Service Recommendation for Composition Creation based on Collaborative Attention Convolutional Network

Ruyu Yan[1, 2], Yushun Fan[1, 2*], Jia Zhang[3], Junqi Zhang[1, 2], Haozhe Lin[1, 2]
1. Department of Automation, Tsinghua University, Beijing 100084, China
2. Beijing National Research Center for Information Science and Technology (BNRist)
3. Department of Computer Science, Southern Methodist University, Dallas, TX 75205, USA
yanry18@mails.tsinghua.edu.cn, fanyus@tsinghua.edu.cn, jiazhang@smu.edu,
jq-zhang15@tsinghua.org.cn, linhz@mail.tsinghua.edu.cn

***Abstract*—Service recommendation for composition creation is a widely applied technique, which expedites mashup development by reusing existing services. The core of service recommendations is to simultaneously understand user needs as well as the functions of available services. However, the descriptions provided by users and service providers may not always be accurate or up to date, which poses significant challenges to composition creating. To tackle this problem, in this paper we propose a deep learning-based service recommendation framework named coACN, short for Collaborative Attention Convolutional Network, which can effectively learn the bilateral information toward service recommendation. On the one hand, a domain-level attention module is constructed to refine user needs embeddings by drawing messages from related service domains. On the other hand, a graph convolutional network is established to excavate the service-composition graph and fuse structured information into service embeddings. For a service node in the graph, the information of its compositions as its first-order neighbor nodes is used to supplement the latest functions and features of the service; and the information of the services as its second-order neighbor nodes may bring collaborative relationships into the service. Extensive experiments on the real-world ProgrammableWeb dataset show the significant improvement of our proposed coACN framework over state-of-the-art methods.**

***Index Terms*—Service Composition, Graph Convolution Neural Network, Attention Mechanism, Service Recommendation, Collaborative Filtering**

I. Introduction

Service recommendation for composition creation is a widely recognized and applied technique to best meet the needs of users, by integrating and reusing existing services [21] instead of recreating the wheel. Nowadays, hundreds of thousands of services are continuously published online. How to select appropriate ones from the mass of services to generate service compositions, so-called mashups or workflows, to expedite mashup development has remained a hot research topic in both academia and industry [1], [7], [17].

As illustrated in Fig 1, the service ecosystem contains a wealth of services and service compositions, together with the interactive relationships among them. Over time, services gradually form various **service domains** in the process of constant competition and collaboration. Each service domain accumulates services with similar functions. In the scenario of service composition creation, both user needs and the functions of services are typically given in form of description sentences. Nevertheless, the descriptions provided by service providers and users may be incomplete or inaccurate, which makes it difficult to learn their proper representations. Here we thoroughly introduce the problems of two descriptions:

Descriptions of User Needs: Users typically describe their needs in natural language, which serve as the basis for many existing recommendation methods. Due to the complexity of user needs, however, simply relying on such descriptions may not be sufficient to identify all aspects of user needs. Take the mashup *Vacationic* in the *ProgrammableWeb* platform as an example. It calls services from four service domains: travel, mapping, photos, and video. However, the description of the mashup says it *"provides an overview of travel highlights around the world using interactive maps with destinations such as cities, beaches, national parks, and cultural sites."* From this description, only mapping and travel domains can be extracted apparently, but not the other two domains of photos and video. Throughout this paper, we will use the phrase *user needs* to represent the descriptions of both new user queries (new mashup aims) and past mashups.

Descriptions of Services: The descriptions of services become fixed once the services are published by service providers on the Internet. However, the functional characteristics of services may change during the process of collaboration. Meanwhile, users may use services in a different way from the service providers' expectation. As a result, simply considering the descriptions of services may not be accurate.

Existing methods mostly learn the embeddings of both user needs (i.e., queries) and services, and then measure the similarity between them to make a decision of recommendation. For example, MF [9] projects user needs and services into a shared latent space to study their match-making. With the rapid advancement of deep learning in recent years, a number of methods based on deep learning have been developed to tackle the service recommendation problem. To improve the non-linear representation ability of the model, NCF [5] explores neural network architectures for collaborative filtering. Zheng et al. [22] present the DeepCoNN to exploit the feedback information existing in the sentences, by coupled deep neural networks to further excavate content information. To deepen

*Yushun Fan is the corresponding author.

