applied sciences

MDPI

Article

Multi-Relational Graph Convolution Network for Service Recommendation in Mashup Development

Wei Gao * and Jian Wu

Department of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China; wujian2000@zju.edu.cn
* Correspondence: gw@zju.edu.cn

Abstract: With the rapid development of service-oriented computing, an overwhelming number of web services have been published online. Developers can create mashups that combine one or multiple services to meet complex business requirements. To speed up the mashup development process, recommending suitable services for developers is a vital problem. In this paper, we address the data sparsity and cold-start problems faced in service recommendation, and propose a novel multi-relational graph convolutional network framework (MRGCN) for service recommendation. Specifically, we first construct a multi-relational mashup-service graph with three types of relations, namely composition relation, functional relation, and tagging relation. These three relations are indispensable and complement each other for capturing multi-view information. Then, the three relations in the graph are seamlessly fused with various strategies. Next, graph convolution is performed on the fused multi-relational graph to capture the high-order relational information between mashups and services. Finally, the relevance between mashup requirements and services is predicted based on the learned features on the graph. We conduct extensive experiments on the ProgrammableWeb dataset and demonstrate that our proposed method can outperform state-of-the-art methods in recommending services when only mashup requirements are available.

Keywords: service recommendation; mashup application; graph convolutional network

Citation: Gao, W.; Wu, J. Multi-Relational Graph Convolution Network for Service Recommendation in Mashup Development. *Appl. Sci.* **2022**, *12*, 924. https://doi.org/10.3390/app12020924

Academic Editor: Kuen-Suan Chen

Received: 22 December 2021
Accepted: 14 January 2022
Published: 17 January 2022

1. Introduction

Service-oriented computing (SOC) has become a significant paradigm for developing low-cost and reliable software applications in software engineering and cloud computing [1]. Web services are the basic build blocks of service-oriented computing, which encapsulate application functionalities and can be accessed through standard interfaces [2]. Nowadays, an increasing number of web services, mainly in the form of RESTful Web APIs, have been published online. According to the recent statistics of *ProgrammableWeb*, the largest Web API portal, there are over 24,000 Web APIs available online. However, the functionality of an individual service is limited and cannot satisfy the complex requirements of developers. As a result, it is common for developers to compose existing services and develop value-added services, also called mashups [3,4]. For example, a developer may create a new mashup that can display ratings and reviews of restaurants on a region map by integrating Google Map service and Yelp Service. However, creating a mashup can be difficult and time-consuming for the inexperienced developer due to the overwhelming number of services on the Internet. Therefore, it is vital to proactively recommend appropriate services that can satisfy the developer's complex requirements and ease the selection burden.

Previous studies on service recommendation can be classified into three categories: collaborative filtering-based approach, content-based approach, and hybrid approach [5]. Collaborative filtering-based approaches utilize the historical composition data between mashups and services [6–10]. Content-based approaches exploit the information of user

requirements and service descriptions, which could be in the form of structured text such as Web services description language (WSDL) or unstructured natural language descriptions [11–14]. Hybrid approaches combine collaborative filtering and content-based approaches by exploiting both content information and composition history [15–18].

However, despite the recent research progress in service recommendation, it still faces several challenges: (1) Sparsity of the mashup-service composition relation. The number of services is large and most mashups are composed of a few services; as a result, the mashup-service composition matrix is extremely sparse. Take ProgrammableWeb as an example; the sparsity of the matrix is 99.87%. Traditional approaches such as matrix factorization are difficult to model the extremely sparse mashup-service relation. (2) Cold-start problem for mashups and services. When a developer creates a new mashup, it has no component services; as a result, it is difficult to recommend existing services using the traditional collaborative filtering model due to a lack of historical information. A similar issue occurs for a new service that has no composition history. (3) Integration of multiple information sources. Mashups and services have diverse information available, such as the composition relation between mashup and service, functional descriptions, and annotated tags of the mashup and service, etc. How to comprehensively utilize and fuse the available information for service recommendation is a non-trivial task.

In this paper, we propose a multi-relational graph convolution network (MRGCN) service recommendation approach to address the above challenges. To address the first challenge, we adopt a neural graph convolution network model that has been widely used for modeling graph-structured data. Compared with the traditional collaborative filtering approaches like matrix factorization, the graph convolution network exploits the high-order relations between mashup and service with a nonlinear interaction. Therefore, it has a more powerful representation capability and can capture more complex relations between mashups and services. To address the second challenge, we propose to model the historical composition record and functional content of the mashup and service as a multi-relational graph. The nodes are mashups and services and the edges belong to one of the three relations: composition relation, functional relation, and tagging relation. The three relations complement each other and all mashups and services are connected in the multi-relational graph, thus obviating the cold-start issue. To address the third challenge, we propose to fuse the multi-relational graph with three relations using several fusion strategies. The fused graph effectively captures the prominent information of the three relations. In particular, our model is extensible and can incorporate different kinds of relations when other side information is available.

The main contributions of this article are threefold:

1. We construct a multi-relational graph that incorporates the composition history, functional descriptions, and annotated tags of the mashup and service;
2. We propose to fuse the multi-relational graph and utilize the graph convolutional network to capture the high-order relation for service recommendation;
3. We conduct a series of experiments on real-world services from ProgrammableWeb, and the results demonstrate the effectiveness of our proposed approach;

The rest of this article is organized as follows: Section 2 presents the related work of service recommendation. Section 3 introduces the details of our proposed approach. Section 4 reports the experimental results and analysis. Section 5 concludes this paper with future work.

2. Related Work

Web service recommendation for mashup construction has been widely researched in recent years. Similar to the taxonomy of general recommender system approaches [5], the service recommendation approach can be divided into three categories: collaborative filtering (CF) based approach, content-based approach, and hybrid approach [19].

CF-based approaches recommend services from the historical composition data of the constructed mashups and their services. It can be further divided into neighborhood-

based approaches and model-based approaches such as matrix factorization. For example, Xu et al. propose a coupled matrix factorization model to predict the multi-dimensional relations among users, mashups, and services [6]. Gao et al. propose using a generalized manifold ranking algorithm on the graph with relations among mashups and services [20]. Liang et al. measure the similarity between mashups based on a heterogeneous information network (HIN) with various meta-paths, and make recommendations similar to user-based CF [7]. Based on this model, Xie et al. propose to measure functional similarities between mashups based on the word vectors of content learned with the word embedding technique [8]; Xie et al. propose a mashup group preference-based service recommendation on HIN, where mashup group preference is utilized to capture the rich interactions among mashups [9]. They also propose to use factorization machines to model the features of mashups and services learned from different kinds of meta-paths on HIN [10].

Content-based approaches recommend services by matching the content similarities between candidate services and the mashup requirements. It can be further divided into three categories: keyword-based approach, latent semantic-based approach, and deep learning-based approach. Keyword-based approaches extract keywords or salient entities from the content and use them to match user requests. For instance, Meng et al. propose to extract keywords from a keyword-candidate list and domain thesaurus to indicate users' preferences [11]. Latent semantic-based approaches usually extract content features using topic models. For instance, Aznag et al. propose Correlated Topic Model (CTM) to extract topics from semantic service descriptions and model their correlations [12]; Gao et al. propose to extract textual features with latent dirichlet allocation (LDA) [21]. Deep learning-based approaches provide a more powerful representation capability for modeling content; thus, they have been widely adopted in recent work. For instance, Bai et al. propose to use stacked denoising autoencoders (SDAE) to perform feature extraction from the content and impose the composition records as a regularization [13]; Shi and Liu propose a Long Short-Term Memory-based (LSTM) model to recommend services with a functional attention mechanism and a contextual attention mechanism [14].

Hybrid approaches recommend services by incorporating various factors including mashup-service composition history, functionalities of mashups and services, and other contextual information. Yao et al. unify both collaborative filtering and content-based recommendations with both rating and content data of services using a probabilistic generative model [16]. Jain et al. incorporate three factors, namely service functionality, usage history, and popularity for recommendation using topic models, matrix factorization, and Bayes' theorem [17]. Xiong et al. propose a model that integrates collaborative filtering and textual content into a deep neural network [18]. Ma et al. propose a multiplex interaction-oriented service recommendation approach where they extract hidden structures and features from various types of interactions between mashups and services and incorporate them into a deep neural network [19]. Xiong et al. propose a service recommendation approach via merging semantic features extracted from descriptions using NLP pipelines, and structural embeddings from the mashup-service network learned from biased random walks [22]. Wang et al. design a knowledge graph to encode the mashup-service relations and exploit random walks with restart to assess their similarities [23]. Later, they learn implicit low-dimensional embeddings of entities in the knowledge graph from truncated random walks [24]. Dang et al. propose a deep knowledge-aware approach for service recommendation that learns text and knowledge graph embeddings and use an attention mechanism to model tags [25]. Nguyen et al. learn the context of mashups and services by using Doc2Vec and enhance the traditional PMF with attentional mechanism to weight their latent features [26].

