in \mathbb{R}^l$, $h \in \mathbb{R}^l$ are parameters of the model, and l is the number of hidden layers of the attentional network. Following the previous work ((Xiao et al., 2017)), we first obtain the attentional score by Equation 3.3 through a rectifier activate function, then we normalize it by a softmax function as in Equation 3.4. We simultaneously learn α_{ij} and compute the features of mashups and APIs by minimizing an attentional sum of squared errors loss function of a PMF model in the following section.

Probabilistic matrix factorization for AMF

Figure 3.5 illustrates the integration of an attentional network into PMF in the form of a probabilistic graphical model ((Mnih & Kavukcuoglu, 2013)). There are n_a APIs and n_m mashups. The matrix $R \in \mathbb{R}^{n_a \times n_m}$ denotes the interaction between mashup-API

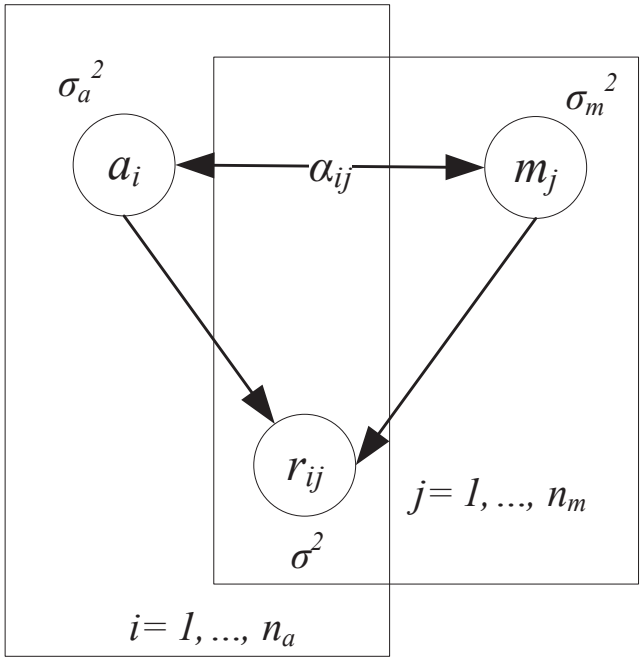

Figure (3.5) The graphical model for AMF

pairs and each interaction between a_i and m_j is r_{ij}. Attentional score between them is α_{ij}. The conditional distribution of the mashup-API interaction data is defined as below:

$$p(R|A, M, \alpha, \sigma^2) = \prod_{i=1}^{n_a} \prod_{j=1}^{n_m} [\mathbb{N}(r_{ij}|f(a_i, m_j, \alpha_{ij}), \sigma^2)]^{I_{ij}} \tag{3.5}$$

, where $\mathbb{N}(\mu, \sigma^2)$ is the probability density function of the Gaussian distribution with mean μ and variance σ. I_{ij} is the interaction index, which is equal 1 if there are invocation between API a_i and mashup m_j, 0 otherwise. The approximate preference of mashup m_j on API a_i is presented by funtion $f(a_i, m_j, \alpha_{ij})$. It is equivalent to the mean of the Gaussian distribution and defined as $f(a_i, m_j, \alpha_{ij}) = {a_i}^T m_j.\alpha_{ij}$, where we calculate the inner product of a_i and m_j, weighted by the attentional score α_{ij}.

We obtain the 0-mean spherical Gaussian priors for mashup and API latent feature matrices as $p(A|\sigma_a^2) = \prod_{i=1}^{n_a} \mathbb{N}(A_i|0, \sigma_a^2 I)$, and $p(M|\sigma_m^2) = \prod_{j=1}^{n_m} \mathbb{N}(M_j|0, \sigma_m^2 I)$. Then,

the posterior distribution of Equation. 3.5 is demonstrated as below:

$$\begin{aligned} & p(A, M|R, \alpha, \sigma^2, \sigma_a^2, \sigma_m^2) \\ \approx\ & p(R|A, M, \alpha, \sigma^2, \sigma_a^2, \sigma_m^2) p(A|\sigma_a^2) p(M|\sigma_m^2) \\ =\ & \prod_{i=1}^{n_a} \prod_{j=1}^{n_m} [\mathbb{N}(r_{ij}|f(a_i, m_j, \alpha_{ij})\sigma^2)]^{I_{ij}} \\ & \times \prod_{i=1}^{n_a} \mathbb{N}(A_i|0, \sigma_a^2 I) \times \prod_{j=1}^{n_m} \mathbb{N}(M_j|0, \sigma_m^2 I) \end{aligned} \tag{3.6}$$

We operate the logarithm of the Equation. 3.6 and obtain the log of posterior distribution as below:

$$\begin{aligned} & \ln p(A, M|R, \alpha, \sigma^2, \sigma_a^2, \sigma_m^2) \\ =\ & -\frac{1}{2\sigma^2} \sum_{i=1}^{n_a} \sum_{j=1}^{n_m} I_{ij} [r_{ij} - f(a_i, m_j, \alpha_{ij})]^2 \\ & -\frac{1}{2\sigma_a^2} \sum_{i=1}^{n_a} A_i^T A_i - \frac{1}{2\sigma_m^2} \sum_{i=1}^{n_m} M_j^T M_j \\ & -\frac{1}{2} \left[\left(\sum_{i=1}^{n_a} \sum_{j=1}^{n_m} I_{ij} \right) \ln \sigma^2 + n_a d \ln \sigma_a^2 + n_m d \ln \sigma_m^2 \right] + C \end{aligned} \tag{3.7}$$

, where C is the independent constant. It is equivalent to minimize the sum of squared errors objective function defined as follows:

$$\min_{A,M} L = \sum_{i=1}^{n_a} \sum_{j=1}^{n_m} \frac{1}{2} I_{ij} (r_{ij} - \alpha_{ij} {a_i}^T m_j)^2 + \frac{\lambda_a}{2} \|A\|_F^2 + \frac{\lambda_m}{2} \|M\|_F^2 \tag{3.8}$$

, where I_{ij} equals 1 if API a_i is invoked by mashup m_j, and 0 otherwise. Regularization parameters are λ_a and λ_m. $|.|_F$ is the Frobenius norm.

3.3.3 Mashup recommendation with integrated with document-context awareness and implicit API relationship

We leverage the advantage of the PMF model and make use of the auxiliary information to reduce the model overfitting and enhance the prediction accuracy. Specifically, we exploit the information of document-context and implicit API relationship for the AMF.

Generally, in recommender systems, both explicit and implicit reviews can be utilized to derive users's preference over items. While the explicit reviews indicate the occurrence of user-item interactions, the implicit reviews obliquely reflect users' sentiment by paying attention to their previous historical interaction. In service recommender systems, mashups are considered as users, APIs are corresponding to items, and ratings are invocations. We interpret the service co-invocation as implicit API relationship and incorporate it with document-context awareness to execute recommendation based on such implicit co-invocations learned from the existing historical data.

Two regularizing terms are used corresponding to the document-context and the implicit relationship for API recommendation. We detail the implementation of these two terms in Section 3.3.3 and Section 3.3.3 respectively.

Document mining with Doc2vec word embedding technique

We implement document mining with the Doc2vec technique for the first regularization term. To prepare the input for the word embedding model, we first collect the data of APIs descriptions from service provider, such as ProgrammableWeb or RapidAPI. The description may include name, category, tags, introduction, and historical mashups. Next, we preprocess the data to extract stemming while keeping stop words as they influence the context of the document.

In the next step, we generate document latent vector v_D from the descriptions which

will be used in our recommendation system model. Namely,

$$D = \begin{bmatrix} | & | & | & & | \\ v_D & v_{w_1} & v_{w_2} & \dots & v_{w_V} \\ | & | & | & & | \end{bmatrix} \tag{3.9}$$

In Equation 3.9, D is the document matrix and $D \in \mathbb{R}^{N\times(V+1)}$, V is the length of the document, and N is the number of latent features according to embedding dimensions for each word w_v with $v \in \{1, .., V\}$. Context similarity between a new mashup m_j and API a_i is denoted by s_{ij} and obtained by the distance between two vector $v_D(m_j)$ and $v_D(a_i)$ by Euclidean norm as below:

$$s_{ij} = |v_D(m_j), v_D(a_i)| \tag{3.10}$$

, where $s_{ij} \in \mathbb{R}^{n_a \times n_m}$. It is woth noting that the similarity s_{ij} is added to attach the context information to the prediction as a regularization term as below:

$$\frac{\lambda_S s_{ij}}{2} |MA^T|_F^2 \tag{3.11}$$

The sensitivity analysis of the proposed method to the setting of N will be discussed in the experiment section.

Implicit APIs relationship mining

We use distribution fitting for co-invocation analysis to build the second term. The recommended APIs have to be appropriate for the new mashup and also compatible with other APIs within the mashup. Therefore, there must be some latent information influencing the co-invocation of APIs, and such latent relationship cannot be directly observed from APIs' description. To solve this problem, ((Yao et al., 2018)) propose a

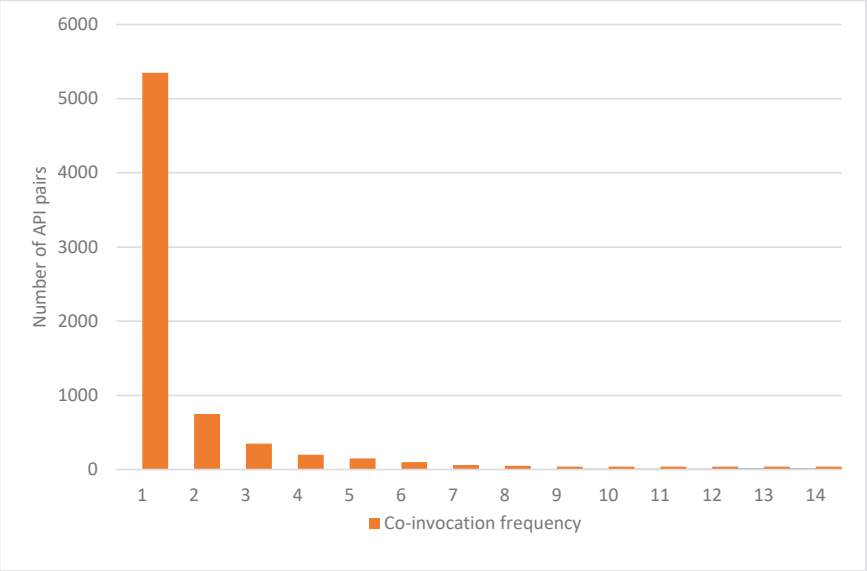

Figure (3.6) Co-invocation frequency of API pairs

pre-trained model to learn this implicit correlation from textual profile and co-invocation history, and add it to the main PMF as a regularization part. Then TF-IDF is used to derive the context similarity of APIs, assuming the distribution is Poissonfor co-invocation counts of each pair of APIs. In this work, we do not pre-train the model but implement statistical analysis for the co-invocation counts and derive a suitable distribution. Such fitting effectively represents the probability of APIs' invocation by the same mashup. Using Easyfit, we fit the distributions for this data. Figure 3.7 ((Rinne, 2008)) is the fitting results showing that Weibull is the best-fit distribution with parameters $\lambda = 0.59726$ and $k = 31.522$. The Goodness of Fit shows how much distributions fit the data and Weibull achieves the best among three different statistical tests including Kolmogorov Smirnov, Anderson Darling, and Chi-Squared:

$$P_{ik} = \frac{k}{\lambda}(\frac{f_{ik}}{\lambda})^{k-1} e^{(\frac{-f_{ik}}{\lambda})^k} \tag{3.12}$$

where f_{ik} is the co-invocation frequency of API a_i and mashup m_j. APIs which are used to be invoked in the same mashup have much chance to be together in the future

Fitting Results

#	Distribution	Parameters
1	Weibull	α= 0.59726 β= 31.522
2	Weibull (3P)	α= 0.37376 β= 48.86 γ= 1.0

Goodness of Fit - Summary

#	Distribution	Kolmogorov Smirnov		Anderson Darling		Chi-Squared	
		Statistic	Rank	Statistic	Rank	Statistic	Rank
1	Exponential	0.664	4	63.417	5	122.69	3
2	Exponential (2P)	0.66603	5	72.706	6	124.97	4
3	Gamma	0.70539	6	26.073	4	79.62	2
4	Gamma (3P)	0.34274	3	15.51	3	N/A	
5	Weibull	0.17993	1	3.137	1	5.1328	1
6	Weibull (3P)	0.20017	2	9.7723	2	N/A	
7	Erlang	No fit					
8	Erlang (3P)	No fit					

Figure (3.7) Distribution fitting for frequency of API pairs

because of their latent relevant features. I.e., if API a_i has a co-invocation list of APIs $a_k \in K(i)$, then we can assume that a_i's latent vector should be closed to that of a_k.

We fully demonstrate the implicit relationship between APIs by both context similarity and prevalence of co-invocation by integrating these two elements into a single measurement of Re_{ik}. Re_{ik} is the implicit relationship of API a_i and a_k and is defined by weighted summation of context similarity and co-invocation prevalence as below:

$$Re_{ik} = \theta s_{ik} + (1 - \theta) P_{ik} \tag{3.13}$$

, where θ is the weight for context similarity. Hence, we revise the original regularization term as below:

$$\frac{\gamma}{2} \sum_{i=1}^{x} |(A_i - \frac{\sum_{k \in K(i)} Re_{ik} \times A_k}{\sum_{k \in K(i)} Re_{ik}})|_F \tag{3.14}$$

The implicit relationship allows the regularization term to treat APIs' co-services differently. If API a_i is very similar to a_k, say $Re_{ik} = 0.99$, then a_k should contribute more influences on features of a_i. Reversely, if $Re_{ik} = 0.01$, then a_k should influences

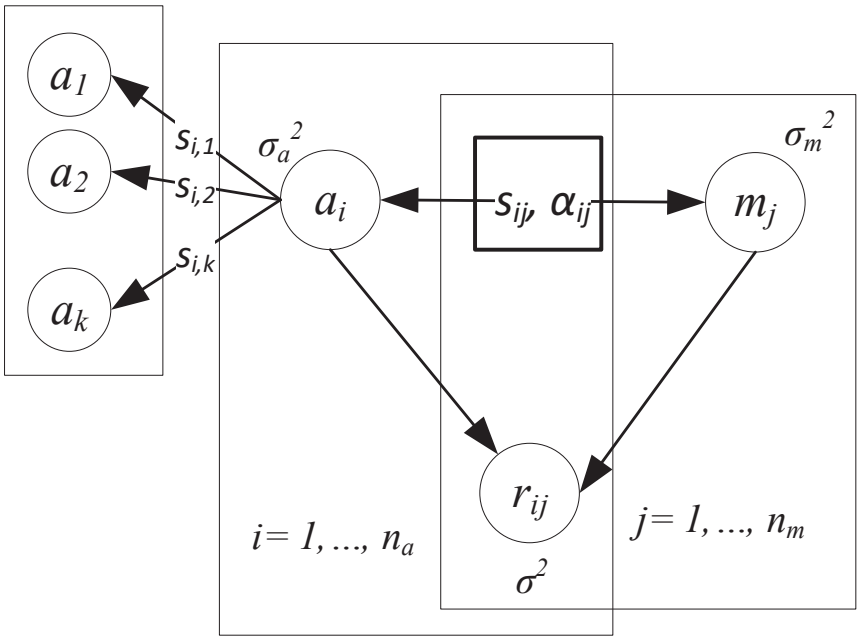

Figure (3.8) The graphical model for AMF with regularization.

less.

Probabilistic matrix factorization of AMF⁺ model

We have defined the context and API co-invocation (refer Section 3.3.3 and Section 3.3.3). We utilize both of them to enhance the AMF as a better model named AMF⁺. Figure 3.8 illustrates the probabilistic graphical model of AMF⁺. Different from AMF, to weight the interaction between mashup m_j and API a_i, we add both s_{ij} and a_{ij} into their latent features inner product. This way allow us to inject the context into the model. Therefore, the conditional distribution of interaction data for AMF⁺ is demonstrated as:

$$p(R|A, M, s, \alpha, \sigma^2) = \prod_{i=1}^{n_a} \prod_{j=1}^{n_m} [\mathbb{N}(r_{ij}|f(a_i, m_j, s_{ij}, \alpha_{ij}), \sigma^2)]^{I_{ij}} \tag{3.15}$$

We also attach the context and the co-invocation of neighbourhood APIs into the regularization parts as defined in Section 3.3.3 and Section 3.3.3. Similar to

Section 3.3.2, we use Bayesian to reconstruct the posterior distribution of Equation 3.15 and transfer to the natural logarithm. Finally, maximizing the log of posterior is similar to minimize a confidence weighted sum of squared errors loss function as in Equation 3.16.

$$\begin{aligned}\min_{A,M} L = & \sum_{i=1}^{n_a}\sum_{j=1}^{n_m}\frac{1}{2}I_{ij}(r_{ij} - s_{ij}\alpha_{ij}{a_i}^T m_j)^2 \\ & + \frac{\gamma}{2}\sum_{i=1}^{n_a}|(A_i - \frac{\sum_{k\epsilon K(i)} Re_{ik}\times A_k}{\sum_{k\epsilon K(i)} Re_{ik}})|_F \\ & + \frac{\lambda_S s_{ij}}{2}|MA^T|_F^2 + \frac{\lambda_a}{2}|A|_F + \frac{\lambda_m}{2}|M|_F\end{aligned} \tag{3.16}$$

The relationship function Re_{ik} allows the context and implicit relationship regularization term treat APIs' correlations differently. If APIs a_i and a_k are very compatible and very similar, then Re_{ik} closes to 1. Reversely, if these two APIs are contextly different and have never been simultaneously invoked in any mashup, Re_{ik} should be close to 0.

We use gradient descent in feature vectors A_i and M_j to obtain the local minimum of the objective function given in Equation 3.16.

$$\begin{aligned}\frac{\partial L}{\partial A_j} = & -\sum_{i=a}^{n_a} I_{ij}(r_{ij} - s_{ij}\alpha_{ij}{a_i}^T m_j)M_j + s_{ij}\lambda_S MA^T M \\ & + \lambda_a A_i + \gamma(A_i - \frac{\sum_{k\epsilon K(i)} Re_{ik}\times A_k}{\sum_{k\epsilon K(i)} Re_{ik}}) \\ & + \gamma\sum_{g\epsilon K^-(i)}\frac{(-Re_{ig})(A_g - \frac{\sum_{k\epsilon K(g)} Re_{gk}\times A_k}{\sum_{k\epsilon K(g)} Re_{gk}})}{\sum_{k\epsilon K(g)} Re_{gk}}\end{aligned} \tag{3.17}$$

$$\frac{\partial L}{\partial M_j} = -\sum_{j=m}^{y} I_{ij}(r_{ij} - s_{ij}\alpha_{ij}{a_i}^T m_j)A_i + s_{ij}\lambda_S MA^T A + \lambda_m M_j \tag{3.18}$$

3.4 Experiments

This section presents the empirical performance of the proposed model in service recommendation, specifically with ProgrammableWeb dataset. We run the experiments on Anaconda environment with Python 3.7 on a PC with Intel i5-6500@3.2 GHz CPU,

16 GB RAM. Our objective is to answer the following research questions:

RQ1 How does the proposed model perform as compared with the state-of-the-art baseline CF methods?

RQ2 How much do the context similarity (Doc2vec word embedding) and API co-invocation prevalence influence the performance of the proposed model?

RQ3 How much impact are the hyperparameters including the number of context latent features, similarity weight θ, regularization λ_S, and γ on the experiment results of the proposed models and baselines as well?

3.4.1 Set up data

We use the dataset of ProgrammableWeb[1] to demonstrate the effectiveness of the proposed model. The dataset consists of textual descriptions of 17829 APIs and 6340 mash-ups, and their historical invocation. Because the dataset does not have the ratings between mashups and APIs, we adopt their invocation data as the ratings. For example, if mashup m_1 invokes API a_5, their invocation data is 1, and we use this value as rating between this mashup- API pair. After that, we do reprocessing for the dataset as follows: (1) We remove blank APIs and mashups, and obtain 5691 mashups and 1170 APIs; (2) We obtain the document vectors of all descriptions by pre-train model Doc2Vec by using library Gensim Doc2Vec and calculate the description similarity between descriptions by Cosine similarity. (3) We use Python to do programming to define the co-invocation frequency between pairs of APIs. (4) We also calculate the historical invocation between pairs of mashup-API; (5) We remove a certain portion of the mashup-API interactions to create a training set, and the original data becomes the test set.

[1]We download the database from https://dev.maxmind.com/in November 2018.

3.4.2 Baselines

We compare the proposed model with the following baselines.

1. SVD++: SVD based Matrix Factorization Model applies singular value decomposition ((Koren, 2010)).

2. PMF: Probabilistic Matrix Factorization Model purely uses ratings for collaborative filtering ((Mnih & Salakhutdinov, 2008)).

3. LDA-MF: This model uses the LDA instead of Doc2Vec techniques to pre-train the context similarity of services and mashup, then attaches them into PMF as a regularization part ((Blei, Ng & Jordan, 2003)).

4. CI-MF: Matrix Factorization based model with API Co-invocation regularization, which uses TF-IDF for text mining ((Yao et al., 2018)).

5. NeuMF: This is the vanilla matrix factorization method that represents users and items as feature vectors and exploits their direct interactions by inner product and binary logistic loss function ((X. He et al., 2017b)).

6. GeoCF: Geographic-aware collaborative filtering for service recommendation which considers graphic location information from a mashup-API interaction to derive the preference degree with better prediction ((Botangen, Yu, Sheng, Han & Yongchareon, 2020)).

7. AMF: AMF model without regularization parts of context-based awareness and API co-invocation relationships.

8. AMF^{+}: AMF model with regularization parts of context-based awareness and API co-invocation relationships.

Table (3.1) Comparison between six models.

Model	Ratings	Context	pre-train	words sequences	Attention	API relationship
SVD++	Yes	No	No	No	No	No
PMF	Yes	No	No	No	No	No
LDA-PMF	Yes	Yes	Yes	No	No	Yes
CI-MF	Yes	Yes	Yes	No	No	Yes
NeuMF	Yes	No	Yes	No	No	No
GeoCF	Yes	Yes	Yes	No	No	Yes
AMF	Yes	No	No	No	Yes	No
AMF$^+$	Yes	Yes	Yes	Yes	Yes	Yes

Table 3.1 shows the comparison between these baselines with six elements, including ratings, context-awareness, pre-train word embedding, words sequence, attention, and API relationship. It can be seen that the proposed model leverages all these elements to obtain a better prediction. We consider our model without context-awareness named AMF to observe how much context information and API implicit relationship influences this prediction model. Existing models such as PMF, SVD++, NeuMF do not integrate context information and API implicit relationship.

3.4.3 Evaluation metric

We test the performance of the above baseline models on the Programmable-Web dataset. We take the original mashup-API matrix and mask a percentage of original invocations where a mashup-API invocation has been taken place for use as a test set. The test set will contain all of the original invocations, while the training set replaces the specified percentage such as 20% of them with a zero in the original invocation matrix.

To measure the performance of classification and recommendation ranking list, we use the two common metrics including Normalized discounted cumulative gain (NDCG) and average precision (MAP). Different from the prediction accuracy metrics which

measure the gap between the prediction value are to the actual ones (e.g., RMSE and MAE), our chosen metrics first consider the degree of precision of recommended APIs based on relevance, and then the capability to predict the precise order of recommendation list according to the degree of preferences.

Normalized Discounted Cumulative Gain (NDCG@k) : From the ranking of relevant APIs in the recommendation list, we measure the effectiveness of recommendation by the Discounted Cumulative Gain (DCG). The gain is accumulated from the top to the bottom of recommendation list for each relevant API. We discount and penalize the APIs which has lower rank. Namely, the DCG from rank 1 to k for the subset of a recommendation list is defined as:

$$\text{NDCG@}k = \sum_{i=1}^{N_a} \frac{2^{r_i} - 1}{\log_2(i+1)} \tag{3.19}$$

, where N_a, N_m is the number of APIs and mashups in the recommendation list.

Mean average precision (MAP@k): Precision is the portion of recommend APIs those are actually invoked by the mashup. Precision is also known as true positive accuracy, that measures the probability that a recommended API is invoked by the mashup. We denote $P@k$ as a precision which is calculated from only a recommendation list from rank 1 through K. Provided a mashup m_j, the Average Precision $AP@k_{m_j}$ is the precision of each relevant API, regarding its position in the ranked recommendation list of k APIs: $AP@k_{m_j} = \frac{\sum_{i=1}^{k} Pr(n) \times I(n)}{N_a}$, where $Pr(n)$ is the precision at cutoff n in the list, and $I(n)$ is equal to 1 if the API at rank n is actually invoked, zero otherwise. The AP metric identifies every relevant recommendation and give prize frontal recommendation with the most probable correct APIs, such as the relevant APIs at the top of the recommendation list. Hence, we call the arithmetic average of the

AP@k of all mashups as the Mean Average Precision :

$$MAP@k = \frac{\sum_{i=1}^{N_m} AP@K_m}{|N_m|} \tag{3.20}$$

3.4.4 Hyperparameter settings

We do the parameters tuning for all baselines models and then compare their performances with the proposed model.

For this work, we set the latent dimension d = 20. We tune hyperparameters for each mentioned models and obtain the hyperparameters with their own best performance. Namely, we tune the λ_a and λ_m with values in (0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1.0). The latent features in range [5, 10,..., 50]. Each models also need different numbers of epoch to have the performance converging. For document-based models including LDA-MF, GeoCF, AMF, and AMF$^+$, we tune the numbers of context latent features N from 5 to 100. Table 3.2 shows the best hyperparameters for each models.

3.4.5 Experimental results

We run the above baselines and our proposed models with the 10%, 20%, 30%, 40%, and 50% masked training dataset. We also generate 5 subsets of the data for each masked settings. The value of k is set to 1, 3, 5, and 10. Then, we use two metrics NDCG and MAP to evaluate the baselines and our models. We also examine the performance of the models by varying their hyperparameters and observe the settings in which our models achieve the best scores.

Comparison with baselines performance

To assess the NDCG of the models, we create 5 settings of masked data 10%, 20%, 30%, 40%, 50%. Then, we randomly generate 5 different subsets from the dataset and

Table (3.2) Comparison of hyperparameters.