978-1-6654-1681-8/21/$31.00 ©2021 IEEE
DOI 10.1109/ICWS53863.2021.00059

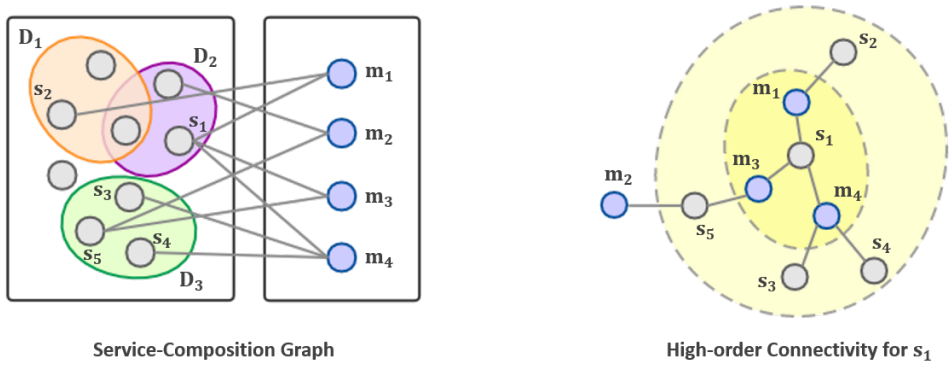

Fig. 1. *Left:* The diagram shows the service ecosystem which contains services and service compositions (mashups). The left-hand nodes represent services and the right-hand nodes represent compositions. The edges between them represent their historical interaction records. Services in the same circle carry similar functions and form corresponding service domains. *Right:* Take service node s_1 as an example, the high-order connectivity for s_1 is shown in a graph.

the use of graph structure with high-order neighbors, He et al. [4] recently propose LightGCN to mine the higher-level implicit information and achieves state-of-the-art performance. However, few of existing approaches consider the fact that the descriptions provided by service providers and users may be incomplete or inaccurate, which makes it difficult to learn their proper representations. In this work, we propose a novel method named Collaborative Attention Convolutional Network(coACN) to exhaustively understand the bilateral information for better service recommendations.

Because user needs aim to be fulfilled by services from service domains, we decide to synergistically integrate service domain information into the descriptions of user needs. To achieve this goal, we learn the domain similarity with user needs and assign a reasonable weight to add the service domain embeddings to user needs embeddings. Inspired by the attention mechanism [15], we have designed a domain-level attention unit to learn the similarity between user needs and each service domain. According to the similarity, related service domain information is reasonably integrated into the embeddings of user needs to enrich and refine the descriptions of user needs.

To solve the problem on the other side of services, we consider using past user needs information and the past collaborative relationships among services to calculate service embeddings. Since mashup compositions can reflect the latest functions and features of services to a certain extent, the mashup descriptions (i.e., descriptions of past user needs) can be a supplement to services. We thus construct a service-composition graph and use the graph convolutional network to mine and aggregate implicit information. As illustrated in Fig 1, the first-order neighbors of a service are the mashup compositions invoking it. The compositions' embeddings can be transferred to the service to complement the latest functions of the service. Meanwhile, the second-order neighbors of the service are other services invoked in the same composition. Through a high-order graph convolutional network, the collaborative relationships among services can be integrated into the service embeddings. Combining these two components, we devise a service recommendation framework for composition creation, which revises descriptions on both sides of the recommendation problem.

We have conducted extensive experiments on the real-world dataset ProgrammableWeb to demonstrate our proposed method, and the overall HR@20 of our coACN is 6.9% higher compared with the state-of-the-art methods.

The main contributions of this paper can be summarized as three-fold:

- We propose coACN, a service recommendation framework for composition creation to exhaustively understand the information on both sides of service recommendation.
- We design a domain-level attention unit to wove domain information into the descriptions of user needs to help identify potential services.
- We construct a graph network together with a graph convolutional network to capture holistic information to enrich and refine service descriptions.

The remainder of this paper is organized as follows. Section II prepares audience with preliminary information. Section III introduces our proposed coACN framework in detail. Section IV describes the experiments. Section V compares our research with related work. Section VI draws conclusions.