Our model also falls into the hybrid category. Different from other works, our method naturally merges the different types of information as interactions between mashups and services on a graph, and uses GCN to further exploit the high-order relatedness between mashups and services. Previous works [27,28] also adopt GCN-related techniques for service recommendation. In [27], Zhang et al. propose an end-to-end approach based on

GCN, where they utilize the service composition relationship in the mashup to construct the service graph, with the textual description as side information, and use a variational graph auto-encoder model as a link prediction task in the graph. However, their method constructs a homogeneous graph of services, and can only detect potential composition patterns in the service network. Our method constructs a multi-relational graph with both mashups and services, and can address the task when only mashup requirements are available without any composing service. In [28], Lian and Tang propose a neural graph collaborative filtering technique for API recommendation. However, their method only exploits high-order connections between users and APIs, and neglects other potentially relevant side information. Therefore, they cannot deal with the cold-start issue where no mashup-service connection information is available.

3. Multi-Relational GCN Based Service Recommendation Model

In this section, we will describe our multi-relational graph convolution network-based service recommendation model in detail. First, in Section 3.1, we describe the service recommendation for the new mashup development problem. Next, we present the overall framework in Section 3.2, and the main components of the model, namely multi-relational graph construction, graph convolution with relation fusion, and model prediction are described in Sections 3.2.1–3.2.3, respectively. Finally, we describe the training process and the model complexity in Section 3.3.

3.1. Problem Definition

In this section, we formulate our problem with necessary notations. Let $M = \{m_1, m_2, \ldots, m_{NM}\}$ be a set of NM mashups, and $S = \{s_1, s_2, \ldots, s_{NS}\}$ be a set of NS services. $N = NM + NS$ represents the total number of mashups and services. $MS = \{(m_1, s_1), (m_1, s_2), \ldots\}$ represents the set of mashup-service pairs that have a composition relation. For each mashup and service, we also collect its textual descriptions and tags. $MD = \{(m_1, d_{m_1}), (m_2, d_{m_2}), \ldots, (m_{NM}, d_{m_{NM}})\}$ denotes the set of mashups and their textual descriptions, where $d_m = w_1, w_2, \ldots$ represents the sequence of words after pre-processing the descriptions of mashup m. Similarly, $SD = \{(s_1, d_{s_1}), (s_2, d_{s_2}), \ldots, (s_{NS}, d_{s_{NS}})\}$ denotes the set of services and their textual descriptions. $MT = \{(m_1, T_1), (m_2, T_2), \ldots, (m_{NM}, T_{NM})\}$ represents the set of mashups and annotated tags, where $T_m = \{t_1, t_2, \ldots\}$ is the set of tags of each mashup m. Similarly, $ST = \{(s_1, T_1), (s_2, T_2), \ldots, (s_{NS}, T_{NS})\}$ represents the set of services and their annotated tags.

The problem can be formulated as follows: Given mashup-service composition relation MS, textual descriptions of mashups and services MD and SD, tags of mashups and services MT and ST, for a mashup m, we aim to recommend a list of services that are likely to be composed by m. In this paper, we consider the case where the mashup is newly created by the developer, i.e., no composition record is registered and only textual descriptions and tags are available. As a concrete example, a developer tries to develop a mashup that can make music recommendations and provide explanations by using linked data and semantic web techniques. He or she can input the requirement in the form of text, and specify a set of tags such as "*music*", "*recommendation*", "*semantics*" to refine the requirements. The system aims to recommend suitable services that can meet the developer's requirements. In this case, *Last.fm* service which can retrieve music resources from *lastfm* (https://www.last.fm/, accessed on 21 December 2021), and *DBpedia* service which organizes the music metadata as a structured knowledge web can be recommended.

3.2. Overall Framework

We present an overall framework of our proposed approach, which is shown in Figure 1. Our approach consists of three main components: multi-relational graph construction, graph convolution with relation fusion, and model prediction. Firstly, we construct three graphs with three different relations between mashup and service based on mashup-service composition relation, textual descriptions, and annotated tags of mashup and

service. Then, we use three strategies to fuse the three types of relations and adopt a graph convolutional neural network (GCN) architecture to propagate high-order relational information between mashups and services. Finally, the embeddings of mashups and services from different propagation layers are combined and the model outputs a preference score between a mashup-service pair.

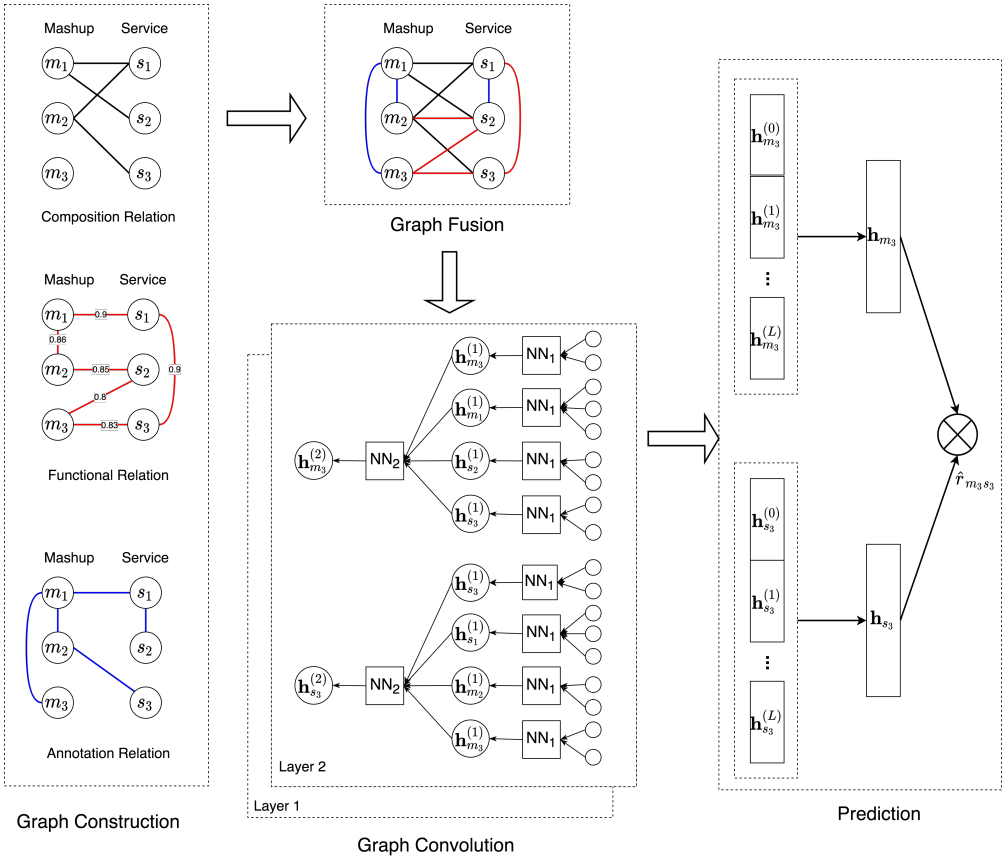

Figure 1. Overall framework of MRGCN for service recommendation.