Model	λ_a, λ_m	Learning rate	No. of latent features	Converged epoch
SVD++	0.01	0.01	35	50
PMF	0.1	0.1	40	500
LDA-MF	0.005	0.005	20	100
CI-MF	0.001	0.001	35	400
NeuMF	NA	0.001	20	100
GeoCF	0.001	0.001	20	500
AMF	0.1	0.0005	20	100
AMF$^+$	0.1	0.005	20	100

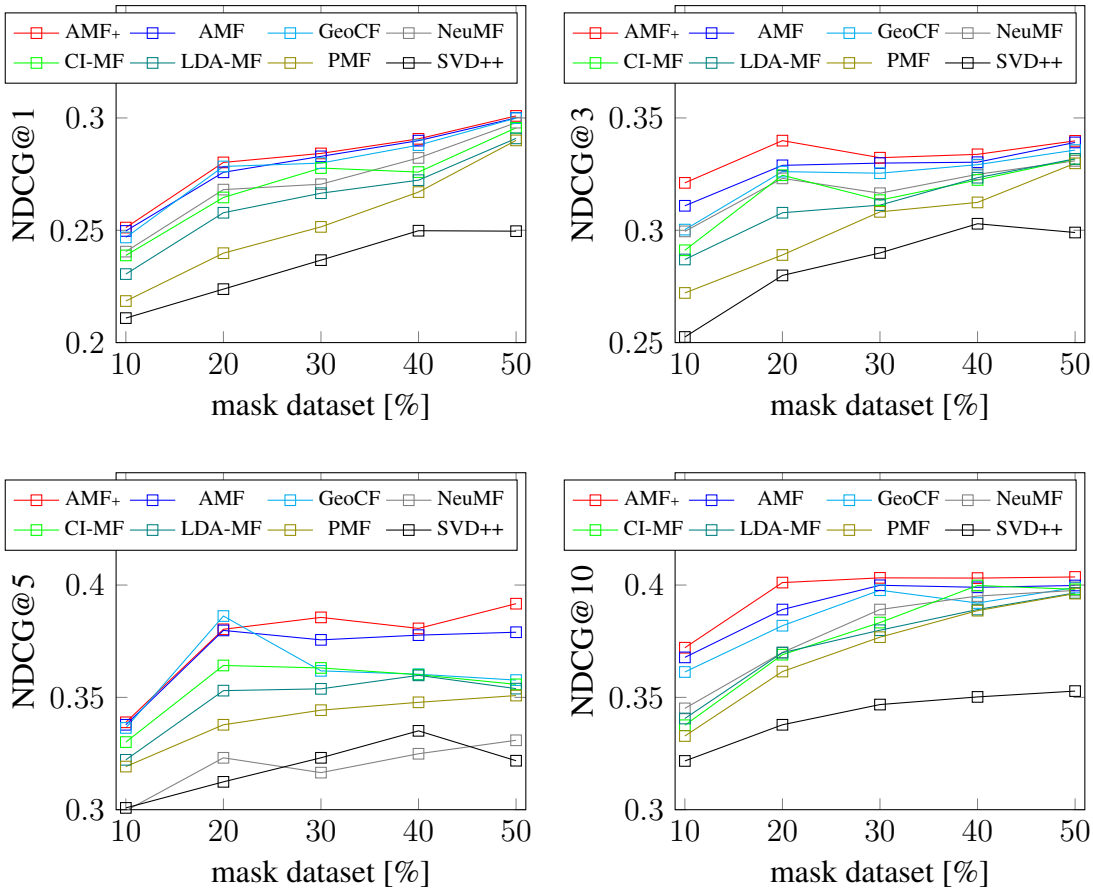

Figure (3.9) NDCG@k Performance comparison

Table (3.3) NDCG scores of baselines and proposed models.

NDCG@1					
	10%	20%	30%	40%	50%
SVD++	0.2109	0.2238	0.2367	0.2498	0.2496
PMF	0.2185	0.2398	0.2515	0.2670	0.2900
LDA-MF	0.2305	0.2578	0.2665	0.2723	0.2909
CI-MF	0.2388	0.2646	0.2777	0.2759	0.2956
NeuMF	0.2405	0.2681	0.2705	0.2822	0.2979
GeoCF	0.2468	0.2784	0.2799	0.2879	0.2999
AMF	0.2497	0.2758	0.2829	0.2899	0.3000
AMF$^+$	**0.2512**	**0.2802**	**0.2842**	**0.2907**	**0.3009**
NDCG@3					
	10%	20%	30%	40%	50%
SVD++	0.2525	0.2799	0.2899	0.3029	0.2990
PMF	0.2721	0.2890	0.3082	0.3124	0.3298
LDA-MF	0.2870	0.3078	0.3112	0.3234	0.3318
CI-MF	0.2911	0.3245	0.3134	0.3224	0.3317
NeuMF	0.2995	0.3231	0.3165	0.3249	0.3309
GeoCF	0.3003	0.3261	0.3254	0.3292	0.3357
AMF	0.3109	0.3289	0.3299	0.3303	0.3391
AMF$^+$	**0.3211**	**0.3399**	**0.3323**	**0.3338**	**0.3398**
NDCG@5					
	10%	20%	30%	40%	50%
SVD++	0.3008	0.3124	0.3231	0.3351	0.3218
PMF	0.3192	0.3378	0.3443	0.3478	0.3508
LDA-MF	0.3222	0.3530	0.3538	0.3598	0.3539
CI-MF	0.3298	0.3552	0.3585	0.3599	0.3556
NeuMF	0.3301	0.3642	0.3631	0.3601	0.3558
GeoCF	0.3364	0.3862	0.3618	0.3603	0.3577
AMF	0.3378	0.3798	0.3756	0.3777	0.3790
AMF$^+$	**0.3389**	**0.3803**	**0.3856**	**0.3807**	**0.3917**
NDCG@10					
	10%	20%	30%	40%	50%
SVD++	0.3217	0.3378	0.3468	0.3502	0.3528
PMF	0.3328	0.3615	0.3768	0.3886	0.3962
LDA-MF	0.3406	0.3699	0.3799	0.3891	0.3965
CI-MF	0.3376	0.3690	0.3833	0.3998	0.3981
NeuMF	0.3451	0.3700	0.3891	0.3951	0.3976
GeoCF	0.3613	0.3819	0.3977	0.3920	0.3989
AMF	0.3678	0.3891	0.3999	0.3990	0.3998
AMF$^+$	**0.3721**	**0.4011**	**0.4032**	**0.4031**	**0.4036**

split them into train-test with setting masked ratios. The result of NDCG comparison is in Table 3.3 and Figure 3.9. It can be seen that both AMF and AMF$^+$ obtain very good NDCG compared with the baseline. It is noticed that NDCG does not decrease when increasing the size of masked data for PMF-based models. Only SVD++ has a slight decrease with above 40% data masked. It is reasonable because the larger masked data means the more actual data in the test set. As NDCG looks for predictions those occur in the actual data, the probability of true positives will grow when the number of actual data goes up.

In addition, based on the NDCG plot in Figure 3.3, we can gather the models in three groups based on how close their NDCG scores. Thus, SVD is the only in the first group. PMF, LDA-MF, CI-MF, and NeuMF are in the second group. The last group includes GeoCF, AMF, and AMF$^+$. The third group has the best performance as they all effectively inject the context information to the PMF structure both in the factorization and regularization parts. The second group has smaller NDCG scores when only using context information as regularization parts in the PMF structure. SVD++ does not use any context data and has the smallest score. Is is also clear that all models except SVD++ achieves closer NDCG scores with 50% masked data for all all settings.

For evaluation on MAP metric, we construct 4 groups of train-test data: $n_{dt} > 4, n_{dt} > 3, n_{dt} > 2$, and $n_{dt} > 1$, where n_{dt} is the number of APIs in a mashup of the dataset. Namely, for group $n_{dt} > 4$, we consider only mashups which have more than 4 APIs in the dataset and randomly pick up 1 API invocation per mashup for test data and obtain the settings: $n_{tr} > 3, n_{te} = 1$. Similarly, we pick up 2 APIs invocation per mashup for test data for setting $n_{tr} > 2, n_{te} = 2$, and likewise for setting $n_{tr} > 1, n_{te} = 3$ and $n_{tr} > 0, n_{te} = 4$. Where n_{tr} is the number of APIs in a mashup of the train data and n_{te} is the number of APIs in a mashup of the test data. Likewise for the remaining groups. Table 3.4 summarizes the data arrangement for MAP calculation process. We obtain the MAP@k with $k = 1, 3, 5, 7, 10$ for each test setting.

Table (3.4) Train-test settings for MAP calculation

Group	Setting	Invocations for train	Invocations for test
$n_{dt} > 4$	$n_{tr} > 3, n_{te} = 1$	2115	345
	$n_{tr} > 2, n_{te} = 2$	1770	690
	$n_{tr} > 1, n_{te} = 3$	1425	1035
	$n_{tr} > 0, n_{te} = 4$	1080	1380
$n_{dt} > 3$	$n_{tr} > 2, n_{te} = 1$	2847	589
	$n_{tr} > 1, n_{te} = 2$	2258	1178
	$n_{tr} > 0, n_{te} = 3$	1669	1767
$n_{dt} > 2$	$n_{tr} > 1, n_{te} = 1$	3801	1066
	$n_{tr} > 0, n_{te} = 2$	2735	2132
$n_{dt} > 1$	$n_{tr} > 0, n_{te} = 1$	5097	2362

The result details of MAP@k scores is shown in Table 3.5 of the four train-test settings which we choose from each group in Table 3.4. Namely, the first part in Table 3.5 is the setting $n_{tr} > 0, n_{te} = 1$ of group $n_{dt} > 1$. The three remaining parts are for settings $(n_{tr} > 0, n_{te} = 2)$, $(n_{tr} > 1, n_{te} = 2)$, and $(n_{tr} > 2, n_{te} = 2)$ of group $n_{dt} > 2$, $n_{dt} > 3$, and $n_{dt} > 4$ respectively. Figure 3.10 shows the comparison of MAP@k scores of baselines and our proposed models. For the first setting at the upper left of Figure 3.10, the scores increase sharply and then slightly rise up with $k > 3$. Whereas, the other three settings have different patterns. Namely, the MAP scores goes down for k from 1 to 3, and goes up afterward. It is because we do not penalize wrong recommendations by MAP when increasing more items in recommendation list.

Generally, our proposed AMF and AMF$^+$ outperform all the baselines in all settings in term of MAP in the top 10 recommendations. Namely, our models are better for APIs recommendation.It proves that our models are more accurate in specifying and ranking proper APIs for more effective recommendations.

Table (3.5) MAP scores of baselines and proposed models.

MAP@k: $n_{tr} > 0, n_{te} = 1$					
k	1	3	5	7	10
SVD++	0.1306	0.1847	0.1927	0.2101	0.2113
PMF	0.1112	0.1453	0.1704	0.1921	0.2001
LDA-MF	0.1487	0.1987	0.2107	0.2201	0.2293
CI-MF	0.1511	0.2065	0.2176	0.2209	0.2309
NeuMF	0.1621	0.2092	0.2178	0.2214	0.2312
GeoCF	0.1651	0.2085	0.2184	0.2221	0.2326
AMF	0.1812	0.2152	0.2198	0.2276	0.2336
AMF$^+$	**0.1821**	**0.2199**	**0.2208**	**0.2289**	**0.2356**
MAP@k: $n_{tr} > 0, n_{te} = 2$					
k	1	3	5	7	10
SVD++	0.1482	0.1471	0.1687	0.1712	0.1759
PMF	0.1306	0.1207	0.1607	0.1651	0.1751
LDA-MF	0.1926	0.1685	0.1833	0.1867	0.1876
CI-MF	0.2028	0.1705	0.1954	0.1960	0.1966
NeuMF	0.2111	0.1701	0.1962	0.1968	0.2101
GeoCF	0.2148	0.1729	0.1971	0.1990	0.2196
AMF	0.2186	0.1899	0.1992	0.2021	0.2211
AMF$^+$	**0.2244**	**0.1998**	**0.2002**	**0.2153**	**0.2420**
MAP@k: $n_{tr} > 1, n_{te} = 2$					
k	1	3	5	7	10
SVD++	0.1828	0.1522	0.1638	0.1707	0.1821
PMF	0.1316	0.1227	0.1624	0.1615	0.1732
LDA-MF	0.2088	0.1622	0.1721	0.1827	0.1962
CI-MF	0.2102	0.1794	0.1834	0.1987	0.2021
NeuMF	0.2129	0.1799	0.1908	0.2001	0.2112
GeoCF	0.2156	0.1832	0.1984	0.2043	0.2102
AMF	0.2182	0.1881	0.2001	0.2093	0.2199
AMF$^+$	**0.2199**	**0.1999**	**0.2131**	**0.2203**	**0.2312**
MAP@k: $n_{tr} > 2, n_{te} = 2$					
k	1	3	5	7	10
SVD++	0.199	0.1209	0.1241	0.1421	0.1512
PMF	0.1316	0.1087	0.1124	0.1315	0.1432
LDA-MF	0.2217	0.1177	0.1222	0.1598	0.1739
CI-MF	0.2342	0.1289	0.1322	0.1622	0.1892
NeuMF	0.2346	0.1381	0.1389	0.1720	0.1926
GeoCF	0.2463	0.1437	0.1583	0.1724	0.1902
AMF	0.2653	0.1688	0.1793	0.1898	0.1976
AMF$^+$	**0.2799**	**0.1688**	**0.1967**	**0.1998**	**0.2211**

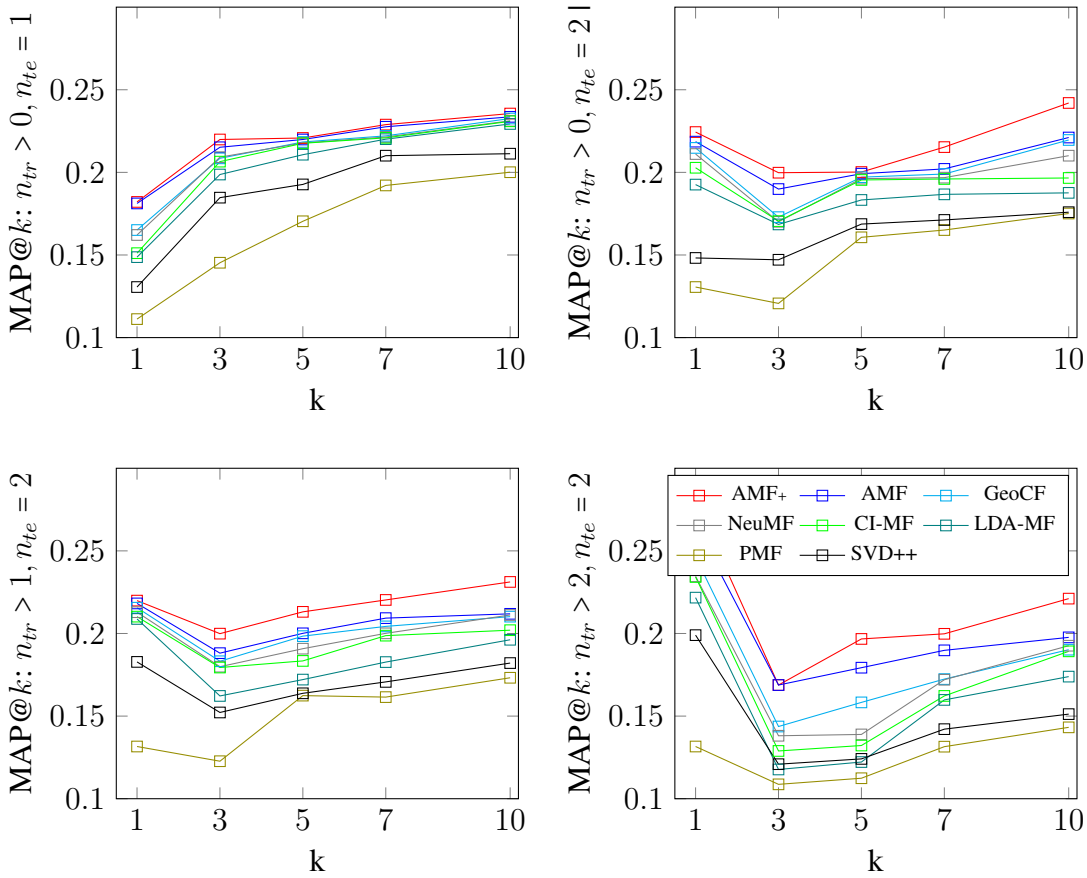

Figure (3.10) MAP@k performance comparison.

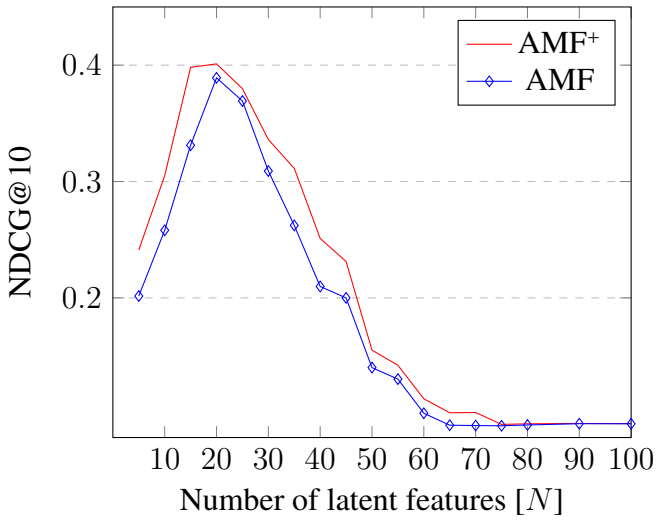

Figure (3.11) NDCG@10 with different numbers of latent feature.

Hyperparameters sensitivity analysis

To verify the significance of document mining embedding and APIs co-invocation to our model's performance, we analyse these following hyperparameters including the number of latent features, the context similarity weight θ, and context regularization λ_S

Influence of number of context latent features in Doc2vec embedding The number of context latent features (see Section 3.3.1) significantly influences the embedding results and then impacts the PMF-based model, which use them as auxiliary information. The larger N embeds more influence of context part into the model. The smaller N means the model less involves the impact of this auxiliary data. information. We set the value of $N = 5, 10, 15, ..., 100$. We then calculate NDCG@10 with 20% masked data and MAP@10 with setting $(n_{tr} > 2, n_{te} = 2)$. As shown in Figure 3.11, the NDCG scores of both AMF and AMF$^+$ generally have the same pattern and AMF$^+$ always outperforms in all settings. Namely, they both rapidly rise up and peak at about 0.4 of $N = 20$ setting. Then, they drop down gradually to about 0.1 as N increase from $N = 20$ to $N = 60$. Later on, the score of AMF plateau with N> 60 while this score also reaches to a limitation from $N > 75$ for AMF$^+$. The same pattern results for MAP score are shown in Figure 3.12.

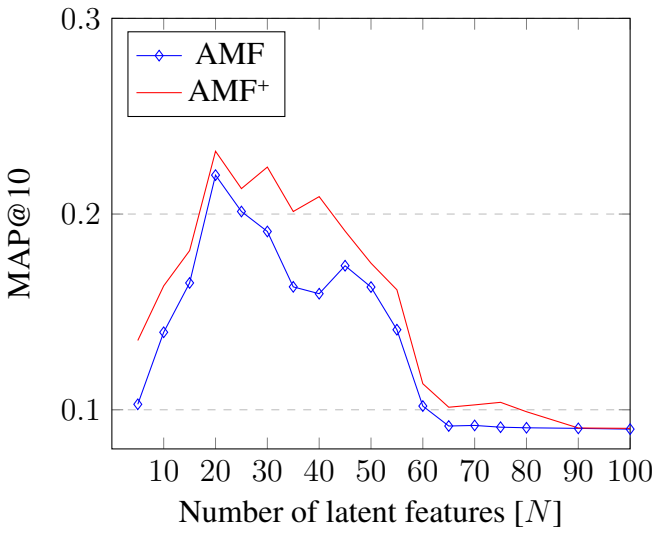

Figure (3.12) MAP@10 with different numbers of latent feature.

This experiment results demonstrate three facts about context auxiliary data. First, provided the range of N values from which the NDCG and MAP scores getting steady or rising up, we have a couple of options for N that can obtain the exceptional performance (e.g, the value $N = \{15 - 40\}$ for both metrics). Second, the performances enhance on the both metrics as N increases, that harmonizes an insight that more latent features would derive more related information. Reversely, less latent features (e.g., $N < 15$) could reduce the ability of exploiting context data, and then leading to poor prediction results. However, considering exaggerated number of latent features drives the models to overfitting ((H. Wu, Yue, Li, Zhang & Hsu, 2018b)) and diminishes their performance. As can be seen in Figure 3.11 and Figure 3.12, the performance is depreciating as N go down beyond 60.

Influence of context similarity weight θ The weight θ (see Section 3.3.2 presents for the importance of context similarity. We examine the influence of θ to the implicit relationship of APIs pair and the results are shown in Table 3.6. At every k values, the implicit relationship increases when we increase the importance of context similarity s_{ij}. This is due to the stronger influence of context information over co-invocation.

In addition, we test the influence of θ on the model performance. At each value of θ,

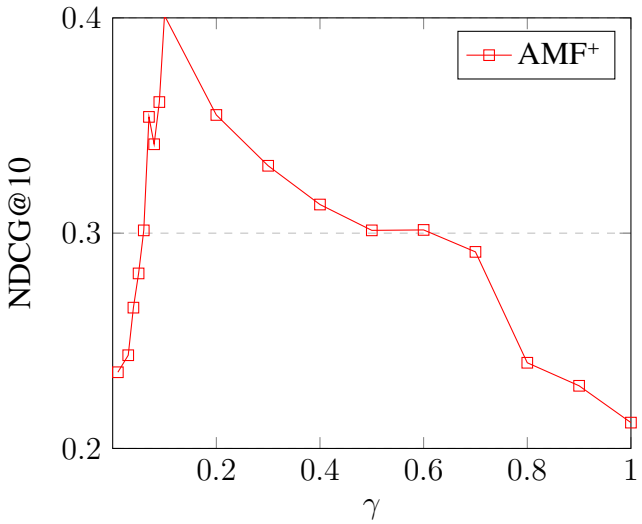

Figure (3.13) Influence of γ on NDCG@10 of the proposed model.

Table (3.6) Influence of θ on Implicit APIs relation.

N	$\theta = 0.1$	$\theta = 0.2$	$\theta = 0.3$	$\theta = 0.4$	$\theta = 0.5$	$\theta = 0.6$	$\theta = 0.7$	$\theta = 0.8$	$\theta = 0.9$
5	0.1996	0.2782	0.3789	0.4381	0.5110	0.5906	0.6692	0.7479	**0.8271**
10	0.2001	0.2817	0.3607	0.4392	0.5117	0.5928	0.6702	0.7501	**0.8284**
15	0.2026	0.2828	0.3612	0.4401	0.5152	0.5966	0.6742	0.7518	**0.8294**
20	0.2086	0.2862	0.3638	0.4414	0.5190	0.5965	0.6740	0.7512	**0.8285**
30	0.2081	0.2852	0.3631	0.4401	0.5180	0.5956	0.6737	0.7511	**0.8284**
40	0.2076	0.2837	0.3609	0.4389	0.5176	0.5945	0.6729	0.7501	**0.8275**
50	0.2062	0.2821	0.3600	0.4372	0.5169	0.5937	0.6717	0.7492	**0.8264**

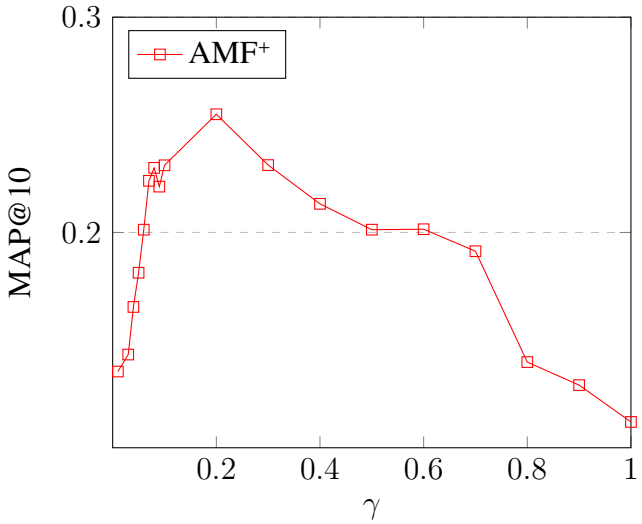

Figure (3.14) Influence of γ on MAP@10 of the proposed model.

Table (3.7) Influence of θ on AMF$^+$ performance.

	θ = 0.1		θ = 0.3		θ = 0.5		θ = 0.7		θ = 0.9	
N	NDCG@10	MAP@10	NDCG@10	MAP@10	NDCG@10	MAP@10	NDCG@10	MAP@10	NDCG@10	MAP@10
5	0.2066	0.0951	0.3038	0.0579	0.2411	***0.1252***	0.3679	0.1216	**0.3827**	0.1044
10	0.3016	0.0961	0.3138	0.0739	0.3051	***0.2222***	**0.3949**	0.1616	0.3847	0.1544
15	0.3886	0.1701	0.3238	0.1599	**0.3981**	***0.2302***	0.3959	0.1966	0.3857	0.1784
20	0.3996	0.2301	0.4008	0.2389	**0.4011**	***0.2312***	0.4009	0.2286	0.3987	0.2224
30	0.3486	0.2271	**0.3918**	0.1859	0.3361	***0.2282***	0.3299	0.2206	0.3737	0.1684
40	0.3126	0.1851	0.2928	0.1579	0.2511	0.1452	0.3169	***0.2156***	**0.3467**	0.1574
50	0.2916	0.1641	0.2038	0.1339	0.1551	0.0862	**0.2979**	***0.1836***	0.2817	0.1474

Table (3.8) Influence of regularization λ_S on model performance.

	λ_S = 0.01		λ_S = 0.05		λ_S = 0.1		λ_S = 0.5		λ_S = 1	
N	NDCG@10	MAP@10	NDCG@10	MAP@10	NDCG@10	MAP@10	NDCG@10	MAP@10	NDCG@10	MAP@10
5	**0.2671**	***0.1858***	0.2036	0.0509	0.2059	0.1622	0.1581	0.0859	0.2277	0.0564
10	**0.2961**	0.1782	0.2818	***0.1811***	0.2476	0.1219	0.2719	0.0636	0.2597	0.0804
15	0.3476	0.2001	0.3188	0.1859	**0.3791**	0.1606	0.3159	***0.2062***	0.3017	0.1524
20	0.3534	0.2102	0.3627	0.1928	**0.4011**	***0.2312***	0.3682	0.1928	0.3956	0.1878
30	0.3516	0.1791	0.3568	0.1539	0.3927	***0.2222***	0.3529	0.1826	**0.3471**	0.1814
40	**0.2621**	0.1141	0.2568	0.0999	0.2836	***0.1882***	0.3299	0.1796	0.3677	0.1034
50	0.2646	0.1011	0.1998	0.0189	0.2771	***0.1462***	0.3009	0.1166	**0.3237**	0.0684

we calculate NDCG@10 and MAP@10 at different number of context latent features $N = 5, 10, 15, 20, 50$. Table 3.7 shows the average NDCG@10 and MAP@10 of the settings for N. For NDCG@10, we use the train-set dataset with 20% mask of the interactions. For MAP@10, we use $n_{tr} > 2$ and $n_{te} = 1$. The highest results are in bold and most of them are at $\theta = 0.5$ and $\theta = 0.7$. We also see that the best scores are observed at $\theta = 0.5$ and N= 20. Hence we use $\theta = 0.5$ as for default setting in the experiment.