II. Preliminaries

In this section, we will introduce the definitions and problem of service recommendation for mashup creation, as well as the symbols to be used in the algorithms.

Definition 1: Service ecosystem. A service ecosystem refers to a complex ecosystem composed of massive services and compositions (mashups). It is represented by a four-tuple $SE = \{M, D^m, S, D^s\}$, in which M refers to mashups and S refers to services, the total numbers of mashups and services are I and J, respectively. The description of mashup $m_i \in M$ is expressed as $D_i^m = \{w_{i1}^m, w_{i2}^m, \ldots, w_{in}^m\}$, where w_{ik}^m is the k-th token and the description content consists of n words.

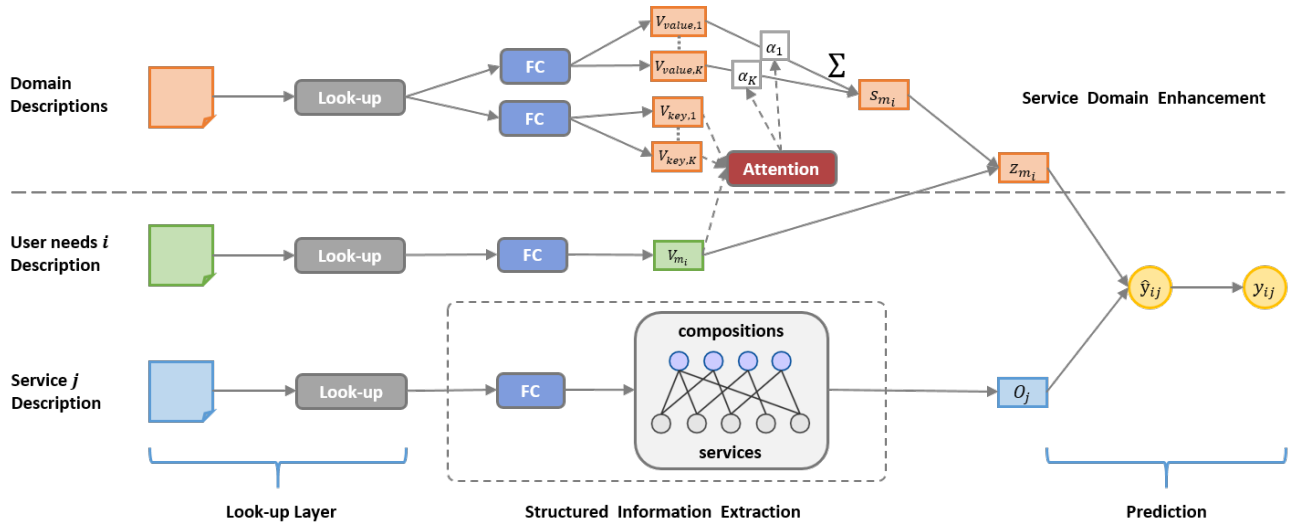

Fig. 2. The structure of coACN, in the context of a prediction process of mashup i and service j. There are four components in total. Among them, *Structured Information Extraction* and *Service Domain Enhancement* are core parts. *Structured Information Extraction* uses a graph convolution network to excavate the service-composition graph to extract structured information and learn proper embeddings on the side of services. *Service Domain Enhancement* uses a domain-level attention unit to enhance domain information on the side of user needs.

Similarly, $D_j^s = \{w_{j1}^s, w_{j2}^s, \ldots, w_{jn}^s\}$ is the description of service $s_j \in S$ comprising n words. It is worth noting that the descriptions of mashups and services are unified into vectors with a fixed length n. In the specific operation, if the length of the mashup description is longer than n, we will truncate it. If the length of the mashup description is shorter than n, zero (0) will be used to complete the vector.

Definition 2: Service domain. Services bearing similar functionalities form a service domain, i.e., service category [17]. The description of a service domain is generated by splicing and integrating the descriptions of the services belonging to the domain. Suppose there are K_s service domains in the service ecosystem, the description of service domain k is represented as $D_k^d = \{w_{k1}^d, w_{k2}^d, \ldots, w_{kn_d}^d\}$, where w_{kl}^d is the l-th token. Similarly, the descriptions of service domains (that is, the descriptions of categories) are processed into vectors with a fixed length of n_d.