3.2.1. Graph Construction

To fully exploit different kinds of relations between mashups and services and alleviate the sparsity and cold-start problem of the mashup-service composition record, we construct a multi-relational mashup-service graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R}, \mathcal{W})$, where the node set $\mathcal{V} = M \cup S$ includes all mashups and services, and $\mathcal{E}$ denotes the edges between the nodes if there exist a certain relation. $\mathcal{R}$ denotes the relation types of edges, and $\mathcal{R} = \{\mathcal{R}_C, \mathcal{R}_D, \mathcal{R}_T\}$ represents the three types of relations: composition relation, functional relation, and tagging relation. $\mathcal{W}$ denotes the weights of edges for each relation type. Next, we describe the construction process of the three types of relations in detail:

- Composition Relation. The composition relation between the mashup and service can be modeled as a bipartite graph, where an edge of type $\mathcal{R}_C$ exists between mashup m and service s if (m, s) appears in MS. The composition relation is represented as an adjacency matrix $A^C \in \{0,1\}^{N \times N}$ with $A^C(i,j) = 1$ if nodes i and j have a composition relation in MS and 0 otherwise. Note that the matrix A^C is extremely sparse as most mashups typically only invoke a few services to meet their demands. In addition, there exist some mashups and services that have no composition history, i.e., the rows and columns of matrix A^C of the cold-start mashups and services are zero vectors. To mitigate the sparsity and cold-start issues of the composition relation, other features of services and mashups such as textual descriptions and tags are utilized.
- Functional Relation. The textual descriptions of mashups and services provide functional properties and can alleviate the sparsity and cold-start problem of the compo-

sition relation. For instance, for a new mashup, m has no composition record, and we can still recommend a possible service s if mashup m' is functionally similar to m and m' invokes service s. We represent the functional relation as a graph where two nodes are connected with type $\mathcal{R}_D$ if their textual descriptions are similar to each other. It is crucial to measure the similarities of textual descriptions between mashups and services. In this paper, we adopt two approaches for learning the feature representations from textual descriptions and the similarity between them:

- LDA [29]. Latent Dirichlet Allocation (LDA) is an unsupervised generative model that can discover topics in a collection of documents and assign topic distributions to each document. We collect the textual descriptions of all mashups and services as a document corpus. After training, the textual description of each mashup/service will have a distribution over all topics, and each topic will have a distribution over all words. Specifically, each mashup/service is mapped to a latent topic space and represented as a topic distribution $d \in \mathbb{R}^{NK}$, where NK is the number of topics.
- Doc2vec [30]. Different from LDA that processes documents as "bag-of-words" where the order of words is ignored, the doc2vec model learns a distributed vector representation for a piece of document. It follows the idea of word2vec where the target word is predicted given context words. Concretely, it concatenates the document vector with several word vectors from the document and predicts the following word in a context window. Compared to word2vec, the only change is the addition of the document vector that acts as a memory of the topic of the document. After training, the textual description of the mashup/service is represented as a dense vector $d \in \mathbb{R}^{ND}$, where ND is the dimension size.

We denote the description embedding matrix as $D \in \mathbb{R}^{N \times NK}$ or $D \in \mathbb{R}^{N \times ND}$, where each row represents the vector representation of the textual description of mashup or service learned from LDA or doc2vec model. The cosine similarity is used to measure the similarity between vectors. We denote the weight matrix for the functional relation $\mathcal{R}_D$ as $W^D \in \mathbb{R}^{N \times N}$, and the functional similarity between any two nodes i and j is measured as $W^D(i,j) = \frac{D_i \cdot D_j}{\|D_i\| \|D_j\|}$. In this case, each node is connected to all other nodes in the graph of relation type $\mathcal{R}_D$ with weights equal to their functional similarity. However, modeling the functional relation as a complete graph is unfavorable as a mashup or service is generally only similar to a few mashups and services, as such, a large proportion of edges with small weights will introduce a lot of noise and increase the memory and computation costs. Therefore, we prune the graph where only τ_1 edges with the largest weights for each node are retained. In this way, each mashup/service is connected to at least τ_1 most similar mashups and services in terms of textual descriptions, and the edge weights are the degrees of similarity. The functional relation is represented as an adjacency matrix $A^D \in \mathbb{R}^{N \times N}$, which only preserves the significant values of W^D.

- Tagging Relation. There is a tagging relation $\mathcal{R}_T$ between the nodes if two nodes share at least one common tag. The tagging relation is represented as an adjacency matrix $A^T \in \{0,1\}^{N \times N}$ with $A^T(i,j) = 1$ if nodes i and j have at least a common tag, i.e., $T_i \cap T_j \neq \emptyset$, and otherwise 0. As some popular tags are annotated by a lot of mashups and services, a service may possess a large number of tagging relations. To reduce the computational burden in the graph convolution, for each node, we randomly sample τ_2 nodes that share the same tag. In this case, the tagging relation can still be preserved as the graph convolution process can learn the indirect tagging relation between services.

In the end, we construct graphs with three relations represented as three adjacency matrices A^C, A^D, and A^T. Figure 1 shows an example of the graphs with three mashups and three services. Different relations are denoted by different edge colors, and the functional relation has associated weights. We can see that, while mashup m_3 has no composition

relation with other services, it has a rich interaction with other mashups and services in terms of functional and tagging relations. Note that other types of relations between nodes can be incorporated, such as services developed by the same company.

3.2.2. Graph Convolution

In the previous section, we enrich the information beyond mashup-service composition record with content and tag information, and build a graph with three relations. In this section, we propose to exploit the multi-faceted relations between mashups and services with a graph convolutional network with relation fusion. Graph convolutional network (GCN) extends the convolutional neural network to graph-structured data, and it exploits the high-order interactions between the nodes [31]. The core idea behind GCN is to iteratively aggregate feature information from local neighbors for each node using neural networks. A single layer of GCN transforms and aggregates feature information from the node's direct neighbors, and by stacking multiple layers, feature information can be propagated across long ranges, and the node features can be enhanced with sufficient high-order neighbor information. The reason we use GCN is that it can leverage both node features as well as graph structure, and it can explore the relation information among nodes that are not directly connected. We first describe the GCN model for a single type of relation, then we extend it with a fusion of multiple relations.

Given the mashup-service graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R}_*, \mathcal{W})$ with a single relation, where $\mathcal{R}_*$ could be either $\mathcal{R}_C$, $\mathcal{R}_D$ or $\mathcal{R}_T$, in each layer of GCN, the goal is to learn a function which takes the features of every node and the graph structure represented as an adjacency matrix, and produce the transformed features of each node. Supposing that the adjacency matrix representing that relation is A, the function for the graph propagation step is:

$$H^{(l+1)} = \text{ReLU}\Big(D^{-\frac{1}{2}}(A+I)D^{-\frac{1}{2}}H^{(l)}W^{(l)}\Big), \tag{1}$$

where $H^{(l)} \in \mathbb{R}^{N \times d_l}$ is the feature matrix at the l-th layer, $W^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}}$ is the weight matrix for feature transformation in the l-th layer. I is the identity matrix, and D is the diagonal degree of matrix $A + I$, which is calculated as $D_{ii} = 1 + \sum_j A_{ij}$. ReLU is the nonlinear activation function.

To write Equation (1) in a vector form for a single node i:

$$h_i^{(l+1)} = \text{ReLU}\Big(\sum_j \frac{1}{\sqrt{|N_i||N_j|}} W^{(l)} h_j^{(l)}\Big), \tag{2}$$

where j is the neighboring node of node i including node i itself. N_i and N_j denote the neighbors of node i and j. We can see that the feature propagation in each layer consists of several steps: first, the feature of each node is transformed through the matrix $W^{(l)}$, then the transformed feature is propagated to the neighboring node, and each node receives the features from its neighboring nodes including itself and adds them together with a normalization term to maintain the scale of the feature vectors. Finally, the aggregated features are going through the ReLU activation function to encode only positive signals.

By stacking multiple propagation layers defined in Equation (1), each node is capable of receiving features from nodes that are l-hop away, and the higher-order relation between mashups and services can be explored. Such high-order relation information can encode the potential relation between the mashup and service that are not connected directly but are 2-hops or 3-hops away, and this latent relation can be crucial to infer the potential composition relation between the mashup and service.

Figure 1 shows an example of a 2-layer GCN that computes the feature representation $h_{m_3}^{(2)}$ of mashup m_3 in layer 2. It aggregates the features learned from the previous layer $h_{m_3}^{(1)}$ and its neighbors $h_{m_1}^{(1)}$, $h_{s_2}^{(1)}$, and $h_{s_3}^{(1)}$, and the feature representations in layer 1 are aggregated from the initial node features. $h_{s_3}^{(2)}$ are learned in the same way. The NN_l

module in the graph represents the feature transformation, aggregation, and nonlinear activation process of layer l, and they share the same parameters $W^{(l)}$ for each layer.