Influence of context latent features N and regularization λ_S We examine the influence of context regularization part to the model performance by tuning the regularization λ_S. We also test the influence of N to the model's performance at different regularization λ_S= 0.01, 0.05, 0.1, 0.5, 1. At each value of N= $5, 10, 15, 20, 30, 40, 50$, we calculate the NDCG@10 and MAP@10. For NDCG@10, we use the 20% mask dataset and for MAP@10 we use $n_{tr} > 4$ and $n_{te} = 1$

.

Influence of co-invocation regularization γ We also test the influence of co-invocation part to the model performance by tuning the co-invocation regularization γ. We set the value of $\gamma = \{0.01, 0.02, ..., 0.09, 0.1, ..., 0.9, 1\}$. For NDCG metric, we implement the 20% masked data and the performance results of our models by varying γ are plotted in Figure 3.13. Overall, the performance scores of AMF$^+$ sharply rise up from $\gamma = 0.01$ and peak at $\gamma = 0.1$. Later on, they steady decrease until $\gamma = 0.6$ and continue to decline more rapidly until the end point of $\gamma = 1$. This quick downturn means that the model is over-regularization and the performance is diminished. For the MAP metric, we use, we use the setting $(n_{tr} > 2, n_{te} = 2)$ and obtain the same patterns. It rapidly increases at $\gamma = 0.1$ to $\gamma = 0.2$, then gradually decreases and plateaus at $\gamma = 0.7$. As $\gamma > 0.7$, the performance quickly goes down due to over-regularization. Generally, we observe that our proposed model AMF$^+$ reaches the superior performance in both metrics at $\gamma = 0.1$ and diminish as $\gamma > 0.7$. It is understandable that a suitable value of γ would effectively integrate the regularization part into the model. However, a larger value of γ would degrade the model because of over-regularization.

3.5 Conclusion

In this chapter, we have enhanced the PMF model by learning the importance of mashup-API invocation through a MLP attentional network, which significantly improves the recommendation performance. We also studied the influence of context information and APIs' co-invocation relationships on the prediction. We found that pre-training document vectors by Doc2Vec to obtain the context similarity of services and conducting statistic analysis for APIs' co-invocation provide effective regularization to our AMF$^+$ model. Our proposed models AMF and AMF$^+$ obtains better results of NDCG and MAP than the conventional PMF and also outperforms some state-of-the-art baselines. Experimental results in various settings have validated the effectiveness of injecting the

context data and API co-invocation history into the proposed models.

In the future, we plan to enhance our methods with deeper models such as graph convolution neural network and auto-encoder architectures.

Chapter 4

Data Augmented High-order Graph Autoencoder in Service Recommendation

4.1 Introduction

The rapid development of service computing promotes the emergence of an abundance of Web services and APIs (APIs in short) on the Internet. The blooming of diverse APIs bring an expansive scope of choices for programmers to develop mashups, which are lightweight web-based apps that integrate multiple APIs. To facilitate the developers discover and select APIs effectively, recommender systems are currently a mainstream approach to service over-choice. Collaborative filtering (CF) (Sarwar et al., 2001a) is a popular technique that has been used to predict the preferred APIs for a mashup by exploiting the similarity patterns among mashup-API invocation (X. Chen et al., 2010; Hu et al., 2014; Xie, Wang et al., 2019; T. Liang et al., 2016; Xie, Chen et al., 2019).

Most of the existing CF-based models use Matrix Factorization (MF) (Mnih & Salakhutdinov, 2008; Koren, 2008; X. He, Zhang, Kan & Chua, 2016) to represents

the latent features of a mashup or an API as a vector and then models a mashup-API invocation as the inner product of their latent vectors (P. He et al., 2014; S. Li et al., 2017; Lo et al., 2012; H. Wu et al., 2018a). Other hybrid methods leverage CF and natural language processing (NLP) to add auxiliary information for the MF-based models, such as (C. Li, Zhang, Huai & Sun, 2014; Xiong et al., 2018; Yao, Wang, Sheng, Benatallah & Huang, 2021). Generally, these CF-based models produced good prediction for service recommendation by using their historical invocations. However, the increasing number of APIs with limited usage by mashups makes the invocation dataset more sparse. It reduces the effectiveness of existing service CF methods, which only consider the direct invocation between each pair of mashup and API. Such limitation is apparent in ProgrammableWeb[1] data.

With the prevalence of Deep Neural Networks (DNN), researchers introduced DNN components to matrix factorization and achieved superior prediction performance than traditional approaches. Most existing DNN-based service CF approachess (S. Liu & Zheng, 2020; Z. Liu, Guo, Wang, Du & Pang, 2019; Xiong et al., 2018) only consider the direct connections between mashups and APIs and do not consider the long-distance relationships and treat the user-item interaction or mashup-API invocation pairs as separate data instances (so called *information isolated island* in (X. Wang, He & Chua, 2020b)) and thus ignore the intrinsic relationships between invocations. In other words, such approaches assume unobserved invocations likely to be negative and thus overlook them from the observed or positive invocations. To overcome this limitation, recent DNN-based CF approaches (Y. Ma, Geng & Wang, 2020; Yao et al., 2021; G. Chen & Chen, 2015) augment the data by learning the high-order connectivity signals. For example, MISR (Y. Ma et al., 2020) leverages the high-order invocation to address the cold-start problem of service recommendation for new mashups.

Motivated by the discovery that the autoencoder architecture has the capability

[1]We download the database from https://dev.maxmind.com/in November 2020.

of forcing the hidden representation to capture information about the structure of the data generating distribution (Goodfellow, Bengio & Courville, 2016), and the recent developments in applying the autoencoder architecture to learn the non-linear graph structure (D. Wang, Cui & Zhu, 2016b), we propose a Data Augmented High-order Graph Autoencoder (DHGA) CF framework for service recommendation. Given the Mashup-API Invocation Graph (MAG), DHGA utilises the data augmentation technique to capture the direct and high-order connectivity, and injects such information to the embeddings of mashups and APIs autoencoders in a general CF-based framework. DHGA also integrates an MLP structure to contain the non-linear information between mashup and API.

In summary, this paper has the following main contributions:

1. We have used the data augmentation to enlarge the training dataset with the high-order connectivity, and proposed an autoencoder structure sharing the same parameters for the set of mashups or APIs.
2. We have proposed the DHGA, a novel data augmentation high-order collaborative filtering framework to integrate the intrinsic relation of the mashup-API graph by the autoencoder architecture.
3. We have conducted empirical studies on Programmable dataset and the results demonstrates the superior performance of DHGA over some state-of-the-art frameworks. We have studied how different order of connectivity and neighbourhood sample size in each order affects the performance of the DHGA framework.

The remaining of this paper is organized as follows. Section 4.2 is the related work; Section 4.3 defines the symbols and concepts used in this work and also discuss the data augmentation for the mashup-API graph; Section 4.4 presents the DHGA model in detail; Section 4.6 reports the experimental results and analysis; and Section 5.5 concludes the paper.

4.2 Related works

Existing works that are closely related to our work can be divided into three categories: *CF-based service recommender systems*, *neural networks for learning mashup-APU graphs*, and *data augmentation in recommender systems*.

4.2.1 CF-based service recommender systems

In recent years, the rapid development of machine learning research promotes collaborative filtering approaches being widely used in service recommendation. The CF-based models leverage the historical data of similar users or mashups to predict. For example, Zheng et al. (Zheng et al., 2012) proposes a neighborhood-integrated approach for recommending the personalized web service QoS. Some works are the neighbourhood integrated matrix factorization approaches such as Chen et al. (X. Chen et al., 2010) made use of the same region historical data to construct a neighbour-based Region KNN model, and Liu et al. (J. Liu et al., 2015) uses location information to define a location-aware CF model for QoS prediction.

In addition, there are some other types of CF-based models in service recommendation. For instance, Hu et al. (Hu et al., 2014) proposes a hybrid random walk model to calculate the similarity between mashups. Zou et al. (Zou et al., 2018) present a reinforced CF model with mashup-intensive and API-intensive which eliminates the interference of the mashup. Other works such as (Xie, Wang et al., 2019; T. Liang et al., 2016; Xie, Chen et al., 2019) use different types of auxiliary information of mashups and APIs to construct a heterogeneous information network (HIN) and calculate the their similarity scores for invocation prediction.

To tackle the limitation of using a prediction model from single mashup-API invocations, many other hybrid models exploit the context information and invocation history in higher order. Specifically, Jain et al. (Jain et al., 2015) and Samanta et al. (Samanta

& Liu, 2017) use the probabilistic model based approaches to calculate the correlation scores between mashups and APIs on the neighbour invocation under CF framework and then obtain the prediction value by using the product function. Such approaches only use the linear and product operator which has limitation on capturing the high-order and complex invocation between mashups and APIs.

4.2.2 Neural networks for learning mashup-API graphs

Early neural network based approaches (S. Liu & Zheng, 2020; Z. Liu et al., 2019; Xiong et al., 2018) has recently attracted more attention on service recommender systems. Specifically, Xiong et al. (Xiong et al., 2018) integrates with the historical mashup-API invocation and their content similarity in a DNN model. Chen et al. (L. Chen, Zheng, Feng, Xie & Zheng, 2018) utilises mashups and API feature vectors for recommendation process for a preference-based neural CF recommender system. Such approaches treat the graph as a set of edges of mashup-API invocation and thus ignore the intrinsic high-order connectivity in the graph. As to applying autoencoders, a recent method VAE-CF (D. Liang, Krishnan, Hoffman & Jebara, 2018) extends variational autoencoders for collaborative filtering (Kingma & Welling, 2013; Rezende, Mohamed & Wierstra, 2014), which consider the direct neighbourhood vector generated from a multinomial distribution. VAE-CF is limited to capturing up to second-order connectivity signals.

Our work is inspired by the success of autoencoders in graph embedding (D. Wang et al., 2016b). We tailor it for bipartite MAG and the problem of collaborative filtering. We also extract higher-order connectivity signals where $order$ is a hyperparameter.

4.2.3 Data augmentation in Recommender system

Data augmentation technique has been used in recommender systems. For example, HOP-Rec (J.-H. Yang, Chen, Wang & Tsai, 2018) uses the high-order proximity to

expand the train data to learn the implicit relation for recommender systems. However, such approach only considers the high-order relation between two types of nodes in the bipartite graph. The popular Bayesian Personalized Ranking (BPR) loss (Rendle, Freudenthaler, Gantner & Schmidt-Thieme, 2012) used in many recommender system approaches implicitly use data augmentation by considering the unobserved data as the negative. Data augmentation is also used in session-based recommendation such as Tan et al. (Tan, Xu & Liu, 2016) uses sequence reprocessing and embedding dropout to create more training sequences. A recent work MA-GNN (C. Ma et al., 2019) leverages a shared memory network to capture the long term relationships among items and draw out item pairs across all users. Another example in service recommendation Ma et al. (Yin et al., 2021) enhances the training set by drawing additional virtual data from the neighbouring relations with their different distances. Later on, the augmented data consists of factual and virtual invocations and used as the training set. However, such approach simply considers the negative invocations and the rich content of high-order connectivity is still unexplored.

Different from existing approaches, this paper enlarges the training dataset by the high-order neighbourhood data instances including the invocation and co-invocation data, which are fed in the autoencoder frameworks for a CF-based recommender system.

4.3 High-order connectivity and data augmentation for Mashup-API Graph

In this section, we first define some definitions used in this paper, and then propose the data augmentation technique to enhance the training dataset .

Table (4.1) Notations used in the paper

Notation	Definition
i, m	mashup index, $i, m \in \{1, ..N_m\}$
j, n	API index, $j, n \in \{1, .., N_a\}$
u_i	mashup i, representation vector of mashup i
v_j	API j, representation vector of API j
x_i, x_j	one-hot encoded invocation vector of u_i, v_j
r_{ij}	invocation between u_i and v_j, $r_{ij} = \{0, 1\}$
s_i, s_j	similarity vector of u_i, v_j; $s_i \in \mathbb{R}^{1\times N_a}$, $s_j \in \mathbb{R}^{1\times N_m}$
e_{ij}	edge between u_i and v_j, $e_{ij} \in E$ (set of edges)
MAG	Mashup API invocation Graph
$MashupAE$	Mashup autoencoder
$ApiAE$	API autoencoder
z	number of orders explored in the model
k	neighbours sample size within an order
d	autoencoder middle layer embedding size
$W_h, W_{h'}$	weight parameters for $MashupAE$
$Q_h, Q_{h'}$	weight parameters for $ApiAE$
b_h, b'_h, d_h, d'_h	layer biases for $MashupAE$ and $ApiAE$
$\mathcal{N}_i, \mathcal{N}_j$	direct neighbours of u_i, v_j
$f(2z, k)$	total number of sampled high-order neighbours
Θ_{MLP}	set of parameters for MLP part
Θ_u, Θ_v	high-order weights of mashup and API
N_X	size of the augmented training dataset
d_l, d_h	layer max dimension of autoencoder and MLP
I	number of iterations
N_r	number of APIs in the recommendation list
r_s	negative instance sample rate
p	top list ranking number used in evaluation metrics

4.3.1 Definitions

The definitions used for the proposed model DHGA are intepreted as below:

Mashup-API Invocation Graph (MAG) An MAG $G = (U, V, E)$ is a bipartite graph where U is the set of mashups, V is the set of APIs, and $E = U \times V$ with $e_{ij} \in E$ is an edge between U and V. Each edge e_{ij} is associated with a weight $r_{ij} \geq 0$. For u_i and v_j not linked by an edge, $r_{ij} = 0$.

First-order connectivity The first-order connectivity is the direct invocation between mashup i and API j and the value is defined by s_{ij}. In this paper, we consider r_{ij} as the

invocation between mashup i and API j.

Second-order connectivity The second-order connectivity is defined on a pair of mashups connected by the same API or a pair of APIs connected by the same mashup. It represents the similarity of the pair's neighbourhood architecture. Let e_{ik} and e_{mk} be two edges with mashup u_i and u_m connect to the same API v_k, and let s_i be the vector containing u_i's similarity with all the mashups $v_j(j = 1..|V|)$ in the MAG, then u_i and u_m has second-order connectivity and the value is defined by the cosine similarity of vectors s_i and s_m. A similar definition applies to a pair of APIs.

High/$(2z)^{th}$ order connectivity We can generalize the above definition to $(2z)^{th}$ order connectivity ($z \geq 1$) between a pair of mashups or APIs: if u_i can reach u_m in $2z$ hops on the MAG, or u_m is the $(2z)^{th}$ order neighbour of u_i which is denoted as $u_m \in N_i^{(2z)}$, then u_i and u_m has $(2z)^{th}$ order connectivity and the value is defined by the cosine similarity of s_i and s_m. Similar definition applies to a pair of APIs.

k-neighbour sampling At all high-order levels, the mashup or API can have no, one or more higher connections. In stead considering all connections, we only randomly sample a number k connections which is called ***k-neighbour sampling*** in this article.

Mashup Autoencoder (MashupAE) and API Autoencoder (ApiAE) An autoencoder is a feedforward neural network consisting two parts: one encoder and one decoder, with multiple non-linear functions. The encoder feeds the input invocation matrix into the representation space while the decoder recovers the representations back to the original network structure. Figure 4.3 shows the structure of MashupAE and ApiAE. Mashups and APIs have their own autoencoders to cater for the bipartiteness of the MAG. Thereafter, provided with input x_i, the hidden representations for each layer are defined as below:

For mashup:

$$u_i^{(1)} = \sigma(W_1 x_i + b_1)$$
$$u_i^{(h)} = \sigma(W_h u_i^{(h-1)} + b_h)$$
$$u'^{(h)}_i = \sigma(W'_h u_i^{(h-1)} + b'_h)$$
$$u'^{(1)}_a = \sigma(W'_1 \hat{x}_i + b'_1)$$

For APIs:

$$v_j^{(1)} = \sigma(Q_1 x_j + d_1)$$
$$v_j^{(h)} = \sigma(Q_h v_j^{(h-1)} + d_h)$$
$$v'^{(h)}_j = \sigma(Q'_h v_j^{(h-1)} + d'_h)$$
$$v'^{(1)}_j = \sigma(Q'_1 \hat{x}_j + d'_1)$$

4.3.2 Motivating Example

Figure 4.2 illustrates an example of the UIG with the Amazon-book dataset, which contains various orders of connectivity. The target is to predict the rating between the user u_i and the book item v_j. According to this example, u_i read the fashion book $v_{i(1,1)}$ and the cuisine book $v_{i(1,2)}$ so the user has the first-order connectivity with these items. Because the users $u_{i(2,1)}$, $u_{i(2,2)}$ also read the book $v_{i(1,1)}$, so they has second-order connectivity with u_i. All the users read the same book $v_{i(1,1)}$ might have the same interest on fashion, so they tend to read another fashion books in the futures. Hence, their second-order connectivity represents for such latent features. Likewise, the u_i has forth-order connectivity with $u_{i(4,1)}$. Specifically, if the book $v_{i(3,1)}$ is a beauty book, the user $u_{i(2,1)}$ might have some interest on beauty as the user $u_{i(4,1)}$ does. As the distance between u_i and $u_{i(4,1)}$ is larger than the distance between u_i and $u_{i(2,1)}$, u_i and $u_{i(4,1)}$

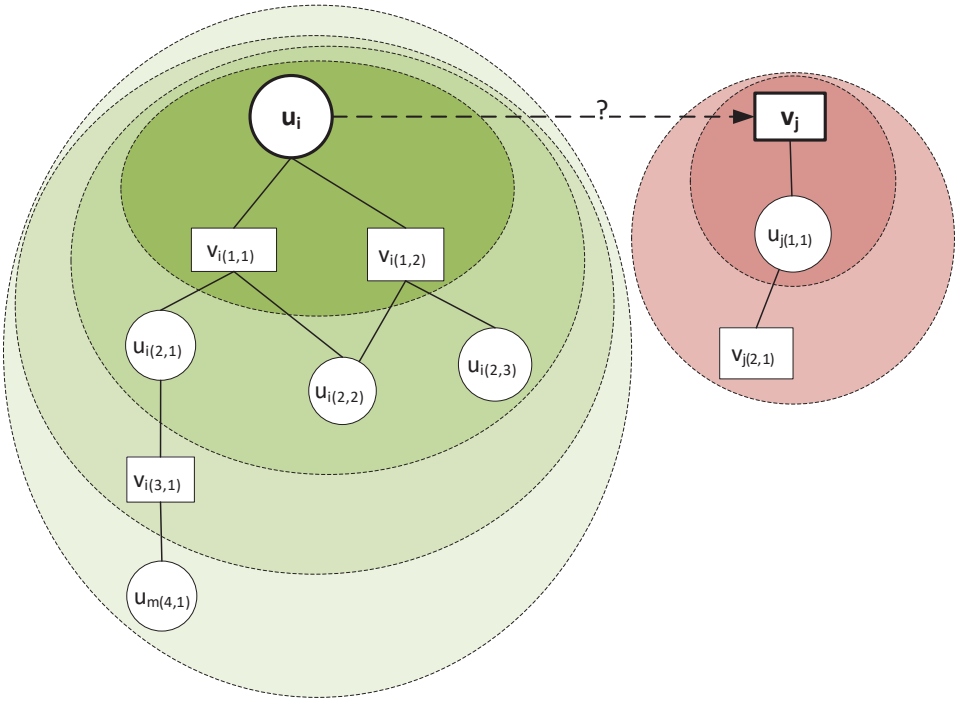

Figure (4.1) An illustration of a user-item interaction graph and the orders of connectivity

might share some similar features but not much as the pair u_i and $u_{i(2,1)}$ do. In this case, fashion and beauty are different but they are very close to each other. Hence, the user u_i who have read the fashion book will be probably interested in the beauty book and read such books in the future. In this example, exploring the high-order connectivity of a user by studying their high-order connected books with different similar topics can help us to predict the recommendation list of books for the user.

Similarly, for the book item v_j, it has first-order connectivity with $u_{j(2,1)}$ and second-order connectivity with $v_{j(2,1)}$.

In general, both direct and high-order interaction contain useful information. The high order connectivity generally contains implicit graph based relation among users and items, and injecting such information into the recommender system potentially improve the prediction for items recommendation list. In this work, we define the problem of recommendation system as below:

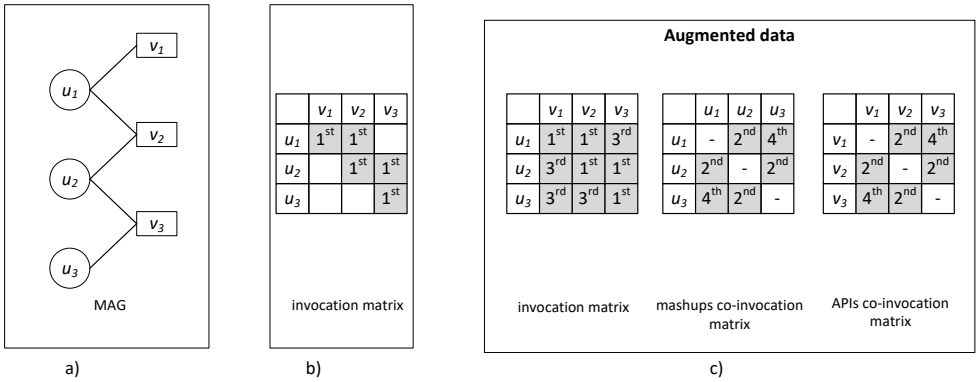

Figure (4.2) Direct and high-order connectivity within observed mashup-API invocations

Problem *How to leverage the intrinsic relation in the UIG by augmenting the data with high-order connectivity, and recommend the most appropriate list of items for each user?*

4.3.3 Data augmentation in MAG

Figure 4.2 illustrates an example of MAG with ***high-order connectivity*** and the invocation data used for recommender system. According to this figure, part (a) presents a bipartite graph built from positive invocations; part (b) shows the matrix with direct mashup-APU invocations (1^{st} order connectivity) which is normally used as input for CF-based recommender systems ; and part (c) is the augmented data including 3 matrices: one invocation matrix with 1^{st} and 3^{rd} order connectivity between mashup-API pair.

Generally, we consider both direct invocation and the intrinsic relation by exploring the high-order connectivity in the MAG, and embedding such relation into the augmented data. While the direct invocation data only considers the positive interactions, the augmented data includes both positive invocations and potential invocations inferred from the high-order connectivity between mashup-API pairs. Augmented data also contains the co-invocation relationship between pairs of mashups or APIs as the even

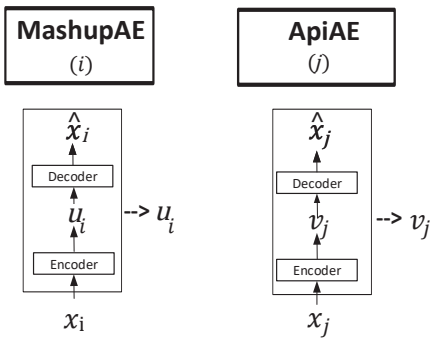

Figure (4.3) Mashup and API autoencoder

high-order connectivity. The number of instances in the augmented data are enhanced according to the high-order parameter z and k-neighbour sampling. It is defined as the process to randomly choose k neighbour mashups or APIs for each next order connectivity. Hence, the total instances in an MAG is calculated by function $f(2z, k)$ with $f(2z, k) = 1 + k + k^2 + ... + k^z$.

The augmented data now contains richer information than the simply direct invocation sparse matrix. In other words, it additionally exploits the high-order mashup-API invocations and co-invocation between pairs of mashups/APIs, which is believed to hold intrinsic relations in the MAG. Therefore, using such augmented data for training model would probably increase the performance for the service recommendation, and in this paper, we define the problem of improving service recommendation as below:

Problem *Given the MAG, how to uncover the intrinsic relation among data instance by using augmented data with high-order connectivity, and recommend the most appropriate of APIs for each mashup?*

The following section will demonstrate the proposed DHGA and how it leverages the augmented data to solve the defined problem.

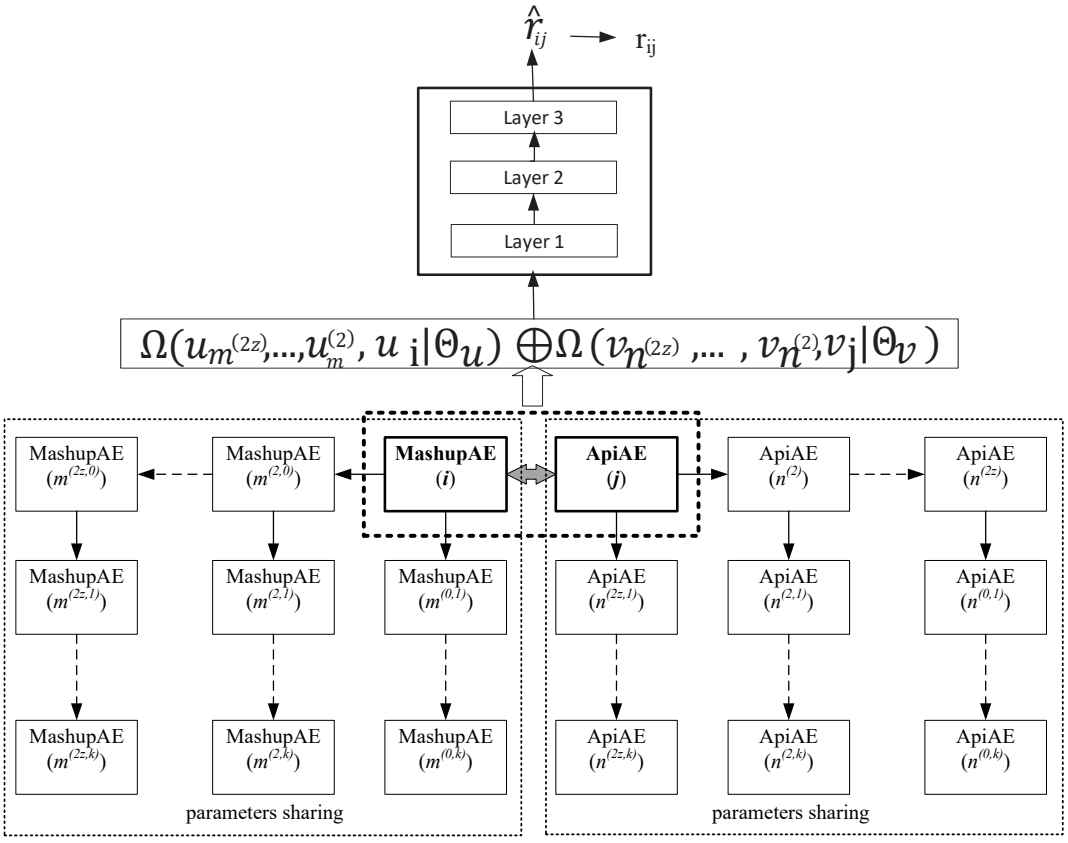

Figure (4.4) The $DHGA^{(2z,k)}$ model

.