Definition 3: Service-composition graph. The historical interaction relationships between services and compositions are constructed to become a service-composition graph. The matrix of the graph is represented by $Y \in \mathbb{R}^{I \times J}$, recording the interaction network of services and mashups. If service s_j is invoked by mashup m_i, $y_{ij} = 1$; otherwise, $y_{ij} = 0$.

Problem: Service recommendation for composition creation: Given the definitions described in section II.A, a new mashup query Q carries query description $D^q = \{w_1^q, w_2^q, \ldots, w_n^q\}$, where w_i^q is the i-th token in D^q. The goal is to recommend services to fulfill part or whole of query Q. The recommended result is given as a ranked list RL, in which a service with a higher rank has a higher probability to be adopted by query Q.

III. Framework coACN

To better perceive user needs and understand service functional characteristics, we devise a new method named coACN. In this section, we firstly provide an overview of coACN, and then detail each of its comprising components separately.

A. Overview of coACN

The overall structure of coACN is illustrated in Fig 2. Its structure consists of the following four components:

- **Look-up Layer:** This layer transforms the input token vectors of descriptions into embeddings with semantic information through a look-up table.
- **Service Domain Enhancement:** A domain-level attention unit is established to enhance service domain information in the descriptions of user needs.
- **Structured Information Extraction:** A graph convolution network is built to extract the structured information carried in service-composition graph, to better understand functional descriptions of services.
- **Prediction Layer:** An inner-product model is utilized to predict the possibility for a mashup query to invoke a service.

Among the four components, two components *Service Domain Enhancement* and *Structured Information Extraction* are essential, which are designed to perceive user needs and to understand service functional characteristics.

B. Look-up Layer

To exploit the semantics of descriptions, we adopt an approach that is widely used in many Natural Language Processing (NLP) applications [3], [22], which inquires each word embedding representation from a look-up table. As the same process is applied to mashup descriptions, service descriptions, and service domain descriptions, we take mashup descriptions as an example. Given the mashup m_i's description $D_i^m = \{w_{i1}^m, w_{i2}^m, \ldots, w_{in}^m\}$, we project each word to its embedding representation: $E_{m_i} = [e_{i1}, e_{i2}, \ldots, e_{in}], e_{ik} \in \mathbb{R}^d$, where n is the length of the description and d is the word embedding dimension.

C. Service Domain Enhancement

The key objective of embedding-based service recommendation is to capture the relations between mashups and services in the latent space. Due to the domain complexity of user needs (mashup queries), simply relying on the descriptions of mashups is insufficient to identify all aspects of user needs. Therefore, we decide to take into account service domain information for the sake of better perceiving user needs. We first use several fully connected networks to increase the nonlinear abilities of our model, and then build a domain-level attention unit to enhance the service domain information in mashup queries.

Fully Connected Network. A fully connection network aims to transform mashup m_i's representation matrix E_{m_i} received by the look-up layer into a one-dimension vector $v_{m_i} \in \mathbb{R}^L$, where $L = n \times d$. Put v_{m_i} into a fully connected network unit:

$$V_{m_i} = \sigma(W_m^T v_{m_i} + b_m), \tag{1}$$

where $W_m \in \mathbb{R}^{dim \times L}$, $b_m \in \mathbb{R}^{dim}$ and σ are weight matrix, bias vector and activation function respectively; and dim represents the embedding dimension. In this paper, $ReLU$ is selected as the activation function.

In order to further enhance service domain information into mashup embeddings, we first transform service domain embeddings into key embeddings and value embeddings. Key embeddings are used to measure the similarity between mashups and service domains, and value embeddings are used to generate the final *additional mashup embedding* s_m, which will be introduced in a later section. We thus obtain the representations of value and key respectively:

$$V_{value,k} = \sigma(W_{value} V_{C_k} + b_{value}), \tag{2}$$

$$V_{key,k} = \sigma(W_{key} V_{C_k} + b_{key}), \tag{3}$$

where $W_{value}, W_{key} \in \mathbb{R}^{dim \times L_c}$ and $b_{value}, b_{key} \in \mathbb{R}^{dim}$ denote the weight matrices and bias vectors of fully connected networks, respectively. Here, $L_c = n_c \times d$. The outcome of this process are the preliminary results of mashup embeddings and service domain embeddings V_{m_i}, $V_{value,k}$, and $V_{key,k}$.