Thus far, we have described the GCN framework on a single relation graph. As there exist three kinds of relations, we propose to fuse the multiple relations into a single relation with three fusion strategies. Inspired by [32], we take the symmetric normalized Laplacian matrices of the graph for the fusion of multiple relations, which can enrich the structural information propagated through nodes. Specifically, the symmetric normalized Laplacian matrix of the composition relation A^C is defined as $L^C = D_C^{-\frac{1}{2}}(A^C + I)D_C^{-\frac{1}{2}}$, where D_C is the degree matrix of $A^C + I$. L^D and L^T are defined in a similar manner that corresponds to the matrices A^D and A^T. We propose three fusion strategies: (1) The *add* strategy. It integrates the graph structure of three relations through adding the Laplacian matrices, i.e., $L_{add} = L^C + L^D + L^T$; (2) The *max-pooling* strategy. It takes the maximum value of the three Laplacian matrices in each entry—for example, if $L^C(i,j) = 0.9$, $L^D(i,j) = 0.8$, $L^T(i,j) = 0.5$, and $L_{max-pool}(i,j) = 0.9$; (3) The *weight-sum* strategy. It adds three Laplacian matrices with different weights, which can integrate relations with different levels of importance. In particular, $L_{weight-sum} = w_c \cdot L^C + w_d \cdot L^D + w_t \cdot L^T$, where w_c, w_d, and w_t are scalar values that multiply with each entry in the corresponding matrix, and we treat them as model parameters. In this way, the three relations are jointly considered and directly incorporated into a comprehensive relation by fusing the Laplacian matrices with different strategies. Alternatively, we can perform GCN directly on the three graphs with one single type of relation, and concatenate the final layer of the learned features of the three graphs. However, it adds the computation complexity of GCN training, and it is hard to tune the feature dimensions for each relation. We will evaluate different fusion strategies in the experiments.

3.2.3. Prediction

As we perform GCN on a fused graph with L layers, we obtain multiple layers of feature representations for each mashup m and service s, namely $h_m^{(1)}, \ldots, h_m^{(L)}$ and $h_s^{(1)}, \ldots, h_s^{(L)}$. Since the feature representations obtained in different layers reflect the information aggregated from different nodes, we concatenate them to constitute the final representation for the mashup m, i.e., $h_m = h_m^{(0)} \parallel \cdots \parallel h_m^{(L)}$, and $\parallel$ is the concatenation operation. Similarly, for service s, $h_s = h_s^{(0)} \parallel \cdots \parallel h_s^{(L)}$. Note that $h_m^{(0)}$ and $h_s^{(0)}$ are the initial feature vectors for mashup m and service s, which are the rows of the feature matrix $H^{(0)}$. In this case, we enrich the initial representations for mashups/services with representations from all layers of GCN capturing the higher-order relational information.

Once the final representations of mashup and service are obtained, we use the inner product to estimate the composition preference of mashup m to service s. Since we aim to output the possibility of service s being selected as a component of mashup m, we use the sigmoid activation function to constrain the value between 0 and 1:

$$\hat{r}_{ms} = \sigma(h_m^T \cdot h_s). \tag{3}$$

3.3. Model Learning

We design a loss function which inputs a pair of a mashup and a service to learn the model parameters. Pointwise and pairwise loss are two main types of loss functions used for service recommendation. Since the mashup-service interaction matrix is extremely sparse as most mashups only invoke a few services, we opt for the pointwise loss function [18]. Specifically, we use a binary cross-entropy loss function:

$$\mathcal{J} = -\sum_{(m,s)\in R^+ \cup R^-} \left(r_{ms} \log \hat{r}_{ms} + (1 - r_{ms}) \log(1 - \hat{r}_{ms})\right) + \lambda ||\Theta||_2^2, \tag{4}$$

where R^+ denotes the set of positive samples consisting of mashups and its invoked services, and r_{ms} is labeled as 1; R^- denotes the set of negative samples consisting of

mashups and randomly sampled services without invocation record, and r_{ms} is labeled as 0. $\hat{r}_{ms}$ is the predicted score of mashup m invoking s, and λ controls the strength of the L_2 regularization to prevent overfitting. Θ denotes all trainable model parameters, which consists of the initial node feature matrix $H^{(0)}$ and parameters $W^{(0)} \sim W^{(L-1)}$ in the GCN propagation layer. We use mini-batch Adaptive Moment Estimation (Adam) [33], a variant of stochastic gradient descent to optimize the model parameters. Concretely, for a randomly sampled batch of mashup-service pairs, in the forward pass, we calculate the node embeddings $h^{(1)}$ to $h^{(L)}$ through L steps of GCN propagation; in the backward pass, the model parameters are updated using the gradients with respect to the loss function $\mathcal{J}$. The whole training process is depicted as pseudo codes in Algorithm 1.

Algorithm 1 Training algorithm of MRGCN

Input: Constructed Adjacency matrices A^C, A^D, A^T
Output: Parameter set Θ

1: $L^C = D_C^{-\frac{1}{2}}(A^C + I)D_C^{-\frac{1}{2}}$;
2: $L^D = D_D^{-\frac{1}{2}}(A^D + I)D_D^{-\frac{1}{2}}$;
3: $L^T = D_T^{-\frac{1}{2}}(A^T + I)D_T^{-\frac{1}{2}}$;
4: $L_{fused} = fusion(L^C, L^D, L^T)$;
5: $R^+ = MS$, $R^- = \emptyset$;
6: **for** each $m \in R^+$ **do**
7: Uniformly sample t uninvoked services $R_m^- = \{(m, s_1^-), \ldots, (m, s_t^-)\}$;
8: $R^- = R^- \cup R_m^-$;
9: **end for**
10: $R = R^+ \cup R^-$;
11: Initialize $H^{(0)}, W^{(0)} \sim W^{(L-1)}$ with Gaussian distribution $\mathcal{N}(0, 0.01)$;
12: **for** $epoch = 1, \ldots, Epochs$ **do**
13: shuffle R randomly and partition R into $R_1, \ldots, R_{bs}$;
14: **for** $b = 1, \ldots, bs$ **do**
15: **for** $l = 1, \ldots, L$ **do**
16: $H^{(l+1)} = \text{ReLU}(L_{fused} H^{(l)} W^{(l)})$;
17: **end for**
18: $\mathcal{J} = 0$;
19: **for** each $(m, s) \in R_b$ **do**
20: $h_m = h_m^{(0)} \parallel \cdots \parallel h_m^{(L)}$;
21: $h_s = h_s^{(0)} \parallel \cdots \parallel h_s^{(L)}$;
22: $\hat{r}_{ms} = \sigma(h_m^T \cdot h_s)$;
23: $\mathcal{J} = \mathcal{J} + (-r_{ms} \log \hat{r}_{ms} - (1 - r_{ms}) \log(1 - \hat{r}_{ms}))$;
24: **end for**
25: $\mathcal{J} = \mathcal{J} + \lambda ||\Theta||_2^2$;
26: Update model parameters with Adam;
27: **end for**
28: **end for**

Lines 1–4 show the graph fusion process, and different fusion strategies can be used in line 4. Lines 5–10 prepare the training dataset including positive and negative samples. Lines 11–28 show the training process for GCN. In each epoch, the training dataset is divided into several batches, and in each batch, the GCN propagation step is performed and the parameters are updated according to the loss function $\mathcal{J}$.

Once the model is trained, for each mashup m that has no composition record, we can calculate the probability score $\hat{r}_{ms}$ for each candidate service s, which represents the probability of s being composed by m. Finally, all scores are sorted and the top-K services are recommended for the development of mashup m.

We further analyze the computational complexity of our approach. In the graph construction process, the construction of graph A^C costs $O(|MS|)$, A^D costs $O(N^2)$, and A^T costs $O(N^2)$. The graph fusion process has the time complexity of $O(|E|)$, where E are the edges in the multi-relational graph. In the graph convolution layer, the time complexity is $O(\sum_{l=1}^{L}|E|d_l d_{l-1})$, where d_l and d_{l-1} are the size of matrix $W^{(l)}$ of the l-th GCN layer. In the prediction layer, the process has the time complexity of $O(\sum_{l=1}^{L} d_l)$. As the number of layers L and the matrix dimension d_l are constants, the cost of the graph fusion, propagation, and prediction process is approximately linear to the number of edges including all relations in the constructed multi-relational graph.

4. Experiments

In this section, we conduct a series of experiments to evaluate our model. In particular, we aim to answer the following research questions:

- How does the proposed MRGCN model for service recommendation perform compared to the state-of-the-art methods?
- Does incorporating the functional relation and tagging relation into the model improve the performance of the model?
- Which graph fusion method has better performance in service recommendation?

All experiments were developed in Python and carried out on a personal PC with Intel Core i7 CPU with 2.5 GHz and 16 GB RAM, running the macOS High Sierra.

4.1. Dataset Description

The dataset is crawled from ProgrammableWeb (PW), the largest online web service and mashup repository from June 2005 to October 2020, including all mashups and services, the composition relation, textual descriptions, and tags of mashups and services. We remove services and mashups that have no textual description. To facilitate the evaluation process, we construct a multi-relational graph with services that have been invoked by at least one mashup. In this case, 6300 mashups and 1609 services are used to construct the graph, and the number of composition relation is 21,474. We use the category information available for both mashups and services in the PW dataset and treat them as tags. We filter out tags that are only used once by the mashup or service, as they are unable to form tagging relation. Table 1 shows the detailed statistics of the dataset.