4.4 The DHGA model

An illustration of the DHGA framework architecture is shown in Figure 4.4. The framework has four main components: 1) the autoencoders of target mashup i and API j; 2) the set of parameter-shared autoencoders $MashupAE$s for capturing high-order connectivity of mashup i; 3) the set of parameter-shared autoencoders $ApiAE$s for capturing high-order connectivity of API j; and 4) the concatenation and MLP for invocation prediction.

Specifically, For each mashup-API invocation, DHGA considers the $MashupAE$s of the mashup and its high-order mashup neighbours reachable from it to explore the mashup-mashup connectivity. All the autoencoders $MashupAE$s share the same set of parameters so that increasing the number of autoencoders does not increase the size of the model. The same setting also applies to $ApiAE$. The reason that two separate sets of autoencoders are used is to accommodate the bipartiteness of MAG: the initial embedding of a mashup contains its invoked APIs only and that of an API contains its invoked mashups only, which means they have different dimension and semantic meaning.

In addition, inspired by the effectiveness of the neural network framework (X. He et al., 2017c), we apply an MLP to capture the non-linear information in mashup-AP invocation. However, in DHGA, to facilitate capturing the high-order connectivity signals in the MAG, we first use the neighbourhood/invocation vector (a row or column in the MAG) to initialize the embedding of a mashup or API. Then, we augment a mashup-API invocation pair with their higher-order neighbour instances, and then feed them to two sets of autoencoders, Mashup Autoencoder ($MashupAE$) and API Autoencoder ($ApiAE$), to capture the graph structure information. $MashupAE$ and $ApiAE$ are defined in Section 4.3 and Figure 4.3.

4.4.1 Embedding layer with Autoencoders

As discussed, a mashup i or an APIs j is represented using a latent vector/embedding u_i or v_j ($u_i, v_j \in R^d$). In DHGA, d is the embedding size of the middle layer of an autoencoder. In early approaches such as matrix factorization (MF) and NeuMF, embeddings are only trained using the isolated invocation data instances. The recent NGCF approach propagates the embeddings on the MAG before achieving the prediction by the dot-product operation. DHGA learns the latent vectors by an autoencoder-based graph embedding method that maps the data to a highly non-linear latent space to retain the graph structure. Such autoencoder-based graph embedding has been demonstrated to be effective in injecting non-linear information into the latent representation and and also robust to sparse graph (D. Wang et al., 2016b).

While MF, NeuMF, and NGCF use one-hot encoded vectors as the input, DHGA uses the invocation vectors, or rows/ columns of the invocation matrix, as the input for the purpose of graph structure reconstruction. Therefore, $x_i = s_i$ and $x_j = s_j$.

4.4.2 High-order connectivity and data augmentation

With the representations developed by the first-order connectivity, we can then enhance the model capability by first augmenting each mashup-API pair with their higher-order neighbours and then feed the augmented data to a set of autoencoders. Specifically, as shown in Figure 4.4, a $(2i)^{th}$ order ($i = 1..z$) neighbour of the mashup, which is also a mashup, is sampled by randomly walking the paths in MAG and then feed to an autoencoder dedicated to this order. Clearly, z is a hyperparameter that indicates how deep, or how many orders, the model wants to explore, and we can also use another hypterparameter k to indicate how many neighbours we want to sample within a certain order. These two hyperparameters will be studied in the Experiments section. As to the set of autoencoders $MashupAE$, it contains one autoencoder for the mashup and one

autoencoder for each $(2i)^{th}$ order (i = 1..z) neighbourhood of the mashup. The above setting also applies to the API. It is worth noting that all autoencoders in $MashupAE$ or $ApiAE$ share the same set of parameters to reflect that all the nodes in one set share the same embedding space.

While NGCF stacks multiple embedding propagation layers to pass the message through different hops of neighbours, DHGA aggregates the high-order embeddings by a weighted average function $\Omega(u_{m^{(2z)}}, ..., u_{m^{(2)}}, u_i|\Theta)$. Θ could be Θ_u or Θ_v, which is the sequence of weights that reflects the importance of each order of connectivity. As the relationship strength decays with the order/distance of connectivity, simple exponential decay or the graph Laplacian norm
$1/\sqrt{|\mathcal{N}_i||\mathcal{N}_{m^{(2z)}}|}$, where $\mathcal{N}_i$ denotes the direct neighbours of mashup i, can be used to set Θ.

After that, DHGA use concatenation to aggregate the high-order embeddings of mashup i with that of API j before loading into the MLP part as
$\Omega(u_{m^{(2z)}}, ..., u_{m^{(2)}}, u_i|\Theta_u) \oplus \Omega(v_{n^{(2z)}}, ..., v_{n^{(2)}}, v_j|\Theta_v)$.

Finally, The invocation of mashup i on API j is predicted as below:

$$\hat{r}_{ij} = MLP(u_i, v_j|\Theta_{MLP}) \tag{4.1}$$

4.4.3 Optimization

To learn the DHGA model, there are two objectives. The first one is to minimize the reconstruction error of the mashup and API invocation vectors. Therefore, we define two loss functions as follows:

Loss function for $MashupAE$

$$Loss_1 = \sum_{i \in U} \| \hat{x}_i - x_i \|_2^2 \tag{4.2}$$

Loss function for $ApiAE$

$$Loss_2 = \sum_{j \in V} \| \hat{x_j} - x_j \|_2^2 \tag{4.3}$$

.

The second objective is to minimize the binary cross-entropy loss which has been used frequently in recommender systems (X. He et al., 2017c). It examines the difference between the real and predicted mashup-API invocation. It assumes that the positive invocations should be assigned higher prediction value than the negative invocations. Hence, the third loss function is defined as:

$$Loss_3 = - \sum_{i \in U, j \in I} r_{ij} \log \hat{r}_{ij} + (1 - r_{ij}) \log (1 - \hat{r}_{ij}) \tag{4.4}$$

This objective function is the binary cross-entropy or log loss which tackles recommendation with implicit data as a binary classification problem. Similar to NeuMF, NGCF randomly samples the negative invocations with uniform distribution under a controlled ratio. For simplicity, we choose the ratio to be 1:1 for negative sampling, which means that one negative invocation is sampled for one positive invocation. Although different sample ratio and non-uniform sampling might lead to better performance, we leave the study for future work.

The DHGA$^-$ model

The autoencoder-based embedding method for a homogeneous graph where all the graph nodes belong to a single type (D. Wang et al., 2016b) uses Laplacian Eigenmaps to preserve the first-order connectivity and uses autoencoders to reconstruct the neighbourhood vectors of the two interacting nodes to capture the second-order connectivity. To further study the effectiveness of the multiple-dimension connectivity in DHGA, we propose another model DHGA$^-$ which uses the $MashupAE$ and $ApiAE$ for the

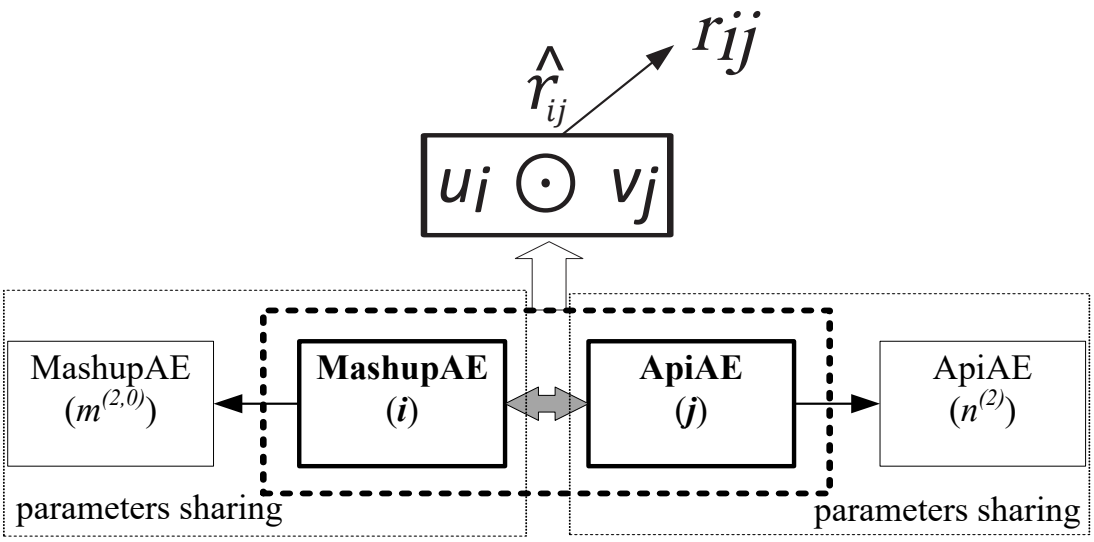

Figure (4.5) The DHGA⁻ model for service recommender system

second-order connectivity only with a dot product function to obtain the predicted invocation. DHGA⁻ does not consider the k-neighbour sampling. In addition, the significant difference is that the DHGA⁻ calculates the prediction value directly from the autoencoder embeddings of the mashup u_i and API v_j while the DHGA learns it from the autoencoder embeddings of u_i, v_j, and their high-order neighbours through an MLP part.

As shown in Figure 4.5, we extend the autoencoder-based embedding and apply it on the bipartite MAG for collaborative filtering. In such extension, the inner-product based mashup-API matrix factorization is extended with two autoencoder frameworks: one for mashup nodes and the other for API nodes. Clearly, such extension can capture connectivity signals up to the second order.

Time complexity The time complexity of the proposed DHGA is $O(N_X d_l d_h I)$, where N_X is the size of the augmented training dataset, d_l is the maximum dimension of the hidden layer of the autoencoders, d_h is the maximum dimension of the hidden layer of MLP, and I is the number of iterations. N_X is related to N_E, the number of edges of the MAG, by $N_X = (r_s + 1) f N_E$, where r_s is the negative instance sample

rate and f is the number of the sampled high-order neighbours of the target mashup or API. f is defined as: $f(2z, k) = 1 + k + k^2 + ... + k^z = O(k^z)$, where $2z$ is the maximum order of connectivity to reach and k is the number of neighbours to sample for each order. Although N_X turns out to be exponential to the order of connectivity, practically speaking we found that DHGA could achieve high performance even with $z = 1$ and $k = 1$ (c.f. Section 4.6).

4.5 Discussion

In this section, we compare existing autoencoder based approaches from the basic autoencoder model to denoising autoencoder CDAE (Y. Wu, DuBois, Zheng & Ester, 2016), and variational autoencoder VAE-CF (D. Liang et al., 2018) with our proposed high-order autoencoder based model. Figure 4.6 shows the structures of such autoencoder frameworks. The framework (a) is the orginal autoencoder to reconstruct the input x into $\hat{x}$. The denoising CDAE (b) adds the noise to the input x and simply uses the data augmentation with a portion of the negative samples as the implicit data. Such work only considers the directed relation and ignore the intrinsic graph structure. Our approach does not only uses the negative instances but also explore the high-order interactions with both negative and positive observes. This augmentation technique makes our model more effective with larger and deeper data. CDAE uses three-layer neural network to learn the embedding of users. Different form CDAE, our model applies the autoencoder framework for the bipartite graph with two types of nodes including users and items. In addition, such autoencoder frameworks combine together according to the interaction graph structure. VAE-CF (c) generally uses an parametrized inference model to learn the mean and variance and inject to the autoencoder part. Our proposed high-order autoencoder does not only execute the autoencoder of a unique input but also embed the autoencoder of the neighbour in high-order connectivity. For example

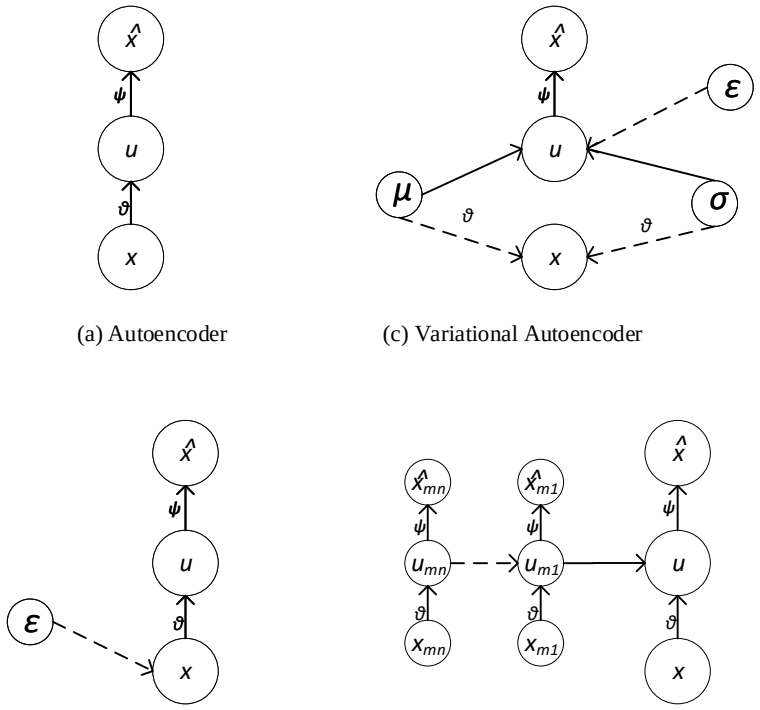

Figure (4.6) Discussion of different Autoencoder models

in Figure 4.6 (d), we consider the m neighbours of the user u from the second-order u_{m1} to the user u_{mn}. Such high-order information is injected to the intermediate hidden representation and support for the main autoencoder of the user u.

The proposed model DHGA with high-order autoencoder can be applied to any bipartite networks/graphs where two types of nodes connect to each other. There are many networks in the real world are bipartite including the user-item graphs used in this work, the Mashup-WebAPI networks in the service computing domain (Nguyen, Yu, Nguyen & Han, 2021) and the bipartite ecological (Dormann, Gruber & Fründ, 2008) networks. While in this paper we applied our model on the recommendation task, DHGA can also be applied in the general link prediction tasks and other prediction tasks such as node classification.

The assumption of this work is that the graph structure, including both direct/first-order connectivity and high-order connectivity brings more good hints than false positive

information. Such assumption first is validated by the experiments done in this work which shows a superior performance than past works. On the other hand, the sparsity of most real-world networks might also suggest that people tend to create links in a network between highly relevant nodes while ignoring links between irrelevant nodes. A recent work FAGCN (Bo, Wang, Shi & Shen, 2021) can utilize both the assortative and disassortative connections of a graph. However, we leave this topic for future exploration.

4.6 Experiments

We have performed experiments on ProgrammableWeb dataset to evaluate our proposed model. Our objective is to answer the following research questions:

- RQ1: How does DHGA perform as compared with the state-of-the-art collaborative filtering methods?
- RQ2: How do *order of connectivity*, *number of mashups or APIs sampled in each order*, and *number of hidden layers of autoencoder* affect the performance of DHGA?

4.6.1 Experimental settings

Datasets We use the dataset of ProgrammableWeb to demonstrate the effectiveness of the proposed model. The dataset consists of textual descriptions of 17829 APIs and 6340 mash-ups, and their historical invocation. Because the dataset does not have the ratings between mashups and APIs, we adopt their invocation data as the ratings. For example, if mashup m_1 invokes API a_5, their invocation data is 1, and we use this value as invocation between this mashup- API pair. After that, we do reprocessing for the dataset as follows. We remove blank APIs and mashups, and obtain 5691 mashups and

Dataset	# Users	# Items	# Interactions	Density
Movielens 1M	6,040	3,706	1,000,209	4.7%
ProgrammableWeb	6,910	1170	10,788	0.13%
Gowalla	29,858	40,981	1,027,370	0.08%
Goodreads-books	25,475	18,892	1,378,033	0.29%
Amazon-books	52,643	91,599	2,984,108	0.06%

Table (4.2) Datasets statistic

1170 APIs. Them we remove a certain portion of the mashup-API invocations to create a training set, and the original data becomes the test set.

We further use Movielens 1M, Gowalla, Goodreads-book, and Amazon-book which are widely used by many collaborative filtering models. For Movielens 1M, we use the version of one million ratings in which each user has at least 20 ratings. This dataset originally is an explicit data, we transform it to implicit interactions. As a results, ratings are converted to 0 and 1 with 1 means that the user has rated the item and 0 means that the user has not rated the item. For Gowalla, users share their locations when doing check-in. This dataset has over one million ratings. We use the 10-score setting (R. He & McAuley, 2016) to assure the quality of dataset which means every user or item has at least 10 interactions. For Goodreads-book, this dataset is from the goodreads website with a focus on the genres of Children and Comics. In order to be consistent with the implicit feedback setting, we keep those with ratings no less than four (out of five) as positive feedback and treat all other ratings as missing entries on all datasets. The Amazon-Book has about three million ratings and use the 10-score setting for each user or item having at least 10 interactions.

Evaluation metrics To evaluate the performance of API recommendation, we use binary accuracy, hit ratio (HR) (X. Wang et al., 2019), Normalized Discounted Cumulative Gain (NDCG), and Mean Average Precision (MAP). According to the common approaches (Tay et al., 2019; D. Wang et al., 2016b), we pick unused items for each mashup in the train dataset. After that, we sort them ascending to their estimated

ratings. According to recent methods he2017, wang2019, we truncate the top-p ranked for the recommendation list. Then HR is obtained from the top-p recommendation list using the following equation:

$$\text{HR@p} = \frac{\text{Number of hits } @p}{N_r} \tag{4.5}$$

Where N_r is the number of APIs in the recommendation list. From the ranking of relevant APIs in the recommendation list, we measure the effectiveness of recommendation by the Discounted Cumulative Gain (NDCG). The gain is accumulated from the top to the bottom of recommendation list for each relevant API. Accordingly, this metric discounts and penalizes the APIs which have lower rank. Namely, the NDCG from rank 1 to p for the subset of a recommendation list is defined as:

$$\text{NDCG@p} = \sum_{i=1}^{p} \frac{2^{r_i} - 1}{\log_2(i+1)} \tag{4.6}$$

For the Mean Average Precision (MAP), we calculate as follows. Precision is the portion of recommended items those are actually invoked by the user which is also known as true positive accuracy. We use $P@k$ as the precision that is calculated from only a recommendation list from rank 1 through k. Given an user u_i, the Average Precision $AP@k_u$ is the precision of each relevant item, regarding its position in the ranked recommendation list of k items: $AP@k_{(u_i)} = \frac{\sum_{n=1}^{k} Pr(n) \times I(n)}{k}$, where $Pr(n)$ is the precision at cutoff n in the list, and $I(n)$ is equal to 1 if the item at rank nth is actually invoked, zero otherwise. The AP metric identifies every relevant recommendation and give prize frontal recommendation with the most probable correct items, such as the relevant items at the top of the recommendation list. Hence, we call the arithmetic average of the AP@k of all users as the Mean Average Precision : $MAP@k = \frac{\sum_{i=1}^{N_u} AP@k_{u_i}}{N_u}$, where N_u is the number of users.

Training set-up All datasets are split into train set and test set with ratio 80% and 20% respectively. We use the train set to train the proposed DHGA model and evaluate the binary accuracy, HR and NDCG on a test set. We do not use the validation set labels for training. For the evaluation process, all the mashups in the test set have a list of invoked APIs. For each mashup, all APIs in the test set are scored with their estimated invocation value which is between 0 and 1. After that, we sort them ascendingly to their estimated ratings. According to recent methods (X. He et al., 2017c; X. Wang et al., 2019), we truncate the top-p ranked for the recommendation list, Then, we get the top p highest score list which will be compared with the test set to see if the APIs are in the test or not. Consequently, the number of APIs are in both top p list and test set is the number of hits. For evaluation metrics, we use p=10 for our experiment.

We use Keras Tensorflow to implement DHGA and some of the baselines. The embedding size is fixed to 32 for the baselines NeuMF, $\text{MF}_{logloss}$, NGCF, and VAE-CF. We use grid search to tune the hyperparameters. Namely, the learning rate is tuned in {0.0001, 0.0005, 0.001, 0.005, 0.01}, the coefficient of L2 normalization in {10^{-6}, 10^{-4}, 10^{-2}, 10^{1}, 10^{2} and the dropout ratio in {0.1, 0.2, 0.3, ..., 0.8}. We use Adadelta optimizer with batch size 256. We use five layers with dimensions 128-64-32-64-128 for the autoencoder and three layers with the tower structure for the MLP. For the hyperparameters of the baselines, the configuration of the best performed version as reported are used. For example, NGCF-3 (X. Wang et al., 2019) is used as our baseline. Furthermore, early stopping is applied and the results of training phase shows that the performance of all versions of DHGA converges at around epoch 100. Specifically, Figure 4.7 shows the score of Accuracy over the first 100 epoches when training the ProgrammableWeb dataset of the baselines and DHGA.

Baselines We compare DHGA with the following baselines:

- **MF** (Mnih & Salakhutdinov, 2008): This is the vanilla matrix factorization

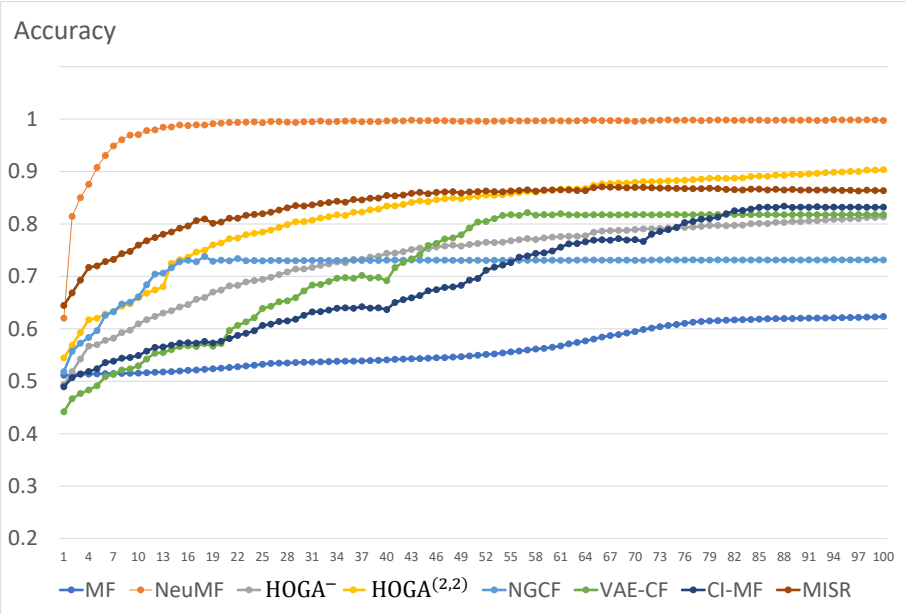

Figure (4.7) Accuracy scores of all models on train dataset in 100 epoches

method that represents users and items as feature vectors and exploits their direct invocations by inner product.

- **NeuMF** (X. He et al., 2017b): This is a neural collaborative filtering model that uses multiple hidden layers above the element-wise and concatenation of mashup and API embeddings to capture their nonlinear feature invocations.

- **NGCF** (X. Wang et al., 2019): This is a state-of-the-art GCN-based collaborative filtering method that propagates embeddings on the MAG to exploit the high-order connectivity signal of users and items.

- **VAE-CF** (D. Liang et al., 2018): This is a state-of-the-art method that applies Variational Autoencoders on collaborative filtering for implicit data with multinomial likelihood and Bayesian inference to tune the parameters.

- **CI-MF** (Yao et al., 2021): Matrix Factorization based model with API Co-invocation regularization, which uses TF-IDF for text mining.

- **DHGA**$^{-}$: This is the model that extends SDNE for a bipartite graph to obtain the embeddings of users and items and then uses dot product to make prediction for their invocation.

- **MISR** (Y. Ma et al., 2020): This is a multiplex invocation-oriented service recommendation model which leverages three types of invocations between services and mashups and incorporates them into a DNN to present for their explicit and implicit relationships.

Variants of DHGA We use (n, k) to distinguish DHGA variants with n denoting the highest order of connectivity to reach and k the number of high-order neighbours to sample in a specific order. For example as below:

- DHGA$^{(1)}$: 1^{st} order connectivity including all the direct invocations/edges in the MAG

- DHGA$^{(2,1)}$: 2^{nd} order connectivity with one sampled mashup/API in the $2nd$ order

- DHGA$^{(2,2)}$: 2^{nd} order connectivity with two sampled mashup/API in the $2nd$ order

- DHGA$^{(4,1)}$: 4^{th} order connectivity with one sampled mashup/API in the $2nd$ order

- DHGA$^{(4,2)}$: 4^{th} order connectivity with two sampled mashup/API in the $2nd$ order

- DHGA$^{(4,2)}$: 4^{th} order connectivity with two sampled mashup/API in the $2nd$ order

In this paper, we test the variants of DHGA with $2z$ = $\{0, 2, 4, 6\}$ and k = $\{1, 2, 3, 4, 5\}$

Table (4.3) Baseline comparison over different p on HR metric

p	1	2	3	4	5	6	7	8	9	10
MF	0.1312	0.3401	0.4801	0.5121	0.5143	0.5201	0.5201	0.5201	0.5201	0.5201
NeuMF	0.185	0.3811	0.5181	0.5534	0.5769	0.6001	0.6079	0.6079	0.6079	0.6079
NGCF	0.2091	0.4429	0.5883	0.6201	0.6245	0.6312	0.6312	0.6312	0.6312	0.6312
VAE	0.2891	0.4592	0.5923	0.6242	0.6392	0.6723	0.6933	0.6933	0.6933	0.6933
CI-MF	0.3023	0.4817	0.6003	0.6519	0.6703	0.6889	0.6991	0.6994	0.6994	0.6996
DHGA$^-$	0.4521	0.6215	0.6509	0.6821	0.7121	0.72	0.7201	0.7212	0.7223	0.7225
MISR	0.4815	0.6494	0.6760	0.7023	0.7211	0.7286	0.7392	0.7392	0.7407	0.7441
DHGA(2,2)	0.4960	0.6645	0.7190	0.7367	0.7483	0.7547	0.7603	0.7604	0.7606	0.7616

4.6.2 Performance comparison (RQ1)

This paper uses the defined metrics Accuracy, HR, and NDCG to compare the performance of all baselines and DHGA variants.