Table 1. Dataset statistics.

Statistics	Value
Number of mashups	6300
Number of services	21,474
Number of services composed by mashup	1609
Average number of services in mashup	2.07
Sparsity of mashup-service composition matrix	99.87%
Number of tags	312
Average number of tags in mashups and services	3.4
Number of mashup-service interaction	13,219

We divide the mashups into the training set and test set. In particular, we randomly select 20% of mashups and the corresponding composition records are used as the test set, and the remaining 80% as the training set. We assume that the textual descriptions and tags of the mashups in the test set are used as the developers' requests for building new mashups. We use the 5-fold cross-validation technique, where in each round, one fold is used for testing and the others for training. The results of the five rounds are averaged as the final result.

4.2. Evaluation Metrics

In our work, developers write and submit their text requirements for mashup creation, and the recommender system (RS) returns top-N most relevant services. Therefore, it can be regarded as an information retrieval (IR) problem. We adopt two widely used metrics in IR and RS to evaluate the recommendation accuracy: Recall@K and NDCG@K [18,34]. Intuitively, Recall measures whether the component service is in the recommendation list, and NDCG gives more weight to correct predictions at the start of the recommendation list and discounts correct predictions farther from the beginning of the list.

Recall@K is the ratio of the number of actual component services in the top-K recommendation list to the number of services composed by the mashup. It is defined as:

$$\text{Recall@K} = \frac{1}{|MT|} \sum_{m \in MT} \frac{|\text{rec}(m) \cap \text{truth}(m)|}{|\text{truth}(m)|}, \tag{5}$$

where MT is the set of mashups in the test set, $\text{rec}(m)$ is the recommendation list of services of size K for mashup m. $\text{truth}(m)$ is the ground truth list of services that are composed by the mashup m in the test set.

Normalized Discounted Cumulative Gain (NDCG) [35] considers the ranking order of the recommended services, and assigns different weights to each service in the top-K recommendation list. it is defined as:

$$\text{NDCG@K} = \frac{1}{|MT|} \sum_{m \in MT} \frac{\sum_{i=1}^{K} \frac{2^{I(i)}-1}{\log_2(i+1)}}{\text{IDCG@K}}, \tag{6}$$

where $I(i)$ indicates whether the service at position i of the ranking list is in $\text{truth}(m)$. IDCG@K is the ideal DCG score of the top K services that can be achieved.

4.3. Baseline Methods

To evaluate the effectiveness of our approach, we compare with seven baseline service recommendation approaches that are designed to recommend services for new mashup development.

- CF [36]. This is the basic collaborative filtering technique to recommend services by identifying similar mashups. Given a new mashup m, we first calculate its textual description similarity to other existing mashups using the doc2vec model and cosine similarity measure, and select $N(m)$ top-similar mashups of m. The recommendation score between m and a candidate service s is calculated as:

$$score(m, s) = \frac{\sum_{m' \in N(m)} sim(m, m') \times I(m', s)}{\sum_{m' \in N(m)} sim(m, m')}, \tag{7}$$

  where $sim(m, m')$ is the cosine similarity between m and m', $I(m', s) = 1$ if m' composes s and 0 otherwise.
- CMF [6]. This approach builds multi-dimensional relationships among mashups, services and topics with a set of coupled matrices, and performs a coupled matrices factorization algorithm to predict unobserved relationships.
- LDA [29]. It learns the representations of textual descriptions of mashups and services with the LDA model and calculates the cosine similarity between mashup request and service contents.
- Doc2vec [30]. It learns the representations of textual descriptions of mashups and services with the doc2vec model and calculates the cosine similarity between mashup request and service contents.
- PaSRec [7]. It is a CF-based approach where it builds a heterogeneous information network (HIN) with mashups, services, content, tags, etc., defines meaningful meta-paths, and calculates the similarities between mashups using the meta-path-based

similarity measurement. It uses a Bayesian personalized ranking (BPR) algorithm to learn the weights of meta-paths.

- DHSR [18]. It combines collaborative filtering and textual content with a multilayer perceptron to capture the complex invocation relations between mashups and services. It computes semantic similarities between contents with three kinds of feature extractors and incorporates several pre-trained word embeddings.
- MRGCN. It is the proposed model in the paper, in which we implement two variants for functional content modeling: MRGCN_LDA in which LDA is used for content feature representation, and MRGCN_D2V in which Doc2vec is used for content feature representation.

4.4. Implementation Details

The details of training and the settings of hyper-parameters are listed as follows:

- In learning the vector representations of textual descriptions of mashups and services, we aggregate the textual descriptions of all 6300 mashups and 21,474 services into one big corpus, and perform a series of pre-processing steps: (1) Tokenization, which splits sentences into words; (2) Stop-words removal, which remove the insignificant words and infrequent words that appear less than 5 times; (3) Lemmatization, which transform words to their root forms. All pre-processing is done using the *NLTK* library. After pre-processing, we feed the dataset to the LDA and doc2vec model in the *gensim* library. The number of topics NK in LDA model and the vector size ND in doc2vec model are both set to 64. The other parameters are set as the default value of the gensim API. In building the functional and tagging relations, τ_1 is set to 30, and τ_2 is set to 15.
- For each positive mashup-service pair, we sample $t = 3$ negative pairs. In training the GCN model, the initial node features are set as the document representations learned with the doc2vec model, and the parameters in GCN are initialized from a Gaussian distribution $\mathcal{N}(0, 0.01)$. The batch size is set to 256, the learning rate is set to 0.005, and the L_2 regularization term λ is set to 10^{-4}. The number of GCN layers L is set to 2, and in each layer, the feature dimension d_l is set to 64. We stop the training process when NDCG@20 on the test set does not increase for 10 successive epochs.

4.5. Experimental Results

Figure 2 presents the recommendation accuracy of different approaches on the two metrics. As the results show, our approach exhibits improvements over all competing methods for different values of K. In particular, we have the following observations:

- LDA and Doc2vec methods only exploit textual similarity to match mashup requirements and candidate services, and ignore the existing mashup-service composition patterns. Therefore, they perform worse than the other methods that utilize both content similarity and composition relation. Moreover, Doc2vec performs better than LDA, indicating that learning the semantics of content with a distributed representation is better than the traditional bag-of-words model.
- CF and CMF methods achieve better performance than LDA and Doc2vec, due to the simultaneous use of mashup-service composition history and textual relations between mashups and services. Moreover, CMF performs better than CF. The reason could be that CMF implicitly considers the semantic relations between mashups and services through the shared topics, while only the semantic similarities between mashups are considered in CF.
- PaSRec is a CF-based method that evaluates the similarities between mashups. Different from the plain CF model, it considers diverse kinds of relations (captured as meta-paths) between mashups including mashups with similar contents, tags as well as services, and combines them with a different set of weights learned from data. It unsurprisingly achieves a better performance than CF.

- DHSR is a state-of-the-art method for service recommendation. It uses a deep neural network to combine the mashup and service features learned from the CF component and content component. Its performance improvement could be attributed to the use of deep networks to characterize the complex relations between mashups and services.
- Finally, our method MRGCN outperforms all these models in both Recall and NDCG. For instance, the Recall@5, NDCG@5, Recall@10, and NDCG@10 of our model were higher than DHSR by 9.9%, 13.1%, 7.3%, and 9.5%, respectively. On the one hand, it effectively combines different kinds of relations between both mashups and services, as opposed to PaSRec that only considers the relations between mashups; on the other hand, it can propagate and aggregate higher-order features of mashups and services with GCN, which can effectively deal with the problem of data sparsity. Moreover, MRGCN_D2V outperforms MRGCN_LDA, which is consistent with the baseline models and shows that empirically doc2vec has an advantage in modeling service functionality.

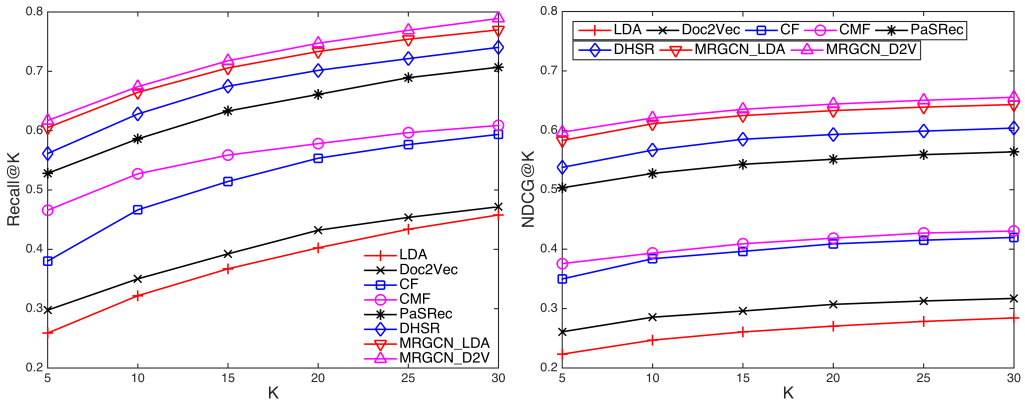

Figure 2. Performance comparison of different approaches. (**a**) Recall@K; (**b**) NDCG@K.

4.6. Discussion

In this section, we evaluate the performance for different parts of the MRGCN model and hyper-parameters. Specifically, Section 4.6.1 evaluates the impact of different relations in the multi-relational graph, Section 4.6.2 evaluates different fusion strategies, Section 4.6.3 evaluates the impact of the number of layers and layer Dimensions in GCN, and Section 4.6.4 evaluates the impact of thresholds τ_1 and τ_2.

4.6.1. Impact of Different Relations

The MRGCN combines three types of relations: composition relation, functional relation, and tagging relation. To demonstrate the necessity of the three relations in the model, we designed several variants of the model for comparison: MRGCN_D model only uses functional relation, the MRGCN_T model only uses tagging relation, the MRGCN_CD model uses both composition and functional relation, the MRGCN_CT model uses both composition and tagging relation, and the MRGCN_DT model uses both functional and tagging relations.

The results of the comparison among the variants of our approach are shown in Figure 3 when evaluated with K = 5, 10, 15, 20. We can observe that, when either relation is dropped from the original model, the performance becomes worse, which suggests that both three relations can indeed help improve the model performance. The more relations are incorporated, the better the model performance we can obtain. Furthermore, MRGCN_CT outperforms MRGCN_CD which in turn outperforms MRGCN_DT, which shows that the composition relation is more important than the tagging relation and in turn more important than the functional relation for achieving a better recommendation

performance. However, when only a single relation is considered, MRGCN_D has a better performance over MRGCN_T. We conjecture the underlying reason is that some services do not have annotated tags, and they become "cold-start" services that have no connections with other nodes in the graph when only tagging information is used. MRGCN achieves the best performance as it jointly considers the three complementary relations, thereby enriching the information from different sources and mitigating the cold-start effect in the recommendation.

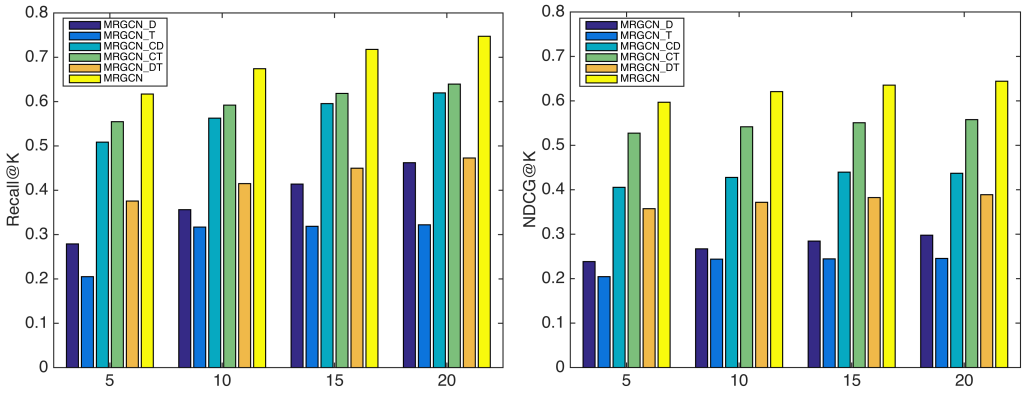

Figure 3. Performance comparison of different variants of MRGCN. (**a**) Recall@K; (**b**) NDCG@K.

4.6.2. Impact of Different Fusion Strategies

Three strategies for relation fusion have been proposed: add, max-pooling, and weighted-sum. We also evaluate the case when GCN is performed separately on three graphs with one single type of relation, and concatenate the final layer of the three graphs (denoted as concat). The results of the compared models are shown in Figure 4. We can observe that: (1) Max-pooling strategy performs the best among them in all evaluated cases, although the performance gain is quite marginal; (2) Weight-sum strategy performs slightly better than the add strategy; (3) The concat strategy performs significantly worse than the other three approaches. It shows that it is better to learn a single representation on a unified graph rather than learning separate representations, which could add more noise and model complexity.

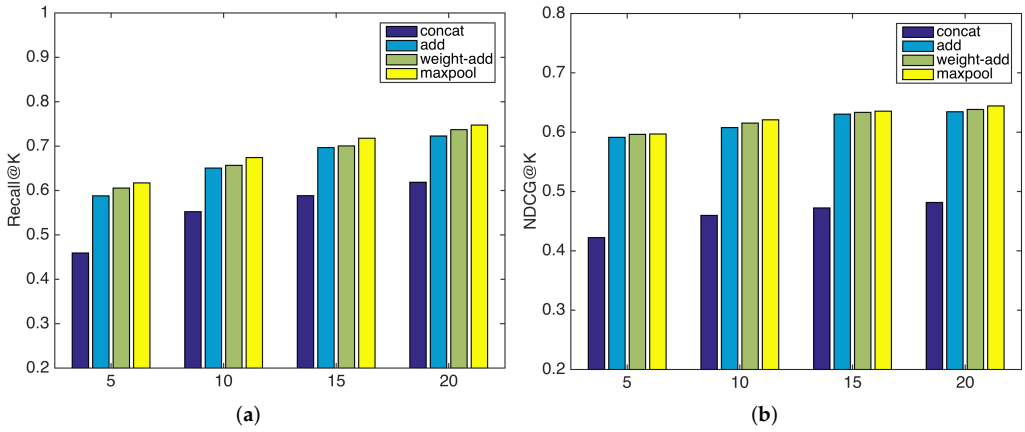

Figure 4. Impact of different fusion strategies. (**a**) Recall@K; (**b**) NDCG@K.

4.6.3. Impact of the Number of Layers and Layer Dimensions

In this section, we evaluate the effectiveness of introducing GCN for learning higher-order mashup and service features. We vary the number of layers L in the GCN from 1 to 3

and evaluate the performance. In addition, we evaluate the impact of feature dimension d_l on the performance, which varies in the set $\{8, 16, 32, 64, 128\}$. The results are shown in Figures 5 and 6. From Figure 5, we can see that the performance of our model increases from one layer to two layers because more layers can aggregate information from higher-order neighbors to enrich mashup and service features. When the number of layers is 3, the performance decreases, which indicates that too many layers can introduce noise and cause the overfitting problem. From Figure 6, we can see that the performance initially increases with the growth of the feature dimension, as a small dimension will probably constrain the model capacity. When the dimension is over 64, the performance decreases when we further increase the dimension, while the training time drastically increases. Therefore, a moderate feature dimension set as 64 is sufficient for model learning.

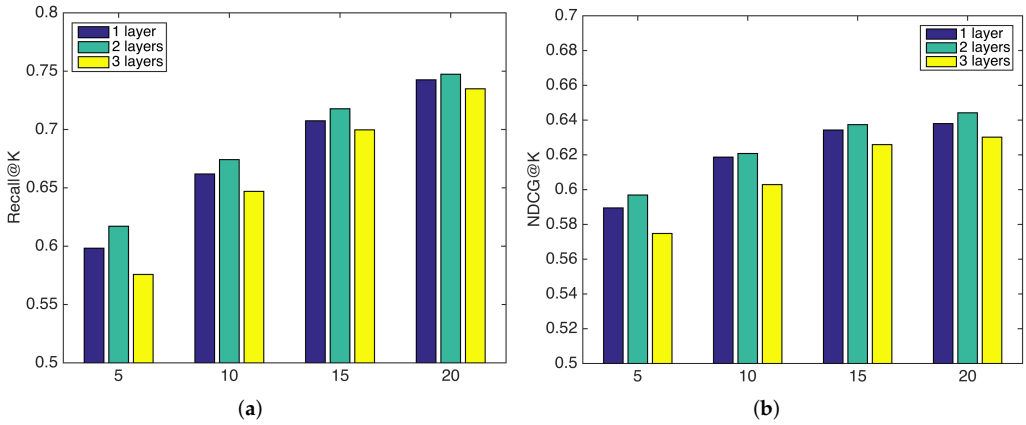

Figure 5. Impact of propagation layer number in GCN. (**a**) Recall@K; (**b**) NDCG@K.