Comparison results on ProgrammableWeb

Overall comparison Both DHGA$^{(2,2)}$ and DHGA$^{(4,2)}$ obtain the best performance among the variants (C.f. Section 4.6.3) with the similar scores, we only choose DHGA$^{(2,2)}$ to compare with the baselines. Table 4.6 shows the comparison result. As we can see, DHGA$^{(2,2)}$ obtains the best results. The MISR leverages the high-order invocation also achieves very good results which is only slightly less than DHGA$^{(2,2)}$. All the autoencoder-based methods, including VAE-CF, DHGA$^-$ also achieves better HR and NDCG than other methods, which may demonstrate the merits of applying autoencoders in collaborative filtering. Both DHGA$^{(2,2)}$ and VAE-CF capture connectivity up to the second order. DHGA$^{(2,2)}$ is better performed than VAE-CF may because VAE-CF only captures the 2^{nd} order connectivity of APIs, while DHGA$^{(2,2)}$ considers the connectivity of both mashups and APIs.

Top-p recommendation comparison We test the performance of top-p recommendation lists of all models and use HR@10 and NDCG@10 as measurement metrics. The values of p is in range $[1, .., 10]$. Table 5.1 and Table 5.2 respectively show the HR@10 and NDCG@10 scores of all models with different top-p recommendation.

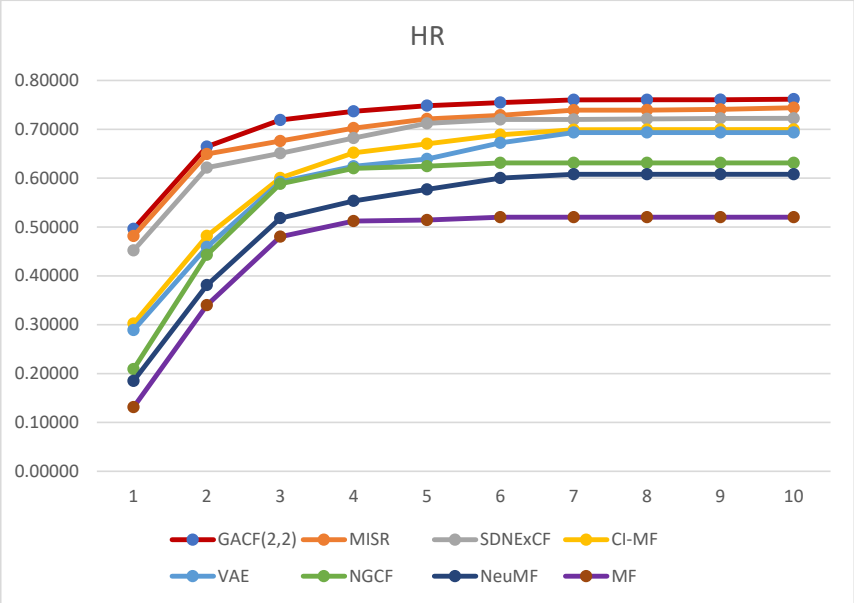

Figure (4.8) Impact of different p on HR@10 metric

Table (4.4) Baseline comparison over different p on NDCG metric

p	1	2	3	4	5	6	7	8	9	10
MF	0.1301	0.3078	0.337	0.3371	0.3367	0.3367	0.3367	0.3367	0.3367	0.3367
NeuMF	0.1521	0.3455	0.3827	0.4260	0.4459	0.4459	0.4459	0.4459	0.4459	0.4459
NGCF	0.1921	0.3521	0.3841	0.4393	0.4647	0.4678	0.4580	0.4603	0.4613	0.4647
VAE	0.2821	0.3921	0.4328	0.4528	0.4921	0.4921	0.4921	0.4921	0.4921	0.4921
CI-MF	0.2888	0.4526	0.4684	0.4773	0.4931	0.5067	0.4959	0.4983	0.4959	0.4938
DHGA^{-}	0.4501	0.5071	0.5151	0.5317	0.5064	0.5132	0.5115	0.5150	0.5161	0.5196
MISR	0.4605	0.5209	0.5050	0.5343	0.5301	0.5351	0.5433	0.5569	0.5568	0.5583
DHGA(2,2)	0.4960	0.5626	0.5666	0.5627	0.5657	0.5711	0.5716	0.5718	0.5725	0.5726

As can be seen, DHGA$^{(2,2)}$ obtains consistent improvement over the other models all values of p. It is also clear that the DHGA$^{(2,2)}$ achieves the converged results in smaller p than other baselines. Specifically, there are two groups of models. The first group including MF, NeuMF, NGCF, VAE, and CI-MF have smaller performance results at the small values of p ($p = 1, 2, 3$). The remaining models DHGA^{-}, MISR, and DHGA$^{(2,2)}$ are in the second group and have much higher performance at the beginning. The second group also achieve the converge value at earlier stage, namely at $p = 4$ for both HR and NDCG metrics while the first group approximately cannot reach the highest value until $p = 6$.

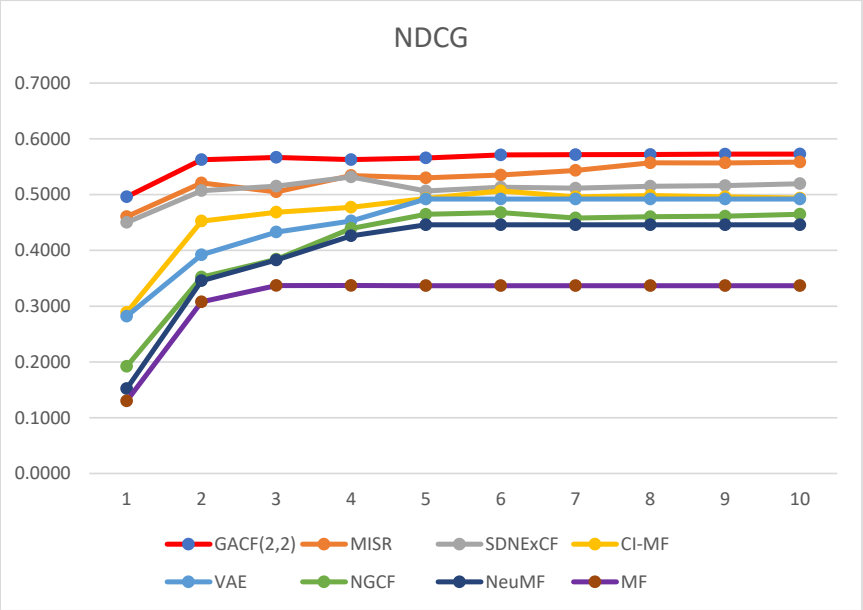

Figure (4.9) Impact of different p on NDCG@10 metric

Table (4.5) Overall performance comparison on ProgrammableWeb

Model	Accuracy	HR@10	NDCG@10
MF	0.6395	0.5201	0.3367
NeuMF	0.6467	0.6079	0.4558
NGCF	0.6521	0.6312	0.4647
VAE-CF	0.6829	0.6933	0.4921
CI-CF	0.7012	0.6996	0.4938
DHGA^{-}	0.7193	0.7225	0.5196
MISR	0.8021	0.7597	0.5683
DHGA$^{+}(2,2)$	**0.8319**	**0.7672**	**0.5726**

Comparison results on Movielens 1M, Gowalla, Goodreads-book, and Amazon-book

For such datasets, we use MAP and NDCG to compare the performance of all baselines.

Overall comparison Table 4.6 shows the comparison result. As DHGA$^{(2,2)}$ obtains the best performance among the four versions, we use it to compare with the baselines. In Figure 4.10 and Figure 4.11, all the autoencoder-based methods, including VAE-CF, DHGA^{-}, and DHGA achieve better MAP and NDCG than other methods, which may demonstrate the merits of applying autoencoders in collaborative filtering. Both

	Movielens 1M		Goodreads-book		Gowalla		Amazon-book	
Model	MAP@10	NDCG@10	MAP@10	NDCG@10	MAP@10	NDCG@10	MAP@10	NDCG@10
NeuMF	0.4741	0.4349	0.5743	0.6199	0.5649	0.6017	0.5614	0.6071
$MF_{logloss}$	0.4654	0.4523	0.5699	0.6003	0.5593	0.6005	0.5509	0.6074
NGCF	0.5203	0.5966	0.5545	0.6102	0.5832	*0.6271*	0.5628	0.6111
VAE-CF	0.6002	0.6301	0.5855	0.6139	*0.5887*	0.6198	*0.5783*	0.6128
$DHGA^{-}$	*0.6012*	*0.6793*	*0.6019*	*0.6508*	0.5743	0.6211	0.5643	*0.6204*
$DHGA^{(2,2)}$	**0.7522**	**0.7956**	**0.6209**	**0.6998**	**0.6125**	**0.6496**	**0.5872**	**0.6424**

Table (4.6) Overall performance comparison

	Movielens 1M		Goodreads-books		Gowalla		Amazon-books	
Model	MAP@10	NDCG@10	MAP@10	NDCG@10	MAP@10	NDCG@10	MAP@10	NDCG@10
$DHGA^{(0,1)}$	0.7207	0.7524	0.6222	0.6882	0.6075	0.6405	0.5854	0.6256
$DHGA^{(2,1)}$	0.7445	0.7884	0.6270	0.6886	0.6100	0.6451	0.5870	0.6266
$DHGA^{(2,2)}$	**0.7522**	**0.7956**	0.6209	**0.6998**	0.6125	**0.6496**	**0.5872**	**0.6424**
$DHGA^{(4,1)}$	0.7011	0.7712	**0.6306**	0.6946	**0.6167**	0.6375	0.5802	0.6421
$DHGA^{(4,2)}$	0.7001	0.7634	0.6301	0.6911	0.6131	0.6329	0.5809	0.6416

Table (4.7) Test performance of different versions of DHGA

$DHGA^{(2,2)}$ and VAE-CF capture connectivity up to the second order. $DHGA^{(2,2)}$ is better performed than VAE-CF may because VAE-CF only captures the 2^{nd} order connectivity of users, while $DHGA^{(2,2)}$ considers the connectivity of both users and items.

4.6.3 Study of DHGA (RQ2)

In this section, we evaluate the performance of DHGA at different high-order of connectivity and number of k-neighbour sampling in each order.

To train the models, we use Keras Tensorflow with cross-entropy loss function and binary accuracymetric [1]. Then, we evaluate the models on the test dataset with metrics Accuracy, HR, and NDCG. The next subsections present the results over different values of $2z$ high-order, k-neighbour, and number of autoencoder hidden layers.

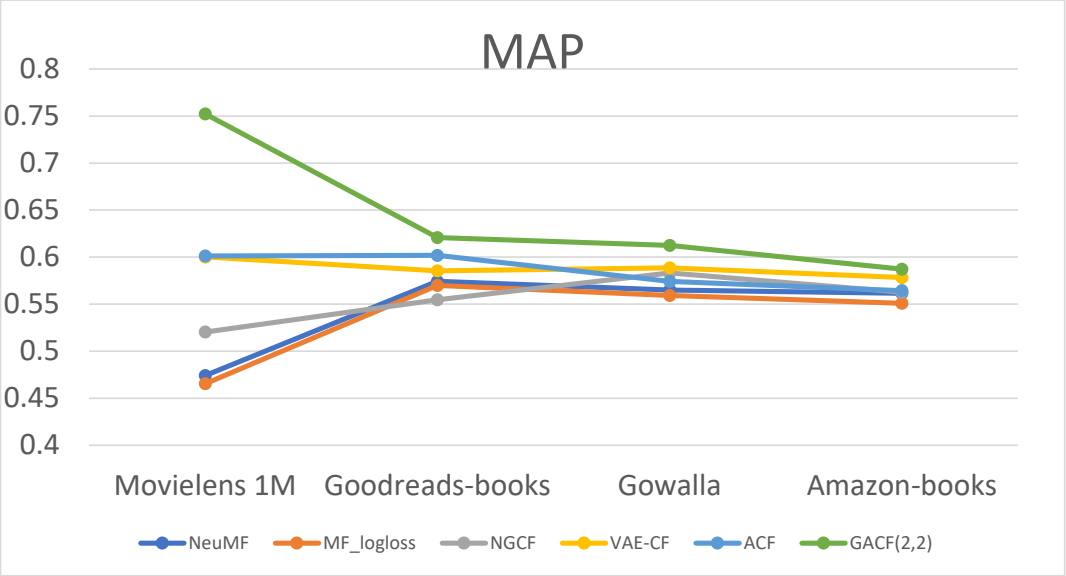

Figure (4.10) Test performance of baselines and DHGA$^{(2,2)}$ on MAP

.

Study results on ProgrammableWeb

Influence of high order and k-neighbour sampling. The high-order and k-neighbour sampling significantly impact the graph autoencoder output and then influence the DHGA model performance. We train a number of DHGA variants and observe the prediction results on the test dataset of these models. We choose the variants with $2z = \{0, 2, 4, 6\}$ and $k = \{1, 2, 3, 4, 5\}$. We train these models with train dataset in 100 epoches. After that, we test their performance on the test dataset with three evaluation metrics including Accuracy, HR@10, and NDCG@10. The Figure 4.12 shows the results of this study.

Specifically, for Accuracy metric, the scores of all models are from 0.72 to 0.82. So, the difference between the best and the least is not much about 0.1. As we can see from the 3B accuracy bar chart, the better score models are in the middle which has the value of $2z = \{2, 4\}$ and $k = \{2, 3, 4\}$. It is also remarkable that DHGA$^{(2,2)}$ and DHGA$^{(4,3)}$ have the best scores and DHGA$^{(6,5)}$ has the smallest performance. Therefore, we can

[1]https://www.tensorflow.org/apidocs/python/tf/keras/metrics/BinaryAccuracy

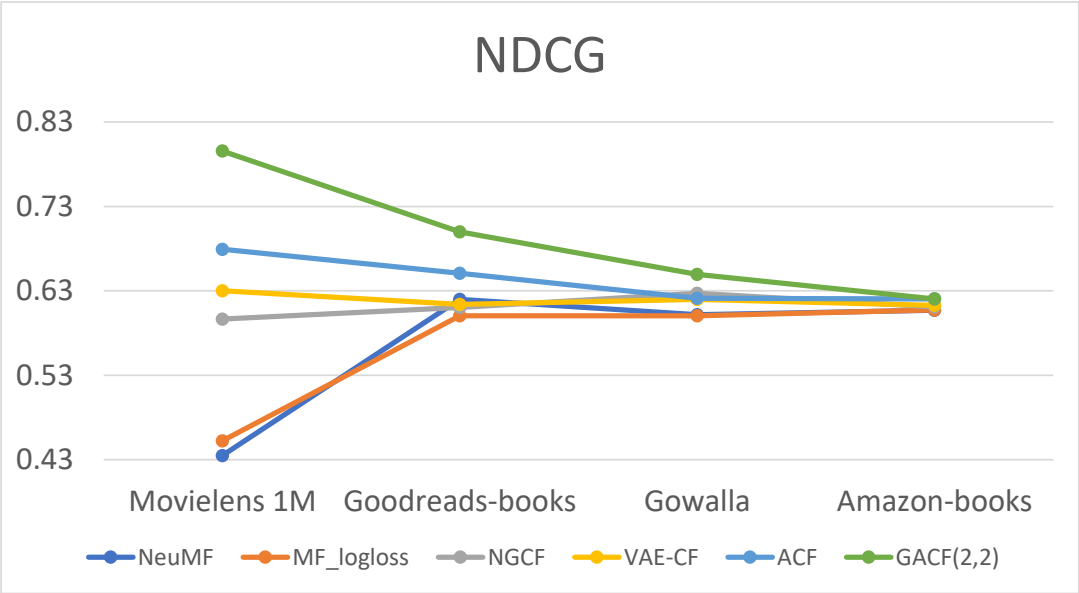

Figure (4.11) Test performance of baselines and DHGA$^{(2,2)}$ on NDCG

.

learn that the performance of DHGA will increase when we increase the high-order and neighbour sampling size at a certain value for example $2z = 4$ and $k = 3$, and then it starts to decrease when these parameters continue to increase. It may be due to the data overfitting in the train dataset distorts the prediction results in the test dataset. These results are similar to HR and $NDCG$ bar charts.

Overall, the variant without high-order DHGA$^{(1,k)}$ has the least performance while the higher order models achieve better scores. The 2th and 4th high-order achieve the best performance and the 6th high-order is not as good as the 2th and 4th but it outperforms the 1st high-order variants DHGA$^{(1,k)}$.

Study results on Movielens 1M, Gowalla, Goodreads-book, and Amazon-book

The key part of the proposed DHGA^{-} is the high order connectivity with data augmentation part. So, we study how the related parameters $2z$ and k influence the performance of the model. Specifically, we evaluate the performance of the five versions of DHGA^{-}

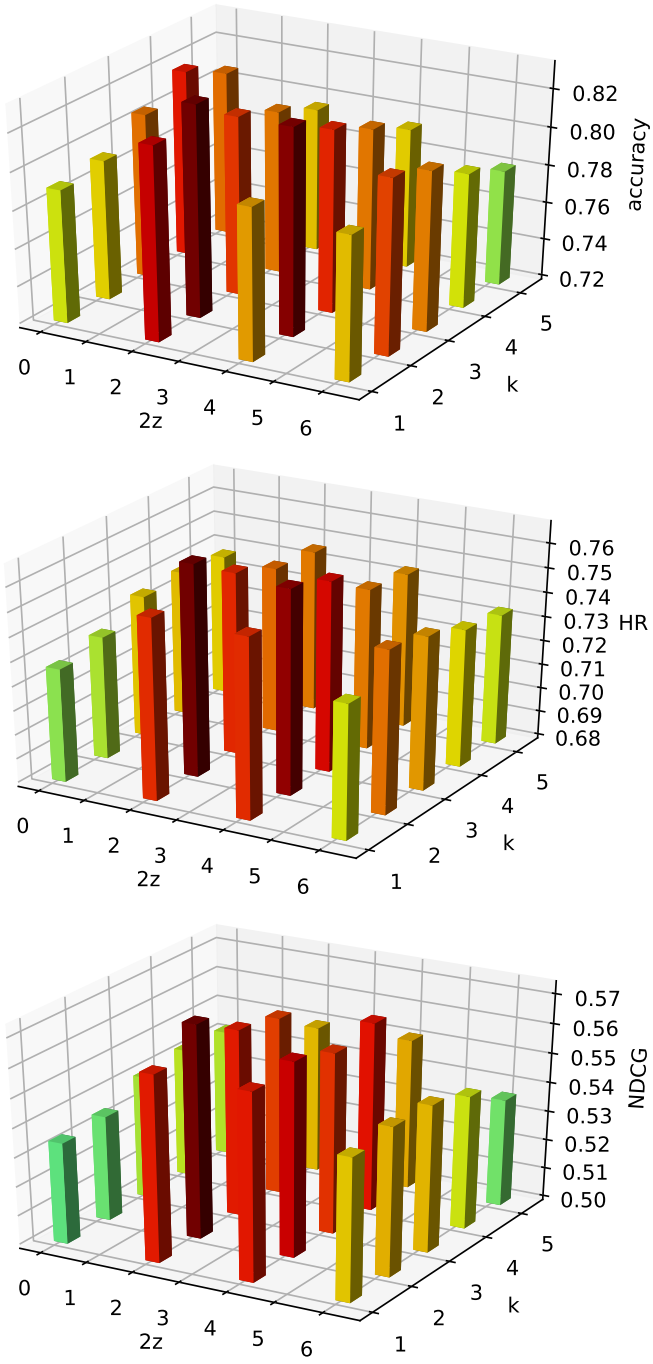

Figure (4.12) Performance of DHGA variants with different values of n and k

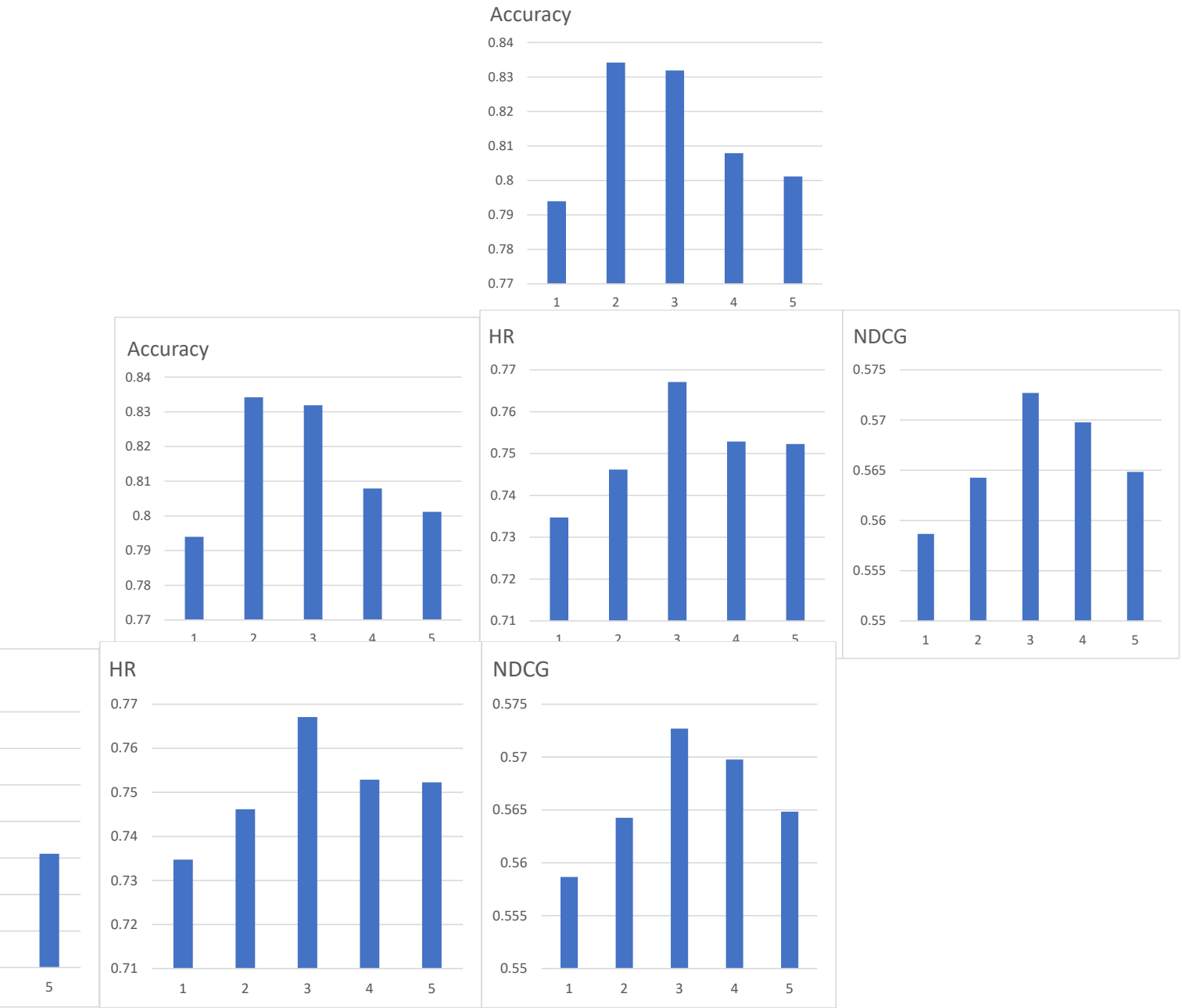

Figure (4.13) The influence of autoencoder hidden layers

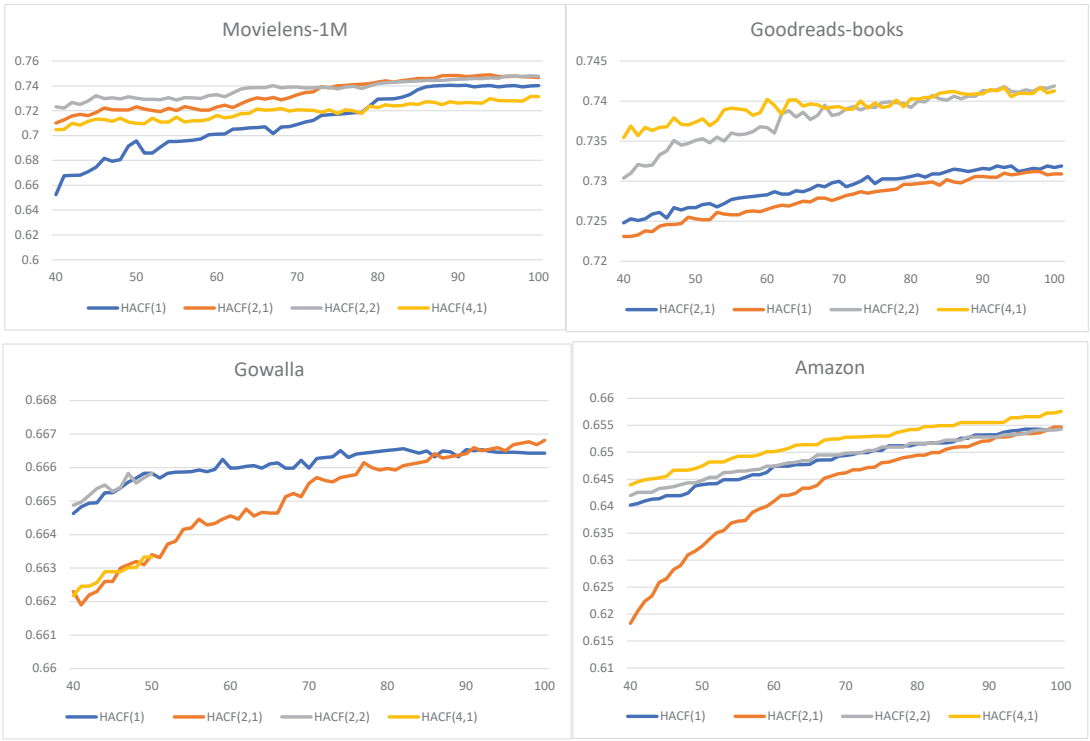

Figure (4.14) Binary accuracy convergence of the training datasets

.

Model	Movielens 1M		Goodreads-books		Gowalla		Amazon-books	
	MAP@10	NDCG@10	MAP@10	NDCG@10	MAP@10	NDCG@10	MAP@10	NDCG@10
HACF$^{(0,1)}$	0.7207	0.7524	0.6222	0.6882	0.6075	0.6405	0.5854	0.6256
HACF$^{(2,1)}$	0.7445	0.7884	0.6270	0.6886	0.6100	0.6451	0.5870	0.6266
HACF$^{(2,2)}$	**0.7522**	**0.7956**	0.6209	**0.6998**	0.6125	**0.6496**	**0.5872**	**0.6424**
HACF$^{(4,1)}$	0.7011	0.7712	**0.6306**	0.6946	**0.6167**	0.6375	0.5802	0.6421
HACF$^{(4,2)}$	0.7001	0.7634	0.6301	0.6911	0.6131	0.6329	0.5809	0.6416

Table (4.8) Test performance of different versions of HACF

on the test datasets. As we can see in Table 4.8, DHGA$^{(0,1)}$ has the lowest performance on both metrics. Then, DHGA$^{(2,1)}$ achieves better results and DHGA$^{(2,2)}$ has the best MAP and NDCG on all datasets. However, the performance of DHGA^{-} does not improve when the order reaches four. Specifically, the results of DHGA$^{(4,1)}$ are better than DHGA$^{(0,1)}$ and DHGA$^{(2,1)}$ but lower than DHGA$^{(2,2)}$. We can also see that the performance of DHGA$^{(4,2)}$ is lower than DHGA$^{(4,1)}$, which could be caused by overfitting.

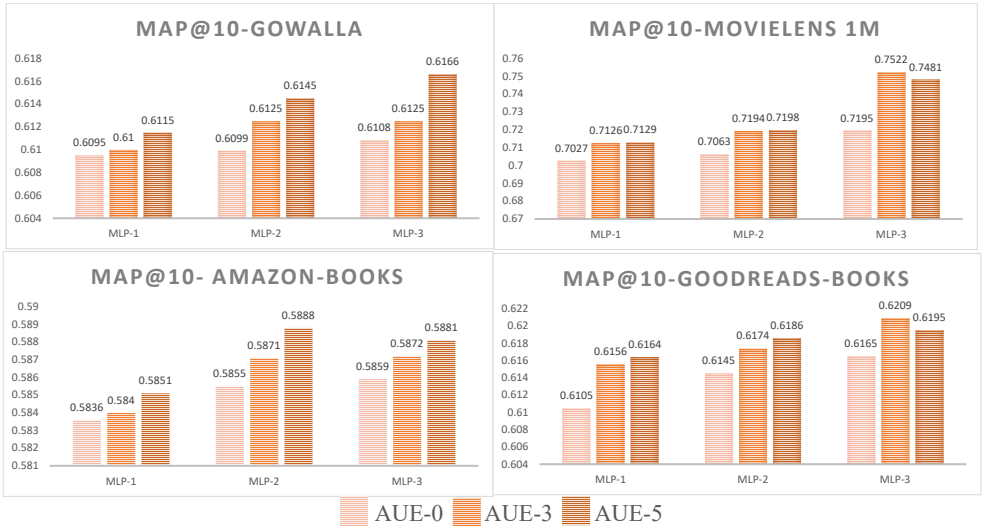

Figure (4.15) The variant of autoencoder hidden layers and MLP layers by MAP

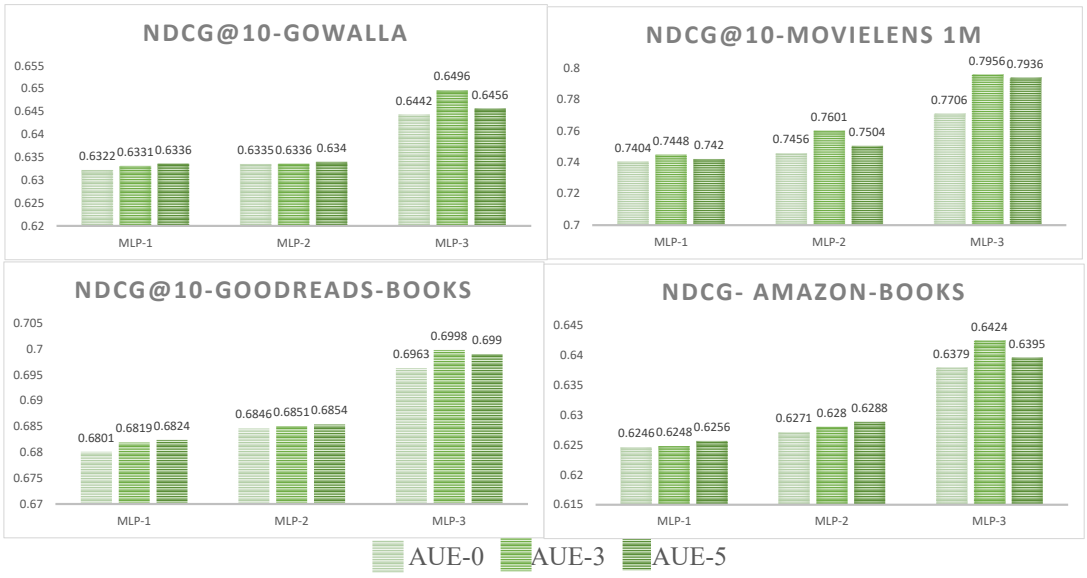

Figure (4.16) The variant of autoencoder hidden layers and MLP layers by NDCG

Influence of autoencoder hidden layers. The number of layers in autoencoder is the key hyperparameters. We further test the influence of autoencoder on the ProgrammableWeb dataset with the number of autoencoder hidden layers = $\{1, 2, 3, 4, 5\}$ with the variants of DHGA$^{(2z,k)}$ with $2z$ = $\{0, 2, 4, 6\}$ and k = $\{1, 2, 3, 4, 5\}$. Figure 4.13 shows the results of Accuracy, HR and NDCG in these different variants. In general, increasing the layer of autoencoder gives better performance. However, there is no improvement if the number of layers reaches to 4. It may be due to overfitting. These results demonstrate the effectiveness of using graph autoencoder and deep learning for collaborative filtering based recommender systems.

4.7 Conclusion and future work

In this paper, we investigated how the autoencoder architecture can be integrated into the collaborative filtering framework to facilitate injecting high-order connectivity signals into the embeddings for better mashup-API invocation prediction. We developed the MAG with high-order connectivity and k-neighbour sampling definitions, and presented the parameter-shared autoencoder structure for mashups and APIs embeddings. The main proposed is a novel data augmented autoencoder framework DHGA with two expansible sets of autoencoders, one for the mashuos and the other for the APIs. In addition, we also design a lower level model DHGA$^-$ which only consider the second-order and dot product tradition CF model, and use DHGA$^-$ as a baseline. Experimental results demonstrate that DHGA outperforms some state-of-the-art neural collaborative models including DHGA$^-$, MISR, NGCF and VAE-CF.

In the future, we plan to further investigate the relationship between the level of sparsity of the data and the high-order of connectivity of DHGA. We also plan to investigate how to apply DHGA in heterogeneous collaborative filtering where graph edges have different types.

Chapter 5

Motif-based Graph Attentional Neural Network for Web service recommendation

5.1 Introduction

As Web service is increasing nowadays, the demand of building suitable Web APIs (or APIs) requires the service recommendation system to execute more accuracy. In recent year, Deep Neural Network (DNN) based models have outperformed the other traditional Collaborative Filtering (CF) based method (Sarwar, Karypis, Konstan & Riedl, 2001b; H. Zhang et al., 2016; X. Chen et al., 2010; Hu et al., 2014; Xie, Wang et al., 2019; T. Liang et al., 2016; Xie, Chen et al., 2019). Such DNN-based service CF approaches only deal with the direct invocation between pairs of mashups and APIs and do not look at the high-order structures. For example, a recent DNN-based approach named Graph Convolutional Network (GCN) [20] uses a layer-wise propagation to operate the spectral convolution with the first order connectivity of nodes. In another work (X. Wang et al., 2020b), the mashup-API invocation pairs are treated as separate

data instances (so called *information isolated island*) and thus ignore the intrinsic structure among invocations. To address this limitation, recent DNN-based CF models such as (Y. Ma et al., 2020; Yao et al., 2021; G. Chen & Chen, 2015) exploit the high-order relations to solve the cold-start problem of service recommendation for new mashups. However, such approaches have not pay attention on the patterns of sub-graphs in the bipartite network.

In this chapter, we introduce a general approach of graph convolution networks which apply the attention motif-based adjacency matrices to secure the higher-order connectivity in graphs. The proposed model is named Motif-based Graph Attention Convolution (MGAT), which used an attention mechanism to attach the sub-graphs or motifs embedding features into the model. Specifically, it uses a Motif-based self-attention graph convolution network that contains intrinsic relation of the graph structure to learn the embedding of mashups and APIs.

This chapter has the following main contributions:

1. We proposed a Motif-based graph convolution self-attention method to attach the variant patterns of connectivity structure.
2. We proposed a Motif-based Graph Attention Collaborative Filtering for service recommendation (MGAT) that aggregates the motif-based neighbour into the learning progress for mashup-API recommendation.
3. We conducted extensive empirical studies on the Programmable dataset [1] and the results demonstrates the superior performance of MGAT over some state-of-the-art frameworks.

The remaining of this chapter is organized as follows. Section 5.2 presents the related work; Section 5.3 describes the sub-graphs in MAG, definitions of motifs proposes our data augmentation method, and presents the MGAT model; Section 5.4 demonstrates

[1] We downloaded the database from https://dev.maxmind.com/in November 2020.

the experiments running with comparison results and further analysis; and Section 5.5 concludes the chapter.

5.2 Related works

5.2.1 Network motifs and high-order Graph Neural Networks

The complex network' structures are presented by graphs as fundamental building blocks of networks, which are known as network motifs with high-order connectivity (Milo et al., 2002). Existing work (Prill, Iglesias & Levchenko, 2005) explores the motifs in biological networks demonstrating that the patterns of particular motifs correlating to the perturbation's robustness. Reversely, (Paranjape et al., 2017) considers motifs in temporal networks and states that each type motif presents different organization structures from different domains.

Existing work have studied the effectiveness of high-order connectivity with different graph-based machine learning models (N. Ahmed et al., 2020; Morris et al., 2019; Rossi, Ahmed & Koh, 2018; C. Yang, Liu, Zheng & Han, 2018). DeepGL (Rossi, Zhou & Ahmed, 2018) learns the inductive relational functions using motifs. (Rossi, Ahmed & Koh, 2018) studies the high-order network embeddings and approves that various motif-based matrix formulas obtain better embeddings.

Another work (C. Yang et al., 2018) proposes a hierarchical motif convolution for graph classification by identifying the task of sub-graphs. In addition, (C. Yang et al., 2018) designs a graph convolution framework for heterogeneous networks by leveraging the motif-based connectivity. Existing work (Morris et al., 2019) also demonstrates that GCN-based models and the one-dimension Weisfeiler-Lehman Isomorphism heuristic have similar above deficiency, so they propose a high-order framework for graph classification.

5.2.2 GNN and Motif-based Network in recommender systems

Existing work using Graph Neural Network model (Belkin, Niyogi & Sindhwani, 2006) to conform GNN to address the recommendation problem (Abu-El-Haija, Kapoor, Perozzi & Lee, 2020; Bahdanau, Cho & Bengio, 2014; Henaff, Bruna & LeCun, 2015). While GraphSAGE (Bahdanau et al., 2014) is an example of learning feature representations through sampling and aggregating strategies. PinSAGE (Henaff et al., 2015) is built for Web-scale recommender systems, which uses a random walk mechanism to learns node representations from chosen neighbor nodes and then integrating such high-order representations with GCN framework. (Abu-El-Haija et al., 2020) use the relationship of users and items to learn their representations. Instead of such random walk based methods, some work (Feichtenhofer, Pinz & Zisserman, 2016; Frasconi, Gori & Sperduti, 1998; X. Han, Liu & Sun, 2018) have used motif to capture the graph structural information. Particularly, a spectral motif convolution approach (Feichtenhofer et al., 2016) is built for convolution filters. Motif-CNN (X. Han et al., 2018) identifies several types of motifs to create the receptive fields for the target node, and then perform a motif-based spatial convolution operations to elicit the first interaction latent features. Another work on graph node classification (Frasconi et al., 1998) presents a motif-level self-attention model to learn the weight of different motifs using differentiation. Luo (Luo, Liu, Peng, Ying & Zhang, 2020) firstly present a Motif-based Neural Network applying for the Reciprocal Recommendation on Online Dating application. The work defines seven kinds of motifs and using a random walk algorithm to sample neighbour users to learn the their embedding features with skip-program mechanism. The interaction of a male and a female will be predicted by a fully connected neural network layers of a concatenation of their embedding feature vectors.

5.3 The MGAT model for bipartite network

In this section, we introduce some definition relating to motif, and present the proposed motif-based architecture and recommendation model. We first defines the Self-attention Motif-based Graph Convolution Layers which learn the embeddings by a Motif-based attention mechanism. After that, such layer is attached with a Neural network Collaborative Filtering based recommendation model.

5.3.1 Motif definitions

In general, Web API is a bipartite network MAG $G = (U, V, E)$ where U denotes the set of mashups, V denotes the set of APIs, and $E = U \times V$ with $e_{ij} \in E$ is an edge between U and V. If the API is invoked by the mashup then the edge e_{ij} exists and is assigned a value $r_{ij} \geq 0$, otherwise $r_{ij} = 0$.

Like social media, we assume that in Web Service network, connected mashups may have similar preferences and therefore have similar latent features. As a result, their embedding features would be close to each other. In the bipartite network, there are only interactions between two different nodes from different types of nodes, which are mashup and API in this thesis.

In the MAG, we defined a set of motifs $M_t = \{M_1, ..., M_T\}$, we build a set of T different motif-induced adjacency matrices $A = \{A_1, ..., A_T\}$ where A_t is defined as $(A_t)_{m,a}$, which is equal to the number of motifs of type M_t which contains both mashup m and a. Different motifs of size two to four in the Figure 5.1 have different neighbour sets. Hence, the weight or attention score of each motifs are varying in frequency between mashup/API pairs.

As mentioned in Figure 3.2, most of mashups invoke an average number of three APIs. Therefore, we believe that only a certain types of motifs can represent for the whole dataset. We consider six common motifs which are plotted in Figure 5.1. We

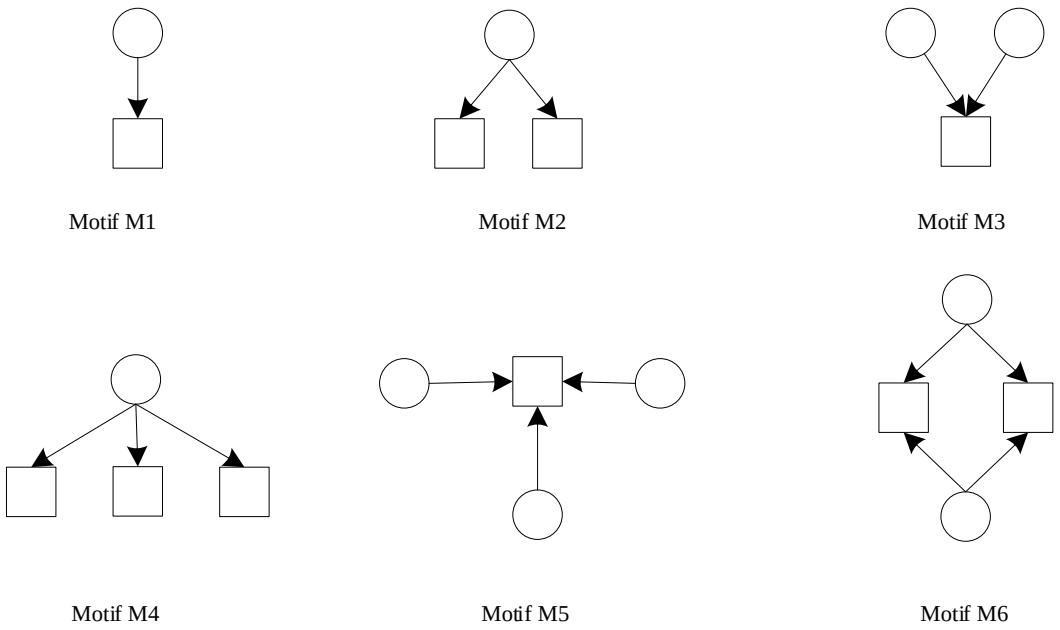

Figure (5.1) Types of motifs for Mashup-APi Graph

denote such motifs as $M_t = M1, M2, ..., M6$. Given the subgraph with one mashup and one API and their interaction, we obtain the first motif M1. Given the subgraph with three entities, we obtain the motifs M2 and M3. The second motif M2 presents the co-invocation of one mashup and two APIs. The motif M3 demonstrates two mashups use the same API. For the subgraph with four entities, we have three motifs M4, M5, and M6. M4 shows triples mashups use the same API. The motif M5 is the co-invocation of three APIs and one mashup while the last motif M6 shows two APIs invoked by both mashups.

5.3.2 Motif-based Graph Convolution Layers with Self-attention

Inspired by GAT, we implement the sole graph attentional layer for the whole MGAT model. We use the features of a set U of mashups and a set V of APIs as the input layer and denoted as $H_m = \{\vec{h_i}\}, i \in U$ and $H_a = \{\vec{h_j}\}, j \in V$, where $h_i, h_j \in \mathcal{R}_D$. From the input features' vectors, the graph attentional layer generates a new latent features for each mashup/API. We denote the set of output features' vectors as $H'_m = \{\vec{h'_i}\}, i \in U$

and $H'_a = \{\vec{h'_j}\}, j \in V$, where $h'_i, h'_j \in \mathcal{R}'_D$. D is the dimension of embedding features.

We update the mashups and APIs embedding features by a linear transformation weighted by shared parameter matrix $W \in \mathcal{R}^{D \times D'}$ applied for all mashups and APIs. Later, a self-attention framework $\vec{a}$ with shared attentional scores $\alpha_{m,t}$, $\alpha_{a,t}$ with $t \in T$ used for mashups and APIs respectively, which represents for the motif-based neighbor of every mashup and API. Different from GAT which explores the one-hop neighbors for the attention score, we use the motif-based neighbours instead. As a result, the number of attentional scores in MGAT depends on the number of motifs considered which are believed to contain intrinsic information of the MAG. Particularly, the motif-based attention values for mashup and API are formulated as below:

$$\alpha_{m,i} = \text{softmax}[(\textbf{LeakyReLU}(\vec{a}^T[W\vec{h_m} \parallel W\vec{h_i}])] \tag{5.1}$$

$$\alpha_{a,j} = \text{softmax}[(\textbf{LeakyReLU}(\vec{a}^T[W\vec{h_a} \parallel W\vec{h_j}])] \tag{5.2}$$

where T is the transpose function and $\parallel$ is the concatenation operation. After that, we use the normalized attention coefficients to combine the features of neighbour mashups/ APIs by a linear activation. The result embedding vectors are the output features for the corresponding nodes. Particularly, the embedding vector h'_m and h'_a are formally defined as below equations.

$$h'_m = \sum_{i \in \mathcal{N}_t(m)} \alpha_{m,i} W h_i \tag{5.3}$$

$$h'_a = \sum_{j \in \mathcal{N}_t(a)} \alpha_{a,j} W h_j \tag{5.4}$$

where the motif-based neighbor lists of the mashup m and API a are denoted as $\mathcal{N}_t(m)$

and $\mathcal{N}_t(a)$ respectively with $t \in M_t$. In the next step, different from GAT, which applies the multi-head attention by using K independence attention mechanisms and concatenate to execute the output representation, we obtain the embedding vector $\vec{h_m}$, $\vec{h_a}$ by adding the attention scores of their motif-based neighbours. The calculation of such vectors are formulated in Equation 5.6 and Equation 5.5.

$$h'_m = \|_{t \in M_t} \sigma\Big(\sum_{i \in \mathcal{N}_t(m)} \alpha^t_{m,i} W^t h_i\Big) \tag{5.5}$$

$$h'_a = \|_{t \in M_t} \sigma\Big(\sum_{j \in \mathcal{N}_t(a)} \alpha^t_{a,j} W^t h_j\Big) \tag{5.6}$$

where $\alpha^t_{m,i}$ are the normalized attentional scores calculated from attention mechanism of the mashup m and its neighbors with motif type M_t and likewise for $\alpha^t_{a,j}.W^t h_i), W^t h_j)$ are the weight parameter matrices for the linear transformation according to the set of mashups and APIs. The combination process of such graph self-attention layer is demonstrated in Figure 5.2. This process is used for both mashup and API. For instance, the Figure 5.2 shows the API h_a and its motif-based neighbors h_1, h_2, h_3, h_4. Clearly, h_a and h_1 are in the same motif M_1 and motif M_2. This motif-based relation is the same to the pairs (h_a, h_4) while the pair (h_a, h_3) is only belong to motif M_3. The pair (h_a, h_2) has the strongest relation with three types of motifs M_1, M_2, and M_3. All the motif-based attention scores are calculated by the attention mechanism and concatenated to generate the output h'_a. As a result, the output sets of features H'_m and H'_a are obtained from the motif-based self-attention layers and contain the high-order graph relation, which can be used as auxiliary information for predictive model in the next subsection.

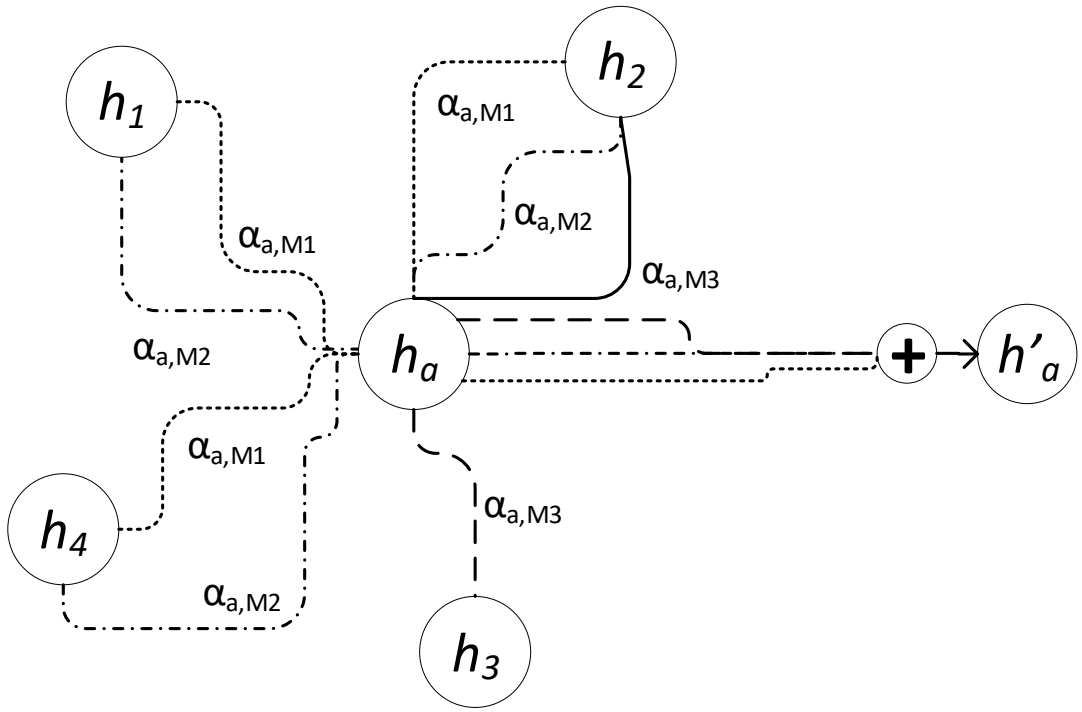

Figure (5.2) Motif-based attention mechanism

5.3.3 Motif-based Graph Attention Collaborative Filtering for service recommendation (MGAT)

In this section, We aim to project the invocation between the pair of mashup m and API a provided their latent representation h'_m and h'_a. The prediction value R will be obtained by the dot product of h'_m and h'_a and learning through an MLP layers.

$$R = h'_m \odot h'_a \tag{5.7}$$

The value of $R \in \{0, 1\}$, in which 1 indicates the existing invocation between the mashup and API and 0 otherwise. We use the cross-entropy loss function to optimize the model.

$$Loss = -\sum_{m \in U, a \in V} r_{ma} \log \hat{r}_{ma} + (1 - r_{ma}) \log (1 - \hat{r}_{ma}) \tag{5.8}$$

5.4 Experimental results

Our model can be applied to any bipartite network and we use the ProgrammableWeb dataset to evaluate our proposed model. Our objective is to answer the following research questions:

- RQ1: How much does MGAT outperform the state-of-the-art graph-based CF methods?
- RQ2: How do different types of motifs influence the performance of the proposed model?

5.4.1 Datasets and baselines

Datasets In Web API application, we use the dataset of ProgrammableWeb which can be presented as a bipartite network to evaluate the success of the proposed model. The dataset consists of 17829 APIs and 6340 mash-ups, and their invocation data. Specifically, if mashup m_1 invokes API a_5, their invocation data is 1, and we use this value is denoted as the invocation of such mashup-API pair. Then, we process the following steps for data reprocessing: (a) removing all blank APIs and mashups to shortly obtain 5691 mashups and 1170 APIs; (b) removing a particular portion of the mashup-API invocations to generate a train set, and use the original data as the test set.

Baselines We compare our proposed MGAT with some state-of-art approaches:

- AMF: an attentional Matrix factorization with document context and API co-invocation.
- NGCF: a GCN-based collaborative filtering method that propagates embeddings on the MAG to exploit the high-order connectivity signal of mashups and users.

- HACF: High-order Data Augmentation Collaborative Filtering method for service recommendation.
- GAT-CF: a graph attention network based collaborative filtering model.
- MISR: a multiplex invocation-oriented service recommendation model which leverages three types of invocations between services and mashups and incorporates them into a DNN to present for their explicit and implicit relationships.

5.4.2 Settings

Evaluation metrics We use hit ratio (HR) and Normalized Discounted Cumulative Gain (NDCG) to evaluate the performance of API recommendation. We pick unused APIs for each mashup in the train dataset. After that, we sort them ascending to their estimated ratings and truncate the top-p ranked for the recommendation list. The HR is calculated from the top-p recommendation list by using the equation:

$$\text{HR@p} = \frac{\text{Number of hits } @p}{N_r} \tag{5.9}$$

Where we denoted N_r as the number of recommended APIs. We then measure the effectiveness of recommendation by the Discounted Cumulative Gain (NDCG) by using the ranking of relevant APIs in the prediction list. Specifically, the NDCG from rank 1 to p for the recommendation list is obtained by the below equation:

$$\text{NDCG@p} = \sum_{i=1}^{p} \frac{2^{r_i} - 1}{\log_2(i+1)} \tag{5.10}$$

Hyper-parameters All datasets are split into train set and test set with ratio 80% and 20% respectively. We use the train set to train the proposed MGAT model and evaluate the HR and NDCG on a test set. For the evaluation process, all the mashups in the test set have a list of invoked APIs. For each mashup, all APIs in the test set

are scored with their estimated invocation value which is between 0 and 1. After that, we sort them ascending to their estimated ratings. According to recent methods, we truncate the top-p ranked for the recommendation list, Then, we get the top p highest score list which will be compared with the test set to see if the APIs are in the test or not. Consequently, the number of APIs are in both top p list and test set is the number of hits. For evaluation metrics, we use p=10 for our experiment.