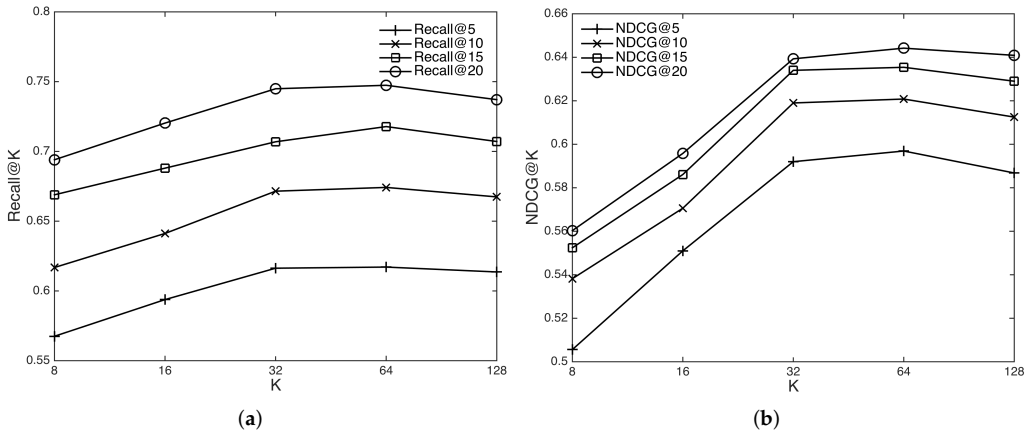

Figure 6. Impact of feature dimensions in GCN. (**a**) Recall@K; (**b**) NDCG@K.

4.6.4. Impact of the Threshold

When constructing graphs with functional relation and tagging relation, we set the number of edges for each node to the threshold τ_1 and τ_2, respectively. To study the parameters' effect on the recommendation performance, we adjust τ_1 and τ_2 from 5 to 40 with a step size of 5. The results are shown in Figure 7. We can see that, for the functional relation, the performance increases when τ_1 increases from 5 to 30, which suggests that exploiting more functional relation of mashup and service brings more beneficial information for recommendation. However, when τ_1 further increases, the performance plateaus and starts to decrease. Perhaps too many neighbors for a mashup or service introduce noisy information and harm the GCN learning. Therefore, we set τ_1 as 30 in the experiment. A similar trend can be seen for τ_2 when constructing the tagging relation. We can see that

optimal value of τ_2 (15 in the experiment) is generally smaller than that of τ_1, which might indicate that the tagging relation is more susceptible to noisy links than functional relation.

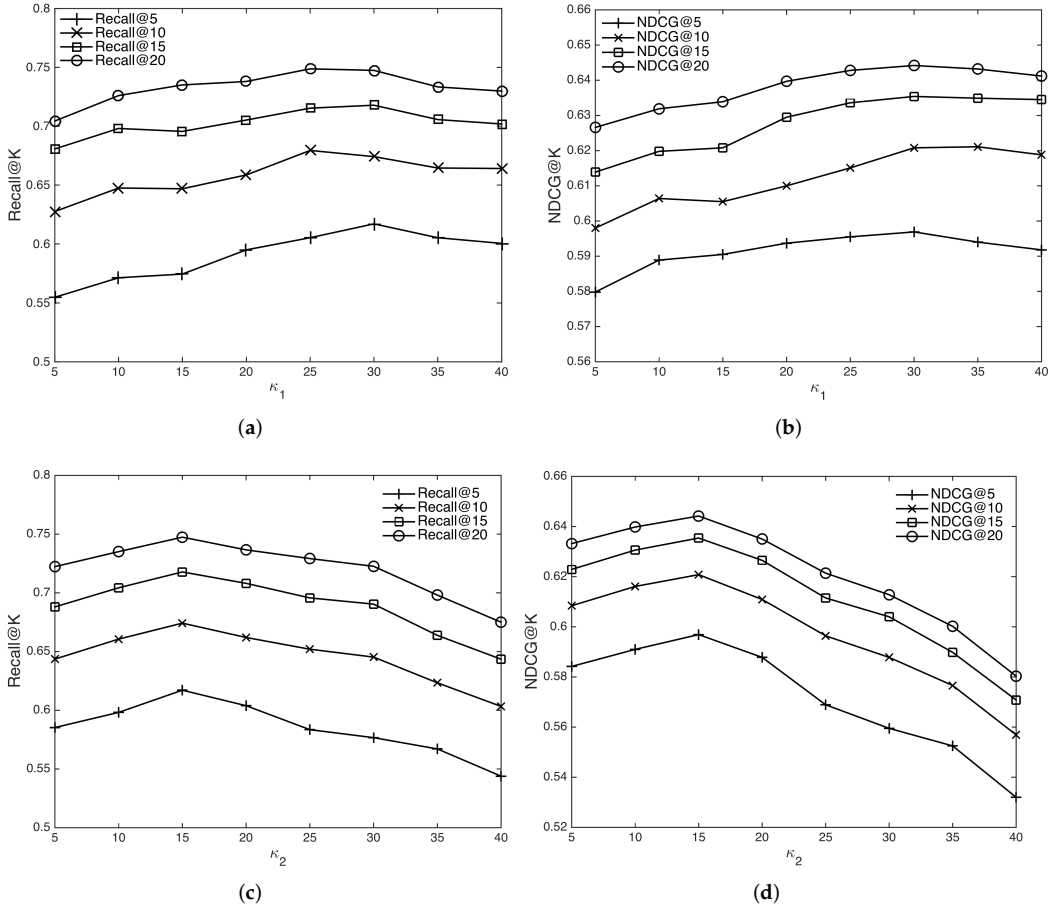

Figure 7. Impact of the threshold τ_1 and τ_2. (**a**) Recall@K with τ_1; (**b**) NDCG@K with τ_1; (**c**) Recall@K with τ_2; (**d**) NDCG@K with τ_2.

5. Conclusions

In this paper, we propose a multi-relational graph neural network model for recommending services (referred to as MRGCN) when only the mashup composition requirement is available. MRGCN addresses the three challenges faced in service recommendation: First, it incorporates functional relation and tagging relation between mashup and service into composition relation, and mashups and services become more densely connected. In addition, each mashup and service can incorporate features from high-order neighbors via graph convolution, which further alleviates the sparsity issue. Second, in the absence of historical composition record, mashup and service can rely on functional and tagging relations. All nodes in the multi-relational graph are connected, thereby obviating the cold-start issue. Third, several fusion strategies are designed into the graph convolution process, which is efficient and effective. The experiments on ProgrammableWeb demonstrate the effectiveness of our proposed approach. In the future, we plan to explore other possible relations that can further improve the recommendation performance, and extend the model framework to the inductive setting so that the model does not need to be retrained when new mashups or services are incorporated.

Author Contributions: Conceptualization, W.G.; methodology, W.G.; software, W.G.; validation, W.G.; formal analysis, W.G.; investigation, W.G.; resources, J.W.; data curation, W.G.; writing—original draft preparation, W.G.; writing—review and editing, J.W.; visualization, W.G.; supervision,