We implement MGAT based on Pytorch using Graph-tools to develop the model. In order to ensure MGAT achieves the best performance, we use grid search to tune the hyperparameters. As a result, the suitable values of hyperparameters obtained are learning rate, convergence epoch, the hidden size , drop out which are set at 0.005, 1000, 8, 0.6 respectively.

5.4.3 Comparison results

For all the baselines and MGAT variants, we run the models until the values of loss converge and obtain the best results of HR and NDCG. We calculate the HR and NDCG at k from 2 to 10. Table 5.1 and Table 5.2 show the performance details of the baselines and MGAT123 which version obtains the best result among MGAT variants (see Subsection 5.4.4). Overall, the results demonstrate our proposed model MGAT123 completely obtains superior numbers against exising work including AMF, NGCF, and MISR which do not consider the high-order connectivity of the MAG. For later approaches such as HACF and GAT-CF which attach mashup-API high-order relation, our MGAT does not completely outperform at all but generally it achieves the best HR and NDCG in most of larger value of k, particularly with k from 5 to 10. The Figure 5.3 and Figure 5.4 provide better visualization for such comparison.

One interesting observation in these both figures is that there are two different groups: the low performance group consists of AMF and NGCF and the high performance group

Table (5.1) Baseline comparison over different p on HR metric

p	2	3	4	5	6	7	8	9	10
AMF	0.4021	0.4152	0.4182	0.4262	0.4352	0.4521	0.4921	0.5201	0.5583
NGCF	0.4372	0.4552	0.5037	0.5537	0.6012	0.6252	0.6317	0.6677	0.6882
MISR	0.6494	0.6760	0.7023	0.7211	0.7286	0.7392	0.7392	0.7407	0.7441
HACF	0.6645	0.7190	0.7367	0.7483	0.7547	0.7603	0.7604	0.7606	0.7616
GAT-CF	**0.5261**	0.6176	0.6716	**0.7271**	**0.7647**	**0.768**	**0.7761**	0.7761	0.7761
MGAT123	**0.5261**	**0.6275**	**0.6748**	**0.7271**	**0.7647**	**0.768**	0.7745	**0.7908**	**0.7908**

Table (5.2) Baseline comparison over different p on NDCG metric

p	2	3	4	5	6	7	8	9	10
AMF	0.3321	0.3498	0.3694	0.3972	0.3999	0.4093	0.4109	0.4271	0.4306
NGCF	0.3521	0.3841	0.4393	0.4647	0.4678	0.4580	0.4603	0.4613	0.4647
MISR	0.5209	0.5050	0.5343	0.5301	0.5351	0.5433	0.5569	0.5568	0.5583
HACF	**0.5626**	**0.5666**	**0.5627**	0.5657	0.5711	0.5716	0.5718	0.5725	0.5726
GAT-CF	0.4857	0.529	0.5541	0.5714	0.5843	0.5833	0.5857	0.5834	0.5809
MGAT123	0.4857	0.5344	0.5559	**0.5724**	**0.5852**	**0.5843**	**0.5861**	**0.5906**	**0.5889**

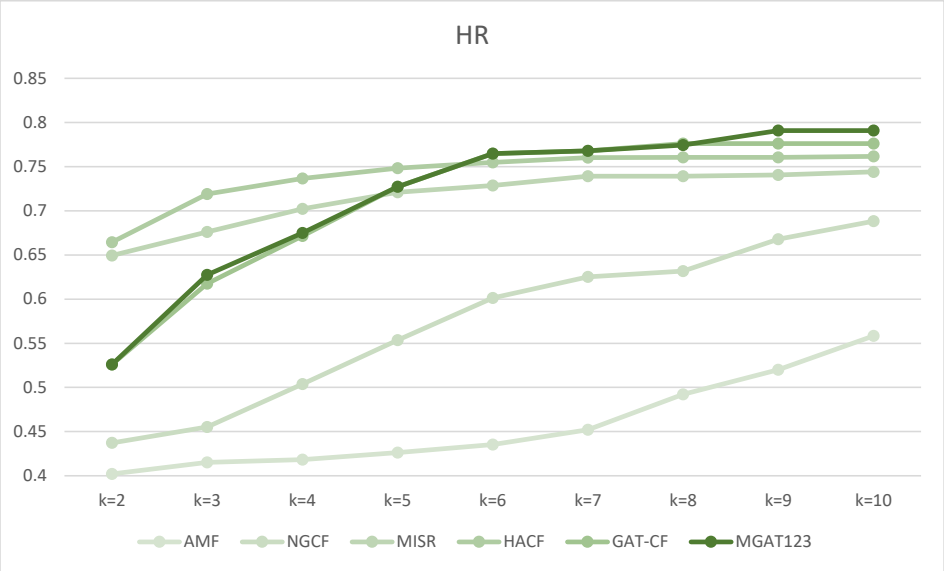

Figure (5.3) HR results of baselines and MGAT123.

are the remaining models which all consider the high-order connectivity of mashup-API relation. It is very clear that the performance gaps between groups are significantly large, which demonstrates the benefit of attaching high-order information into to predictive models. In the high performance group, MGAT123 achieves the best numbers, which approves the significant role of motifs to enhance the accuracy of recommender systems.

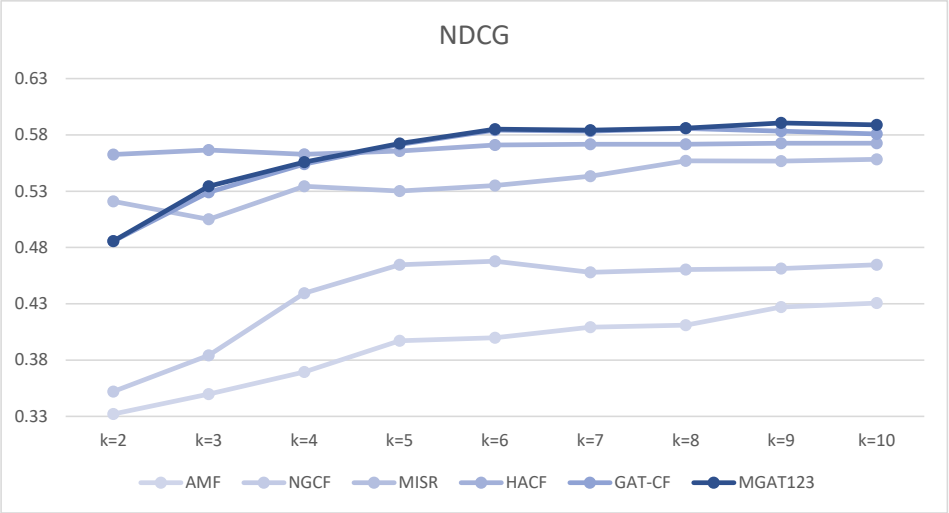

Figure (5.4) NDCG results of baselines and MGAT123.

Table (5.3) HR results of variants of MGAT

Variant	2	3	4	5	6	7	8	9	10
GAT-CF	0.5261	0.6176	0.6716	0.7271	0.7647	0.768	0.7761	0.7761	0.7761
MGAT12	0.5261	0.6291	0.6716	0.7271	0.7647	0.768	0.7729	0.7859	0.7892
MGAT13	0.5261	0.6275	0.6748	0.7271	0.7647	0.768	0.7761	0.7761	0.7761
MGAT14	0.5261	0.6307	0.6716	0.7255	0.7647	0.768	0.7729	0.7892	0.7908
MGAT15	0.5261	0.6275	0.6748	0.7271	0.7647	0.768	0.7761	0.7761	0.7761
MGAT16	0.5261	0.6291	0.6748	0.7271	0.7647	0.768	0.7761	0.7794	0.781
MGAT1234	0.5261	0.6291	0.6716	0.7271	0.7647	0.768	0.7761	0.7892	0.7908
MGAT123456	0.5261	0.6209	0.6748	0.7271	0.7647	0.768	0.7761	0.7761	0.7761
MGAT123	0.5261	0.6275	0.6748	0.7271	0.7647	0.768	0.7745	0.7908	0.7908
MGAT145	0.5261	0.6258	0.6748	0.7288	0.7647	0.768	0.7761	0.7761	0.7761

5.4.4 The influence of different types of motifs on the MGAT's performance

Bringing the motif awareness to the MAG has demonstrated the improvement of recommendation results. In this subsection, we study how each type of motif contribute to the success of this innovation. Table 5.3 and Table 5.4 show the detail of HR and NDCG scores when applying different types of motifs on the MGAT framework. Generally, the numbers are quite similar at small value of k, they are more distinct when $k > 5$.

Table (5.4) NDCG results of variants of MGAT

Variant	2	3	4	5	6	7	8	9	10
GAT-CF	0.4857	0.529	0.5541	0.5714	0.5843	0.5833	0.5857	0.5834	0.5809
MGAT12	0.4857	0.5352	0.5548	0.5722	0.5851	0.5841	0.5855	0.5889	0.5881
MGAT13	0.4857	0.5344	0.5561	0.5722	0.5851	0.5841	0.5865	0.5844	0.5821
MGAT14	0.4857	0.536	0.555	0.5718	0.5852	0.5843	0.5856	0.5901	0.5882
MGAT15	0.4857	0.5344	0.5561	0.5722	0.5851	0.5841	0.5865	0.5844	0.5821
MGAT16	0.4857	0.5352	0.556	0.5725	0.5853	0.5844	0.5868	0.5871	0.5852
MGAT1234	0.4857	0.5352	0.5546	0.5724	0.5852	0.5842	0.5866	0.5901	0.5885
MGAT123456	0.4857	0.5307	0.5555	0.5717	0.5846	0.5836	0.586	0.5839	0.5817
MGAT123	0.4857	0.5344	0.5559	0.5724	0.5852	0.5843	0.5861	0.5906	0.5889
MGAT145	0.4857	0.5335	0.5557	0.5727	0.5851	0.5841	0.5865	0.5843	0.5818

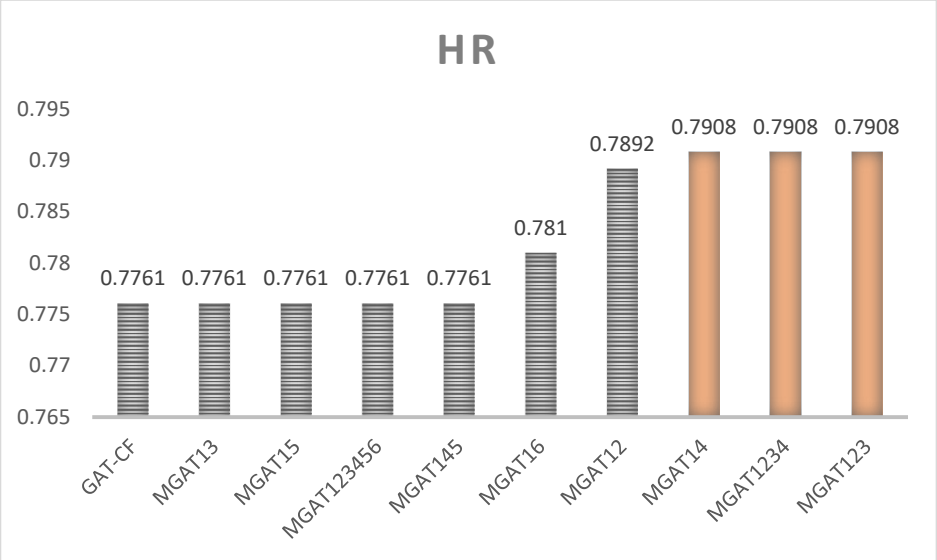

Figure (5.5) HR results of MGAT variants

5.5 Conclusion and future work

The chapter studies the subgraph or types of motifs in the mashup-API bipartite and the influence of graph structure on the embedding feature of services. The proposed MGAT uses the motif-based architecture and attaches it with the CF-based model through graph neural convolution layers. The model also gives different attentional scores to types of motifs, which are simultaneously learned during the model training. The results of experiment shows the exceptional success of MGAT compared with other models.

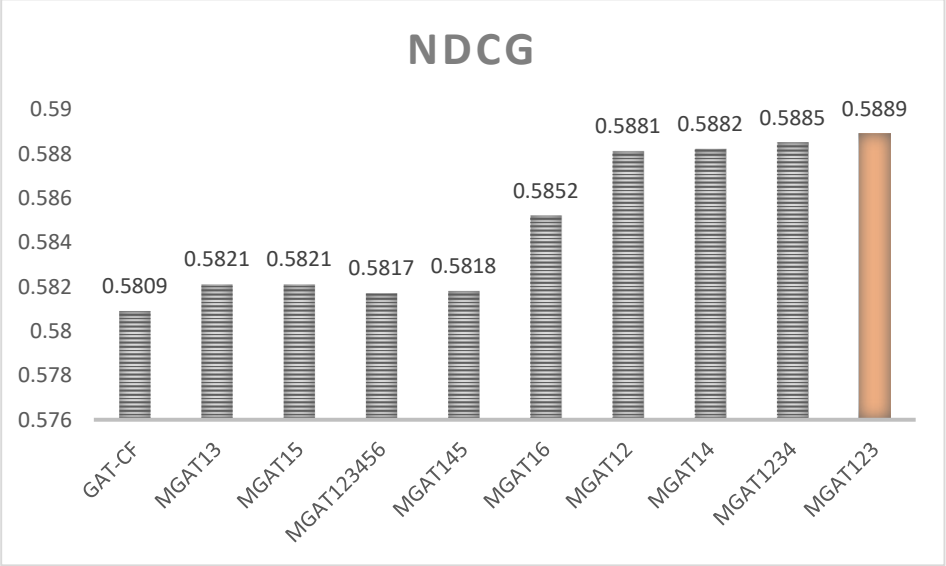

Figure (5.6) NDCG results of MGAT variants.

Chapter 6

Conclusion

6.1 Introduction

Web service has been increased quickly during last decade with a great number of APIs published on the internet. Such APIs have been commonly applied to variants of Web and mobile applications. This blooming growth brought difficulties to select proper APIs for building the mashups. A great number of existing work have exploited variant elements in mashup-API invocation data to predict the most suitable APIs including direct invocation data and other auxiliary information such as context description, location, tags, etc. However, the intrinsic relation in a graph structure has not been consider in service recommendation. This thesis mainly focuses on the challenges associated with service recommendation using auxiliary information and graph structure with high-order neighbour features. It proposes the use of embedding techniques to learn the latent features of mashups and APIs through autoencoder frameworks, which effectively discover and attach the data intrinsic relation into the prediction models. Particularly, the proposed models exploit the graph structural information through a high-order connectivity autoencoders with multiple neural layers and a motif-based convolution neural network. Such deep neural network models adequately investigate

the latent features of the whole bipartite network.

Generally, the thesis has some contributions as followings. First, Chapter 3 of the thesis proposes a model for leveraging the document context and co-invocation historical data to regularize the attentional matrix factorization model. Second, Chapter 4 of the thesis proposed a data augmented high-order graph autoencoder to further explore the intrinsic relation of the mashup-API bipartite network. Third, Chapter 5 of the thesis proposes the motif-based graph attentional neural network for web service recommendation. Chapter 6 concludes the thesis by summing up the contributions of the thesis, the main experimental results and figure out some directions for future work. Section 6.1 presents more details on the contributions of this thesis. Section 6.2 presents the disadvantages and probable improvements of proposed frameworks. Section 6.3 summarizes an overview the directions for future work.

6.1.1 Research contributions

The thesis aim attention at the challenges related to existing Web service CF-based recommender systems. Such challenges consist of (1) the primary dependence on a very sparsity of the mashup-API invocation data and the rich information from service document context and co-invocation historical data had not been effectively utilized in existing recommender systems; (2) only using the direct mashup-API invocations limits the recommender system to explore higher intrinsic relationship in a bipartite graph; (3) the subgraphs or motif patterns of graph which potentially contain rich information of high-order connectivity among mashups and APIs, which are not been explored and leveraged. The three mentioned challenges are addressed by the research questions defined as following: (1) How to enhance the prediction accuracy for service recommendation models which reply only in a very sparse dataset?. This question is address by applying an attentional mechanism matrix factorization framework regularized by

document context and mashup-API co-invocation; (2) How to explore the intrinsic relation among mashups and APIs in a bipartite graph? This question is solved by using data augmented technique with high-order graph autoencoder frameworks to learn the embedding features of services; (3) Will the motif awareness bring more benefit to service recommender system? The question is addressed by applying a motif-based convolution network within a CF-based framework which integrated the most frequent used motifs in the mashup-API graph. The detail contributions of the thesis are presented as following.

Matrix Factorization based CF model with Attention mechanism, document context, and API co-invocation.

Leveraging the auxiliary information in CF-based models have been successfully enhance the prediction accuracy for service recommender systems. The lack of distinguishing the different weights of invocations limits the performance of such models. In reality, not all the latent features are important and some might be noise to the model. In addition, the latent features are mostly learnt from direct invocation, which is very sparse. Different works focus on variant types of auxiliary information beyond the invocation such as location, description, etc. However, there is no research on the co-invocation and document context of the mashups and APIs to apply for PMF-based models.

Chapter 3 deals with above challenges and proposes an attentional matrix factorization CF model which is regularizing by the document context and mashup-API co-invocation. The model uses common techniques in Natural Language Processing like Doc2Vec to learn the document embeddings for each description of mashup/API. The chapter presents a comprehensive statistical analysis for the frequency of invocations to draw out the APIs co-invocation pattern, which contain rich information of the compatibility of the services. Inheriting the success of attention mechanism, the

proposed AMF model applies this framework integrated with document context and co-invocation, which together make a strong service recommender system to produce a very accuracy prediction compared with current baselines.

Augmenting data and high order connectivity graph embedding

The DNN has played a successful role in many existing matrix factorization models to enhance the prediction accuracy for recommendation. However, most of such models only focus on the direct interaction or the first order connectivity among mashups and APIs. So the intrinsic relation which has potential contribution is not utilized. Particularly, all the unobserved are treated as negative instances. Because such negative data is very large compared with observed points so many useful information from it are ignored.

Chapter 4 leverages the high-order connectivity and explores this element with the autoencoder framework, which was successfully in learning the hidden representation to draw out the information of data structure. The chapter proposes a data augmentation technique to secure the first and high-order connectivity, which augments the sparse dataset with more useful information under latent features. Such features are attached to the services' embedding for better prediction. The chapter presents the experimental results on many datasets including ProgrammableWeb, Movielens, Gowalla, Goodreads-books, and Amazon-books, which demonstrate the superior performance of DHGA over other baselines.

Motif-based Graph Attentional Neural network

The use of high-order connectivity in DNN model has improved a lot the accuracy for service recommendation as in Chapter 4. However, the improvement is never enough for a better recommender system. Inspired by the success of subgraph or motif-based techniques, the Chapter 5 explores further on the graph structure of the mashup-API

bipartite to dig out more potential information of mashup-API relationship. Like AMF and DHGA, the chapter uses the attention mechanism and presents a Motif-based Graph Attention Convolution model. The chapter also defines some common used motifs and run experiment on different types of motifs to figure out their different influences on the recommendation results.

6.1.2 Limitations and future direction

Even though the proposed models effectively solve the research questions defined in this thesis, there are still some limitations that might occur. This section lists such limitations and some of the potential improvement for future work.

First, in spite of the superior performance the AMF model obtains, such model still has some limitations. The incorporation of document context and co-invocation which contain lot of rich information are only applied as the regularization parts in a shallow PMF-based model while the main model still mostly relies on the direct invocation with attentional scores. Building a deep model with such valuable information should be a potential work to enhance the recommendation precision.

Second, the DHGA model is a deep model and has superior performance compared with AMF. DHGA shows its success on exploiting and learning the intrinsic relation of the mashup-API graph through the autoencoder frameworks. However, it is restricted in a traditional neural network with a simple combination of separated autoencoder parts. Such model has not utilized the graph structure in its main model. A graph convolution neural network attached to DHGA could bring more useful graph structural information for the model. In addition, DHGA assumes that first and high-order connectivity bring more effective information than false positive instances. However, in the sparsity network, there are more invocations among highly relevant services than among irrelevant ones. Future research can focus on both assortative and disassortative

services.

Finally, the MGAT is the most superior model among our proposed models in this thesis. MGAT uses the latest motif-based model with graph neural convolution layers in the CF-based framework. Such model considers only some common used motifs in the graph and rely only on the invocation data. Future work can explore whether more complex types of motifs and additional information can bring more achievements for such model. Also, a random walk technique can be used to explore higher-order connectivity in the graph convolution layers.

References

Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M. & Steggles, P. (1999). Towards a better understanding of context and context-awareness. In *International symposium on handheld and ubiquitous computing* (pp. 304–307).

Abu-El-Haija, S., Kapoor, A., Perozzi, B. & Lee, J. (2020). N-gcn: Multi-scale graph convolution for semi-supervised node classification. In *uncertainty in artificial intelligence* (pp. 841–851).

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 and data engineering*, *17*(6), 734–749.

Afify, Y. M., Moawad, I. F., Badr, N. L. & Tolba, M. F. (2014). Cloud services discovery and selection: survey and new semantic-based system. In *Bio-inspiring cyber security and cloud services: Trends and innovations* (pp. 449–477). Springer.

Ahmed, A., Shervashidze, N., Narayanamurthy, S., Josifovski, V. & Smola, A. J. (2013). Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on world wide web* (pp. 37–48).

Ahmed, N., Rossi, R. A., Lee, J., Willke, T., Zhou, R., Kong, X. & Eldardiry, H. (2020). Role-based graph embeddings. *IEEE Transactions on Knowledge and Data Engineering*.

Akkiraju, R., Farrell, J., Miller, J. A., Nagarajan, M., Sheth, A. P. & Verma, K. (2005). Web service semantics-wsdl-s.

Al-Masri, E. & Mahmoud, Q. H. (2008). Investigating web services on the world wide web. In *Proceedings of the 17th international conference on world wide web* (pp. 795–804).

Angles, R. & Gutierrez, C. (2008). Survey of graph database models. *ACM Computing Surveys (CSUR)*, *40*(1), 1–39.

Ankolekar, A., Burstein, M., Hobbs, J. R., Lassila, O., Martin, D., McDermott, D., ... others (2002). Daml-s: Web service description for the semantic web. In *International semantic web conference* (pp. 348–363).

Atwood, J. & Towsley, D. (2016). Diffusion-convolutional neural networks. In *Advances in neural information processing systems* (pp. 1993–2001).

Backstrom, L. & Leskovec, J. (2011). Supervised random walks: predicting and recommending links in social networks. In *Proceedings of the fourth acm international conference on web search and data mining* (pp. 635–644).

Bahdanau, D., Cho, K. & Bengio, Y. (2014). Neural machine translation by jointly

learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Belkin, M., Niyogi, P. & Sindhwani, V. (2006). Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, *7*(11).

Bhagat, S., Cormode, G. & Muthukrishnan, S. (2011). Node classification in social networks. In *Social network data analytics* (pp. 115–148). Springer.

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

Bo, D., Wang, X., Shi, C. & Shen, H. (2021). Beyond low-frequency information in graph convolutional networks. *arXiv preprint arXiv:2101.00797*.

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

Botangen, K. A., Yu, J., Sheng, Q., Han, Y. & Yongchareon, S. (2020). Geographic-aware collaborative filtering for web service recommendation. *Expert Syst. Appl.*, *151*, 113347.

Bouguettaya, A., Sheng, Q. Z. & Daniel, F. (2014). *Web services foundations*. Springer.

Bouguettaya, A., Singh, M., Huhns, M., Sheng, Q. Z., Dong, H., Yu, Q., ... others (2017). A service computing manifesto: the next 10 years. *Communications of the ACM*, *60*(4), 64–72.

Broens, T., Pokraev, S., Van Sinderen, M., Koolwaaij, J. & Costa, P. D. (2004). Context-aware, ontology-based service discovery. In *European symposium on ambient intelligence* (pp. 72–83).

Bruna, J., Zaremba, W., Szlam, A. & LeCun, Y. (2013). Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*.

Buqing, C., Tang, M. & Huang, X. (2014). Cscf: A mashup service recommendation approach based on content similarity and collaborative filtering. *International Journal of Grid & Distributed Computing*, *7*(2).

Cao, B., Liu, J., Tang, M., Zheng, Z. & Wang, G. (2013). Mashup service recommendation based on user interest and social network. In *2013 ieee 20th international conference on web services* (pp. 99–106).

Cao, S., Lu, W. & Xu, Q. (2015). Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th acm international on conference on information and knowledge management* (pp. 891–900).

Cao, S., Lu, W. & Xu, Q. (2016). Deep neural networks for learning graph representations. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 30).

Casati, F. & Shan, M.-C. (2001). Dynamic and adaptive composition of e-services. *Information systems*, *26*(3), 143–163.

Chamberlain, B. P., Clough, J. & Deisenroth, M. P. (2017). Neural embeddings of graphs in hyperbolic space. *arXiv preprint arXiv:1705.10359*.

Channabasavaiah, K., Holley, K. & Tuggle, E. (2003). Migrating to a service-oriented architecture. *IBM DeveloperWorks*, *16*, 727–728.

Chen, G. & Chen, L. (2015, 08). Augmenting service recommender systems by

incorporating contextual opinions from user reviews. *User Modeling and User-Adapted Interaction*, *25*. doi: 10.1007/s11257-015-9157-3

Chen, L., Wu, L., Hong, R., Zhang, K. & Wang, M. (2020). Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach. *arXiv preprint arXiv:2001.10167*.

Chen, L., Zheng, A., Feng, Y., Xie, F. & Zheng, Z. (2018). Software service recommendation base on collaborative filtering neural network model. In *International conference on service-oriented computing* (pp. 388–403).

Chen, W., Paik, I. & Hung, P. C. (2013). 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).

Chung, F. R. & Graham, F. C. (1997). *Spectral graph theory* (No. 92). American Mathematical Soc.

De Bruijn, J., Lausen, H., Polleres, A. & Fensel, D. (2006). The web service modeling language wsml: An overview. In *European semantic web conference* (pp. 590–604).

De Oliveira, M. F. & Levkowitz, H. (2003). From visual data exploration to visual data mining: A survey. *IEEE transactions on visualization and computer graphics*, *9*(3), 378–394.

Donnat, C., Zitnik, M., Hallac, D. & Leskovec, J. (2018). Learning structural node embeddings via diffusion wavelets. In *Proceedings of the 24th acm sigkdd international conference on knowledge discovery & data mining* (pp. 1320–1329).

Dormann, C. F., Gruber, B. & Fründ, J. (2008). Introducing the bipartite package: analysing ecological networks. *interaction*, *1*(0.2413793).

Duan, Y., Fu, G., Zhou, N., Sun, X., Narendra, N. C. & Hu, B. (2015). Everything as a service (xaas) on the cloud: origins, current and future trends. In *2015 ieee 8th international conference on cloud computing* (pp. 621–628).

Dustdar, S. & Schreiner, W. (2005). A survey on web services composition. *International journal of web and grid services*, *1*(1), 1–30.

Feichtenhofer, C., Pinz, A. & Zisserman, A. (2016). Convolutional two-stream network fusion for video action recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 1933–1941).

Finkelstein, A. & Savigni, A. (2001). A framework for requirements engineering for context-aware services..

Fortunato, S. (2010). Community detection in graphs. *Physics reports*, *486*(3-5), 75–174.

Frasconi, P., Gori, M. & Sperduti, A. (1998). A general framework for adaptive processing of data structures. *IEEE transactions on Neural Networks*, *9*(5), 768–786.

Gao, L. & Li, C. (2008). Hybrid personalized recommended model based on genetic

algorithm. In *2008 4th international conference on wireless communications, networking and mobile computing* (pp. 1–4).

Gao, W., Chen, L., Wu, J. & Bouguettaya, A. (2016). Joint modeling users, services, mashups, and topics for service recommendation. In *2016 ieee international conference on web services (icws)* (pp. 260–267).

Gao, Y., Li, Y.-F., Lin, Y., Gao, H. & Khan, L. (2020). Deep learning on knowledge graph for recommender system: A survey. *arXiv preprint arXiv:2004.00387*.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O. & Dahl, G. E. (2017). Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*.

Goodfellow, I., Bengio, Y. & Courville, A. (2016). Machine learning basics. *Deep learning*, *1*, 98–164.

Goyal, P. & Ferrara, E. (2018). Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, *151*, 78–94.

Gronmo, R., Skogan, D., Solheim, I. & Oldevik, J. (2004). Model-driven web service development. *International Journal of web Services Research (IJWSR)*, *1*(4), 1–13.

Grover, A. & Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 855–864).

Guinard, D., Trifa, V. & Wilde, E. (2010). A resource oriented architecture for the web of things. In *2010 internet of things (iot)* (pp. 1–8).

Guo, H., Tang, R., Ye, Y., Li, Z. & He, X. (2017). Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*.

Guo, L., Wang, S., Kang, L. & Cao, Y. (2015). Agent-based manufacturing service discovery method for cloud manufacturing. *The International Journal of Advanced Manufacturing Technology*, *81*(9), 2167–2181.

Guo, Q., Zhuang, F., Qin, C., Zhu, H., Xie, X., Xiong, H. & He, Q. (2020). A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*.

Hamilton, W. L., Ying, R. & Leskovec, J. (2017a). Inductive representation learning on large graphs. In *Proceedings of the 31st international conference on neural information processing systems* (pp. 1025–1035).

Hamilton, W. L., Ying, R. & Leskovec, J. (2017b). Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*.

Han, S. N. & Crespi, N. (2017). Semantic service provisioning for smart objects: Integrating iot applications into the web. *Future Generation Computer Systems*, *76*, 180–197.

Han, X., Liu, Z. & Sun, M. (2018). Neural knowledge acquisition via mutual attention between knowledge graph and text. In *Thirty-second aaai conference on artificial intelligence.*

He, P., Zhu, J., Zheng, Z., Xu, J. & Lyu, M. R. (2014). Location-based hierarchical matrix factorization for web service recommendation. In *2014 ieee international conference on web services* (pp. 297–304).

He, Q., Yan, J., Jin, H. & Yang, Y. (2014). Quality-aware service selection for service-based systems based on iterative multi-attribute combinatorial auction. *IEEE Transactions on Software Engineering*, *40*(2), 192–215.

He, R. & McAuley, J. (2016). Vbpr: visual bayesian personalized ranking from implicit feedback. In *Thirtieth aaai conference on artificial intelligence.*

He, X., Liao, L., Zhang, H., Nie, L., Hu, X. & Chua, T.-S. (2017a). Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web* (pp. 173–182).

He, X., Liao, L., Zhang, H., Nie, L., Hu, X. & Chua, T.-S. (2017b). Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web* (pp. 173–182).

He, X., Liao, L., Zhang, H., Nie, L., Hu, X. & Chua, T.-S. (2017c). Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web* (pp. 173–182).

He, X., Zhang, H., Kan, M.-Y. & Chua, T.-S. (2016). Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th international acm sigir conference on research and development in information retrieval* (p. 549–558). New York, NY, USA: Association for Computing Machinery. Retrieved from `https://doi.org/10.1145/2911451.2911489` doi: 10.1145/2911451.2911489

Henaff, M., Bruna, J. & LeCun, Y. (2015). Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*.

Henderson, K., Gallagher, B., Eliassi-Rad, T., Tong, H., Basu, S., Akoglu, L., ... Li, L. (2012). Rolx: structural role extraction & mining in large graphs. In *Proceedings of the 18th acm sigkdd international conference on knowledge discovery and data mining* (pp. 1231–1239).

Henricksen, K. (2003). A framework for context-aware pervasive computing applications.

Henricksen, K. & Indulska, J. (2004). A software engineering framework for context-aware pervasive computing. In *Second ieee annual conference on pervasive computing and communications, 2004. proceedings of the* (pp. 77–86).

Herlocker, J. L., Konstan, J. A., Borchers, A. & Riedl, J. (2017). An algorithmic framework for performing collaborative filtering. In *Acm sigir forum* (Vol. 51, pp. 227–234).

Hinton, G. E. & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, *313*(5786), 504–507.

Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Hoff, P. D., Raftery, A. E. & Handcock, M. S. (2002). Latent space approaches to social network analysis. *Journal of the american Statistical association*, *97*(460), 1090–1098.

Hu, Y., Peng, Q., Hu, X. & Yang, R. (2014). Time aware and data sparsity tolerant web service recommendation based on improved collaborative filtering. *IEEE Transactions on Services Computing*, *8*(5), 782–794.

Jain, A., Liu, X. & Yu, Q. (2015). Aggregating functionality, use history, and popularity of apis to recommend mashup creation. In *International conference on service-oriented computing* (pp. 188–202).

Kearnes, S., McCloskey, K., Berndl, M., Pande, V. & Riley, P. (2016). Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, *30*(8), 595–608.

Kim, D., Park, C., Oh, J. & Yu, H. (2017). Deep hybrid recommender systems via exploiting document context and statistics of items. *Information Sciences*, *417*, 72–87.

Kingma, D. P. & Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kipf, T. & Welling, M. (2016). Variational graph auto-encoders, 2016. In *Bayesian deep learning workshop (nips 2016), arxiv preprint (arxiv: 161107308).[google scholar]*.

Kipf, T. N. & Welling, M. (2016a). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kipf, T. N. & Welling, M. (2016b). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Ko, I.-Y., Ko, H.-G., Molina, A. J. & Kwon, J.-H. (2016). Soiot: Toward a user-centric iot-based service framework. *ACM Transactions on Internet Technology (TOIT)*, *16*(2), 1–21.

Kolos-Mazuryk, L., Poulisse, G.-J. & van Eck, P. (2005). Requirements engineering for pervasive services. In *Oopsla-workshop on building software for pervasive computing, san diego, usa.*

Koren, Y. (2008). Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th acm sigkdd international conference on knowledge discovery and data mining* (p. 426–434). New York, NY, USA: Association for Computing Machinery. Retrieved from `https://doi.org/10.1145/1401890.1401944` doi: 10.1145/1401890.1401944

Koren, Y. (2010). Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *4*(1), 1.

Krogstie, J. (2001). Requirement engineering for mobile information systems. In *Proceedings of the seventh international workshop on requirements engineering: Foundations for software quality (refsq'01)* (p. 74).

Le, Q. & Mikolov, T. (2014). Distributed representations of sentences and documents. In *International conference on machine learning* (pp. 1188–1196).

Lee, J. B., Rossi, R. A., Kong, X., Kim, S., Koh, E. & Rao, A. (2019). Graph convolutional networks with motif-based attention. In *Proceedings of the 28th acm international conference on information and knowledge management* (pp. 499–508).

Lee, J.-N., Huynh, M. Q., Kwok, R. C.-W. & Pi, S.-M. (2003). It outsourcing evolution—past, present, and future. *Communications of the ACM*, *46*(5), 84–89.

Lemos, A. L., Daniel, F. & Benatallah, B. (2015). Web service composition: a survey of techniques and tools. *ACM Computing Surveys (CSUR)*, *48*(3), 1–41.

Li, C., Zhang, R., Huai, J. & Sun, H. (2014). A novel approach for api recommendation in mashup development. In *2014 ieee international conference on web services* (pp. 289–296).

Li, S., Wen, J., Luo, F., Cheng, T. & Xiong, Q. (2017). A location and reputation aware matrix factorization approach for personalized quality of service prediction. In *2017 ieee international conference on web services (icws)* (pp. 652–659).

Li, Y., Tarlow, D., Brockschmidt, M. & Zemel, R. (2015). Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*.

Liang, D., Krishnan, R. G., Hoffman, M. D. & Jebara, T. (2018). Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 world wide web conference* (p. 689–698). Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee. Retrieved from `https://doi.org/10.1145/3178876.3186150` doi: 10.1145/3178876.3186150

Liang, T., Chen, L., Wu, J., Dong, H. & Bouguettaya, A. (2016). Meta-path based service recommendation in heterogeneous information networks. In *International conference on service-oriented computing* (pp. 371–386).

Liben-Nowell, D. & Kleinberg, J. (2007). The link-prediction problem for social networks. *Journal of the American society for information science and technology*, *58*(7), 1019–1031.

Lim, S. & Lee, J.-G. (2016). Motif-based embedding for graph clustering. *Journal of Statistical Mechanics: Theory and Experiment*, *2016*(12), 123401.

Linden, G., Smith, B. & York, J. (2003). Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, *7*(1), 76–80.

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

Liu, S. & Zheng, Y. (2020). Long-tail session-based recommendation. In *Fourteenth acm conference on recommender systems* (pp. 509–514).

Liu, Z., Guo, S., Wang, L., Du, B. & Pang, S. (2019). A multi-objective service composition recommendation method for individualized customer: hybrid mpa-gso-dnn model. *Computers & Industrial Engineering*, *128*, 122–134.

Lo, W., Yin, J., Deng, S., Li, Y. & Wu, Z. (2012). An extended matrix factorization approach for qos prediction in service selection. In *2012 ieee ninth international conference on services computing* (pp. 162–169).

Ludwig, H. & Petrie, C. (2006). 05462 session summary–" cross cutting concerns". In *Dagstuhl seminar proceedings.*

Luo, L., Liu, K., Peng, D., Ying, Y. & Zhang, X. (2020). A motif-based graph neural network to reciprocal recommendation for online dating. In *International conference on neural information processing* (pp. 102–114).

Ma, C., Ma, L., Zhang, Y., Sun, J., Liu, X. & Coates, M. (2019). Memory augmented graph neural networks for sequential recommendation. *arXiv preprint arXiv:1912.11730*.

Ma, Y., Geng, X. & Wang, J. (2020). A deep neural network with multiplex interactions for cold-start service recommendation. *IEEE Transactions on Engineering*

Management, *68*(1), 105–119.

Maamar, Z., Hacid, H. & Huhns, M. N. (2011). Why web services need social networks. *IEEE Internet Computing*, *15*(2), 90–94.

MacKenzie, C. M., Laskey, K., McCabe, F., Brown, P. F., Metz, R. & Hamilton, B. A. (2006). Reference model for service oriented architecture 1.0. *OASIS standard*, *12*(S 18).

Manes, A. T. (2001). Enabling open, interoperable, and smart web services—the need for shared context. In *Proc. w3c web services workshop, http://www. w3. org/2001/03/wsws-popa/paper29.*

Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., ... others (2004). Bringing semantics to web services: The owl-s approach. In *International workshop on semantic web services and web process composition* (pp. 26–42).

Martinez-Cruz, C., Porcel, C., Bernabé-Moreno, J. & Herrera-Viedma, E. (2015). A model to represent users trust in recommender systems using ontologies and fuzzy linguistic modeling. *Information Sciences*, *311*, 102–118.

Maximilien, E. M. & Singh, M. P. (2004). A framework and ontology for dynamic web services selection. *IEEE Internet Computing*, *8*(5), 84–93.

McIlraith, S. A., Son, T. C. & Zeng, H. (2001). Semantic web services. *IEEE intelligent systems*, *16*(2), 46–53.

Meng, S., Dou, W., Zhang, X. & Chen, J. (2014). Kasr: a keyword-aware service recommendation method on mapreduce for big data applications. *IEEE Transactions on Parallel and Distributed Systems*, *25*(12), 3221–3231.

Meyer, S., Ruppen, A. & Magerkurth, C. (2013). Internet of things-aware process modeling: integrating iot devices as business process resources. In *International conference on advanced information systems engineering* (pp. 84–98).

Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D. & Alon, U. (2002). Network motifs: simple building blocks of complex networks. *Science*, *298*(5594), 824–827.

Miorandi, D., Sicari, S., De Pellegrini, F. & Chlamtac, I. (2012). Internet of things: Vision, applications and research challenges. *Ad hoc networks*, *10*(7), 1497–1516.

Mnih, A. & Kavukcuoglu, K. (2013). Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in neural information processing systems* (pp. 2265–2273).

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

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G. & Grohe, M. (2019). Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 33, pp. 4602–4609).

Muñoz, J. & Pelechano, V. (2006). Applying software factories to pervasive systems: A platform specific framework. In *Iceis (3)* (pp. 337–342).

Munoz, J., Valderas, P., Pelechano, V. & Pastor, O. (2006). Requirements engineering for pervasive systems. a transformational approach. In *14th ieee international*

requirements engineering conference (re'06) (pp. 351–352).

Murphy, K., Weiss, Y. & Jordan, M. I. (2013). Loopy belief propagation for approximate inference: An empirical study. *arXiv preprint arXiv:1301.6725*.

Naïm, H., Aznag, M., Durand, N. & Quafafou, M. (2016). Semantic pattern mining based web service recommendation. In *International conference on service-oriented computing* (pp. 417–432).

Nguyen, M., Yu, J., Nguyen, T. & Han, Y. (2021). Attentional matrix factorization with context and co-invocation for service recommendation. *Expert Systems with Applications*, 115698.

Noor, T. H., Sheng, Q. Z., Ngu, A. H. & Dustdar, S. (2014). Analysis of web-scale cloud services. *IEEE Internet Computing*, *18*(4), 55–61.

Ou, M., Cui, P., Pei, J., Zhang, Z. & Zhu, W. (2016). Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 1105–1114).

Papazoglou, M. P. (2003). Service-oriented computing: Concepts, characteristics and directions. In *Proceedings of the fourth international conference on web information systems engineering, 2003. wise 2003.* (pp. 3–12).

Papazoglou, M. P. & Van Den Heuvel, W.-J. (2007). Service oriented architectures: approaches, technologies and research issues. *The VLDB journal*, *16*(3), 389–415.

Paradarami, T. K., Bastian, N. D. & Wightman, J. L. (2017). A hybrid recommender system using artificial neural networks. *Expert Systems with Applications*, *83*, 300–313.

Paranjape, A., Benson, A. R. & Leskovec, J. (2017). Motifs in temporal networks. In *Proceedings of the tenth acm international conference on web search and data mining* (pp. 601–610).

Park, C., Kim, D., Oh, J. & Yu, H. (2016). Improving top-k recommendation with truster and trustee relationship in user trust network. *Information Sciences*, *374*, 100–114.

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).

Pastore, S. (2008). The service discovery methods issue: A web services uddi specification framework integrated in a grid environment. *Journal of Network and Computer Applications*, *31*(2), 93–107.

Pautasso, C., Zimmermann, O. & Leymann, F. (2008). Restful web services vs." big"'web services: making the right architectural decision. In *Proceedings of the 17th international conference on world wide web* (pp. 805–814).

Peltz, C. (2003). Web services orchestration and choreography. *Computer*, *36*(10), 46–52.

Perera, C., Zaslavsky, A., Christen, P. & Georgakopoulos, D. (2013). Context aware computing for the internet of things: A survey. *IEEE communications surveys & tutorials*, *16*(1), 414–454.

Perozzi, B., Kulkarni, V. & Skiena, S. (2016). Walklets: Multiscale graph embeddings

for interpretable network classification. *arXiv preprint arXiv:1605.02115*, 043238–23.

Prill, R. J., Iglesias, P. A. & Levchenko, A. (2005). Dynamic properties of network motifs contribute to biological network organization. *PLoS biology*, *3*(11), e343.

Rahman, M. M., Liu, X. & Cao, B. (2017). Web api recommendation for mashup development using matrix factorization on integrated content and network-based service clustering. In *2017 ieee international conference on services computing (scc)* (pp. 225–232).

Rao, J. & Su, X. (2004). A survey of automated web service composition methods. In *International workshop on semantic web services and web process composition* (pp. 43–54).

Rendle, S., Freudenthaler, C., Gantner, Z. & Schmidt-Thieme, L. (2012). Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*.

Rezende, D. J., Mohamed, S. & Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*.

Rinne, H. (2008). *The weibull distribution: a handbook*. Chapman and Hall/CRC.

Roh, T. H., Oh, K. J. & Han, I. (2003). The collaborative filtering recommendation based on som cluster-indexing cbr. *Expert systems with applications*, *25*(3), 413–423.

Rong, W. & Liu, K. (2010). A survey of context aware web service discovery: From user's perspective. In *2010 fifth ieee international symposium on service oriented system engineering* (pp. 15–22).

Rossi, R. A., Ahmed, N. K. & Koh, E. (2018). Higher-order network representation learning. In *Companion proceedings of the the web conference 2018* (pp. 3–4).

Rossi, R. A., Zhou, R. & Ahmed, N. K. (2018). Deep inductive network representation learning. In *Companion proceedings of the the web conference 2018* (pp. 953–960).

Rostami, N. H., Kheirkhah, E. & Jalali, M. (2013). Web services composition methods and techniques: A review. *International Journal of Computer Science, Engineering & Information Technology*, *3*(6), 10–5121.

Salton, G. & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing & management*, *24*(5), 513–523.

Samanta, P. & Liu, X. (2017). Recommending services for new mashups through service factors and top-k neighbors. In *2017 ieee international conference on web services (icws)* (pp. 381–388).

Sandvig, J. J., Mobasher, B. & Burke, R. D. (2008). A survey of collaborative recommendation and the robustness of model-based algorithms. *IEEE Data Eng. Bull.*, *31*(2), 3–13.

Sarwar, B., Karypis, G., Konstan, J. & Riedl, J. (2001a). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on world wide web* (pp. 285–295).

Sarwar, B., Karypis, G., Konstan, J. & Riedl, J. (2001b). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on world wide web* (pp. 285–295).

Schlichtkrull, M., Kipf, T. N., Bloem, P., Van Den Berg, R., Titov, I. & Welling, M. (2018). Modeling relational data with graph convolutional networks. In *European semantic web conference* (pp. 593–607).

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).

Shen, W., Hao, Q., Wang, S., Li, Y. & Ghenniwa, H. (2007). An agent-based service-oriented integration architecture for collaborative intelligent manufacturing. *Robotics and Computer-Integrated Manufacturing*, *23*(3), 315–325.

Sheng, Q. Z., Qiao, X., Vasilakos, A. V., Szabo, C., Bourne, S. & Xu, X. (2014). Web services composition: A decade's overview. *Information Sciences*, *280*, 218–238.

Sim, K. M. (2011). Agent-based cloud computing. *IEEE transactions on services computing*, *5*(4), 564–577.

Sun, Z., Guo, Q., Yang, J., Fang, H., Guo, G., Zhang, J. & Burke, R. (2019). Research commentary on recommendations with side information: A survey and research directions. *Electronic Commerce Research and Applications*, *37*, 100879.

Tan, Y. K., Xu, X. & Liu, Y. (2016). Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st workshop on deep learning for recommender systems* (pp. 17–22).

Tang, M., Zhang, T., Liu, J. & Chen, J. (2015). Cloud service qos prediction via exploiting collaborative filtering and location-based data smoothing. *Concurrency and Computation: Practice and Experience*, *27*(18), 5826–5839.

Tay, Y., Zhang, S., Luu, A., Hui, S., Yao, L. & Vinh Tran, L. (2019, 07). Holographic factorization machines for recommendation. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*, 5143-5150. doi: 10.1609/aaai.v33i01.33015143

Tian, G., Wang, J., He, K., Sun, C. & Tian, Y. (2017). Integrating implicit feedbacks for time-aware web service recommendations. *Information systems frontiers*, *19*(1), 75–89.

Tsai, W.-T., Sun, X. & Balasooriya, J. (2010). Service-oriented cloud computing architecture. In *2010 seventh international conference on information technology: new generations* (pp. 684–689).

Tselentis, G., Domingue, J. & Galis, A. (2009). *Towards the future internet: A european research perspective*. IOS press.

Vishwanathan, S. V. N., Schraudolph, N. N., Kondor, R. & Borgwardt, K. M. (2010). Graph kernels. *Journal of Machine Learning Research*, *11*, 1201–1242.

Wang, D., Cui, P. & Zhu, W. (2016a). Structural deep network embedding. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 1225–1234).

Wang, D., Cui, P. & Zhu, W. (2016b). Structural deep network embedding. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 1225–1234).

Wang, X., He, X. & Chua, T.-S. (2020a). Learning and reasoning on graph for recommendation. In *Proceedings of the 13th international conference on web search and data mining* (pp. 890–893).

Wang, X., He, X. & Chua, T.-S. (2020b). Learning and reasoning on graph for recommendation. In *Proceedings of the 13th international conference on web search and data mining* (pp. 890–893).

Wang, X., He, X., Wang, M., Feng, F. & Chua, T.-S. (2019). Neural graph collaborative filtering. In *Proceedings of the 42nd international acm sigir conference on research and development in information retrieval* (pp. 165–174).

Wei, Y. & Blake, M. B. (2010). Service-oriented computing and cloud computing: Challenges and opportunities. *IEEE Internet Computing*, *14*(6), 72–75.

Wu, H., Yue, K., Li, B., Zhang, B. & Hsu, C.-H. (2018a). Collaborative qos prediction with context-sensitive matrix factorization. *Future Generation Computer Systems*, *82*, 669–678.

Wu, H., Yue, K., Li, B., Zhang, B. & Hsu, C.-H. (2018b). Collaborative qos prediction with context-sensitive matrix factorization. *Future Generation Computer Systems*, *82*, 669–678.

Wu, Y., DuBois, C., Zheng, A. X. & Ester, M. (2016). Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the ninth acm international conference on web search and data mining* (pp. 153–162).

Xiao, J., Ye, H., He, X., Zhang, H., Wu, F. & Chua, T.-S. (2017). Attentional factorization machines: Learning the weight of feature interactions via attention networks. *arXiv preprint arXiv:1708.04617*.

Xie, F., Chen, L., Lin, D., Zheng, Z. & Lin, X. (2019). Personalized service recommendation with mashup group preference in heterogeneous information network. *IEEE Access*, *7*, 16155–16167.

Xie, F., Wang, J., Xiong, R., Zhang, N., Ma, Y. & He, K. (2019). An integrated service recommendation approach for service-based system development. *Expert Systems With Applications*, *123*, 178–194.

Xiong, R., Wang, J., Zhang, N. & Ma, Y. (2018). Deep hybrid collaborative filtering for web service recommendation. *Expert systems with Applications*, *110*, 191–205.

Xu, W., Cao, J., Hu, L., Wang, J. & Li, M. (2013). A social-aware service recommendation approach for mashup creation. In *2013 ieee 20th international conference on web services* (pp. 107–114).

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.

Xue, H.-J., Dai, X., Zhang, J., Huang, S. & Chen, J. (2017). Deep matrix factorization models for recommender systems. In *Ijcai* (Vol. 17, pp. 3203–3209).

Yang, C., Liu, M., Zheng, V. W. & Han, J. (2018). Node, motif and subgraph: Leveraging network functional blocks through structural convolution. In *2018 ieee/acm international conference on advances in social networks analysis and mining (asonam)* (pp. 47–52).

Yang, J.-H., Chen, C.-M., Wang, C.-J. & Tsai, M.-F. (2018). Hop-rec: high-order proximity for implicit recommendation. In *Proceedings of the 12th acm conference on recommender systems* (pp. 140–144).

Yao, L., Sheng, Q. Z., Ngu, A. H., Yu, J. & Segev, A. (2014). Unified collaborative

and content-based web service recommendation. *IEEE Transactions on Services Computing*, *8*(3), 453–466.

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*.

Yao, L., Wang, X., Sheng, Q. Z., Benatallah, B. & Huang, C. (2021, apr). Mashup recommendation by regularizing matrix factorization with api co-invocations. *IEEE Transactions on Services Computing*, *14*(02), 502-515. doi: 10.1109/TSC.2018.2803171

Yao, L., Wang, X., Sheng, Q. Z., Ruan, W. & Zhang, W. (2015). Service recommendation for mashup composition with implicit correlation regularization. In *2015 ieee international conference on web services* (pp. 217–224).

Yin, Y., Xu, H., Liang, T., Chen, M., Gao, H. & Longo, A. (2021, March). Leveraging data augmentation for service qos prediction in cyber-physical systems. *ACM Trans. Internet Technol.*, *21*(2). Retrieved from https://doi.org/10.1145/3425795 doi: 10.1145/3425795

Zhang, H., Shen, F., Liu, W., He, X., Luan, H. & Chua, T.-S. (2016). Discrete collaborative filtering. In *Proceedings of the 39th international acm sigir conference on research and development in information retrieval* (p. 325–334). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/2911451.2911502 doi: 10.1145/2911451.2911502

Zhang, N., Wang, J. & Ma, Y. (2017). Mining domain knowledge on service goals from textual service descriptions. *IEEE Transactions on Services Computing*, *13*(3), 488–502.

Zhang, S., Hu, Z., Subramonian, A. & Sun, Y. (2020). Motif-driven contrastive learning of graph representations. *arXiv preprint arXiv:2012.12533*.

Zhang, S., Yao, L., Sun, A. & Tay, Y. (2019). Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, *52*(1), 1–38.

Zhang, Y.-C., Blattner, M. & Yu, Y.-K. (2007). Heat conduction process on community networks as a recommendation model. *Physical review letters*, *99*(15), 154301.

Zhang, Y.-C., Medo, M., Ren, J., Zhou, T., Li, T. & Yang, F. (2007). Recommendation model based on opinion diffusion. *EPL (Europhysics Letters)*, *80*(6), 68003.

Zhao, J., Wang, X., Shi, C., Hu, B., Song, G. & Ye, Y. (2021). Heterogeneous graph structure learning for graph neural networks. In *35th aaai conference on artificial intelligence (aaai)*.

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.

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., Tan, W. & Zhang, J. (2016). Web service recommendation with reconstructed profile from mashup descriptions. *IEEE Transactions on Automation Science and Engineering*, *15*(2), 468–478.

Zou, G., Jiang, M., Niu, S., Wu, H., Pang, S. & Gan, Y. (2018). Qos-aware web service recommendation with reinforced collaborative filtering. In *International conference on service-oriented computing* (pp. 430–445).