J.W.; project administration, J.W.; funding acquisition, J.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially supported by the National Key R&D Program of China under Grant No. 2019YFB1404802, the National Natural Science Foundation of China under Grant Nos. 62176231 and 62106218, the Zhejiang public welfare technology research project under Grant No. LGF20F020013, and Wenzhou Bureau of Science and Technology of China under Grant No. Y2020082.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Bouguettaya, A.; Singh, M.P.; Huhns, M.N.; Sheng, Q.Z.; Dong, H.; Yu, Q.; Neiat, A.G.; Mistry, S.; Benatallah, B.; Medjahed, B.; et al. A service computing manifesto: The next 10 years. *Commun. ACM* **2017**, *60*, 64–72. [CrossRef]
2. Tan, W.; Fan, Y.; Ghoneim, A.; Hossain, M.A.; Dustdar, S. From the Service-Oriented Architecture to the Web API Economy. *IEEE Internet Comput.* **2016**, *20*, 64–68. [CrossRef]
3. Im, J.; Kim, S.H.; Kim, D. IoT Mashup as a Service: Cloud-Based Mashup Service for the Internet of Things. In Proceedings of the 2013 IEEE International Conference on Services Computing, Santa Clara, CA, USA, 28 June–3 July 2013; pp. 462–469. [CrossRef]
4. Yu, J.; Benatallah, B.; Casati, F.; Daniel, F. Understanding Mashup Development. *IEEE Internet Comput.* **2008**, *12*, 44–52. [CrossRef]
5. Bobadilla, J.; Ortega, F.; Hernando, A.; Gutiérrez, A. Recommender systems survey. *Knowl. Based Syst.* **2013**, *46*, 109–132. [CrossRef]
6. Xu, W.; Cao, J.; Hu, L.; Wang, J.; Li, M. A Social-Aware Service Recommendation Approach for Mashup Creation. In Proceedings of the 2013 IEEE 20th International Conference on Web Services, Santa Clara, CA, USA, 28 June–3 July 2013; pp. 107–114. [CrossRef]
7. Liang, T.; Chen, L.; Wu, J.; Dong, H.; Bouguettaya, A. Meta-Path Based Service Recommendation in Heterogeneous Information Networks. In Proceedings of the Service-Oriented Computing—14th International Conference (ICSOC 2016), Banff, AB, Canada, 10–13 October 2016; Lecture Notes in Computer Science; Sheng, Q.Z., Stroulia, E., Tata, S., Bhiri, S., Eds.; Springer: Berlin/Heidelberg, Germany, 2016; Volume 9936, pp. 371–386. [CrossRef]
8. Xie, F.; Wang, J.; Xiong, R.; Zhang, N.; Ma, Y.; He, K. An integrated service recommendation approach for service-based system development. *Expert Syst. Appl.* **2019**, *123*, 178–194. [CrossRef]
9. Xie, F.; Chen, L.; Lin, D.; Zheng, Z.; Lin, X. Personalized Service Recommendation With Mashup Group Preference in Heterogeneous Information Network. *IEEE Access* **2019**, *7*, 16155–16167. [CrossRef]
10. Xie, F.; Chen, L.; Ye, Y.; Zheng, Z.; Lin, X. Factorization Machine Based Service Recommendation on Heterogeneous Information Networks. In Proceedings of the 2018 IEEE International Conference on Web Services (ICWS 2018), San Francisco, CA, USA, 2–7 July 2018; pp. 115–122. [CrossRef]
11. Meng, S.; Dou, W.; Zhang, X.; Chen, J. KASR: A Keyword-Aware Service Recommendation Method on MapReduce for Big Data Applications. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *25*, 3221–3231. [CrossRef]
12. Aznag, M.; Quafafou, M.; Jarir, Z. Leveraging Formal Concept Analysis with Topic Correlation for Service Clustering and Discovery. In Proceedings of the 2014 IEEE International Conference on Web Services (ICWS 2014), Anchorage, AK, USA, 27 June–2 July 2014; pp. 153–160. [CrossRef]
13. Bai, B.; Fan, Y.; Tan, W.; Zhang, J. DLTSR: A Deep Learning Framework for Recommendations of Long-Tail Web Services. *IEEE Trans. Serv. Comput.* **2020**, *13*, 73–85. [CrossRef]
14. Shi, M.; Tang, Y.; Liu, J. Functional and Contextual Attention-Based LSTM for Service Recommendation in Mashup Creation. *IEEE Trans. Parallel Distrib. Syst.* **2019**, *30*, 1077–1090. [CrossRef]
15. Yao, L.; Sheng, Q.Z.; Segev, A.; Yu, J. Recommending Web Services via Combining Collaborative Filtering with Content-Based Features. In Proceedings of the 2013 IEEE 20th International Conference on Web Services, Santa Clara, CA, USA, 28 June–3 July 2013; pp. 42–49. [CrossRef]
16. Yao, L.; Sheng, Q.Z.; Ngu, A.H.H.; Yu, J.; Segev, A. Unified Collaborative and Content-Based Web Service Recommendation. *IEEE Trans. Serv. Comput.* **2015**, *8*, 453–466. [CrossRef]
17. Jain, A.; Liu, X.; Yu, Q. Aggregating Functionality, Use History, and Popularity of APIs to Recommend Mashup Creation. In Proceedings of the Service-Oriented Computing—13th International Conference (ICSOC 2015), Goa, India, 16–19 November 2015; Lecture Notes in Computer Science; Barros, A., Grigori, D., Narendra, N.C., Dam, H.K., Eds.; Springer: Berlin/Heidelberg, Germany, 2015; Volume 9435, pp. 188–202. [CrossRef]

18. Xiong, R.; Wang, J.; Zhang, N.; Ma, Y. Deep hybrid collaborative filtering for Web service recommendation. *Expert Syst. Appl.* **2018**, *110*, 191–205. [CrossRef]
19. Ma, Y.; Geng, X.; Wang, J. A Deep Neural Network With Multiplex Interactions for Cold-Start Service Recommendation. *IEEE Trans. Eng. Manag.* **2021**, *68*, 105–119. [CrossRef]
20. Gao, W.; Chen, L.; Wu, J.; Gao, H. Manifold-Learning Based API Recommendation for Mashup Creation. In Proceedings of the 2015 IEEE International Conference on Web Services (ICWS 2015), New York, NY, USA, 27 June–2 July 2015; pp. 432–439. [CrossRef]
21. Gao, W.; Chen, L.; Wu, J.; Bouguettaya, A. Joint Modeling Users, Services, Mashups, and Topics for Service Recommendation. In Proceedings of the IEEE International Conference on Web Services, ICWS 2016, San Francisco, CA, USA, 27 June–2 July 2016; Reiff-Marganiec, S., Ed.; pp. 260–267. [CrossRef]
22. Wei, X.; Zhao, W.; Bing, L.; Bo, H. Automating Mashup service recommendation via semantic and structural features. *Math. Probl. Eng.* **2020**, *2020*, 4960439. [CrossRef]
23. Wang, X.; Wu, H.; Hsu, C. Mashup-Oriented API Recommendation via Random Walk on Knowledge Graph. *IEEE Access* **2019**, *7*, 7651–7662. [CrossRef]
24. Wang, X.; Liu, X.; Liu, J.; Chen, X.; Wu, H. A novel knowledge graph embedding based API recommendation method for Mashup development. *World Wide Web* **2021**, *24*, 869–894. [CrossRef]
25. Dang, D.; Chen, C.; Li, H.; Yan, R.; Guo, Z.; Wang, X. Deep knowledge-aware framework for web service recommendation. *J. Supercomput.* **2021**, *77*, 14280–14304. [CrossRef]
26. Nguyen, M.; Yu, J.; Nguyen, T.; Han, Y. Attentional matrix factorization with context and co-invocation for service recommendation. *Expert Syst. Appl.* **2021**, *186*, 115698. [CrossRef]
27. Zhang, Y.; Yang, H.; Kuang, L. A Web API Recommendation Method with Composition Relationship Based on GCN. In Proceedings of the IEEE International Conference on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom 2020), Exeter, UK, 17–19 December 2020; pp. 601–608. [CrossRef]
28. Lian, S.; Tang, M. API recommendation for Mashup creation based on neural graph collaborative filtering. *Connect. Sci.* **2021**, 1–15. [CrossRef]
29. Blei, D.M.; Ng, A.Y.; Jordan, M.I. Latent Dirichlet Allocation. *J. Mach. Learn. Res.* **2003**, *3*, 993–1022.
30. Le, Q.V.; Mikolov, T. Distributed Representations of Sentences and Documents. In Proceedings of the 31th International Conference on Machine Learning (ICML 2014), Beijing, China, 21–26 June 2014; Volume 32, pp. 1188–1196.
31. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In Proceedings of the 5th International Conference on Learning Representations (ICLR 2017), Toulon, France, 24–26 April 2017.
32. Chen, L.; Xie, Y.; Zheng, Z.; Zheng, H.; Xie, J. Friend Recommendation Based on Multi-Social Graph Convolutional Network. *IEEE Access* **2020**, *8*, 43618–43629. [CrossRef]
33. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), San Diego, CA, USA, 7–9 May 2015.
34. Manning, C.D.; Raghavan, P.; Schütze, H. *Introduction to Information Retrieval*; Cambridge University Press: Cambridge, UK, 2008; Volume 39.
35. Järvelin, K.; Kekäläinen, J. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.* **2002**, *20*, 422–446. [CrossRef]
36. Su, X.; Khoshgoftaar, T.M. A Survey of Collaborative Filtering Techniques. *Adv. Artif. Intell.* **2009**, *2009*, 421425. [CrossRef]