Domain-level Attention Unit. In order to encode the membership degree of mashups in service domains to promote the mashup representations, we fuse service domain embeddings into an *additional mashup embedding* s_m. To calculate s_m through a non-uniform coefficient, we adopt the following formula:

$$s_m = \sum_{k=0}^{K_s} \alpha_{mk} V_{value,k}. \tag{4}$$

We apply the attention mechanism, a popular mechanism widely used in many recommender approaches [2], [18], to model the various service domain importance in a mashup-sensitive fashion. Specifically, the domain-level attention unit learns a specific weight α_{mk} for every service domain embedding $V_{value,k}$:

$$\alpha_{mk} = \frac{\exp\left(V_{m_i}^T V_{key,k}\right)}{\sum_{j=0}^{K} \exp(V_{m_i}^T V_{key,j})}. \tag{5}$$

The inputs of the domain-level attention unit are mashup embeddings and service domain embeddings, toward making the learned attention score sensitive to the domain membership degree of mashup.

Afterwards, we exploit the two resulting mashup embeddings (mashup embedding V_{m_i} through the fully connected network and additional mashup embedding s_m), and fuse them into a unified embedding. It has been proven effective to fuse embedding vectors learned from multi-modal data to combine signals from the available recommendation models [2], [19]. The unified embedding via fusion can be given as follows:

$$z_{m_i} = (1 - \beta) s_m + \beta V_{m_i}, \tag{6}$$

where β is a hyper-parameter to control the proportion of service domain information. The aim of β is to balance the information from the service domain and mashup m_i itself.

D. Structured Information Extraction

To understand the functional characteristics of services accurately and completely, we take structured information into consideration. Be more specific, we construct a service-composition graph which is a bipartite graph composed of services and mashups, and build a graph convolution network to excavate it. Through the graph network, the content of user needs can be transferred from the compositions, which are the first-order neighbors of a service, to help improve service perception. Meanwhile, the collaborative relationships between services can be integrated by using the second-order graph convolution.

Fully Connected Network. Similar to the part in *Service Domain Enhancement*, we can adopt fully connection network to obtain the representation of a service description as follows:

$$V_{s_j} = \sigma(W_s^T v_{s_j} + b_s). \tag{7}$$

Graph Convolution. Our objective is to leverage the content of past user needs to improve the service embeddings and capture the collaborative relationships among services. We thus design a graph convolution network to excavate the service-composition graph. Inspired by LightGCN [4], we utilize light graph convolution and layer combination to achieve our goal.

Refer to the definitions given in section II, the matrix of a service-composition graph is denoted as $Y \in \mathbb{R}^{I \times J}$, where I and J indicate the numbers of mashups and services, respectively. We thus obtain its adjacency matrix as:

$$A = \begin{pmatrix} 0 & Y \\ Y^T & 0 \end{pmatrix}, \tag{8}$$

where $A \in \mathbb{R}^{(I+J)\times(I+J)}$. The embedding matrix of graph convolution at the 0-th layer is defined as $X^{(0)} = concat(V_{m_0}, \ldots, V_{m_I}, V_{s_0}, \ldots, V_{s_J}) \in \mathbb{R}^{(I+J)\times dim}$. V_{s_j} is service s_j embedding received by fully connected network, and V_{m_i} is mashup m_i embedding. The matrix equivalent form of light graph convolution can be written as:

$$X^{(k+1)} = (D^{-\frac{1}{2}} A D^{-\frac{1}{2}}) X^{(k)}, \tag{9}$$

where $D \in \mathbb{R}^{(I+J)\times(I+J)}$ refers to a diagonal matrix, in which each entry D_{ii} represents the number of non-zero entries in the i-th row vector of the adjacency matrix A (also named as degree matrix). Finally, we get the final service embedding matrix:

$$X = \sum_{l=0}^{L_s} a_l X^{(l)} = \sum_{i=0}^{L_s} a_l \hat{A}^l X^{(0)}, \tag{10}$$

where $a_k \geq 0$ indicates the importance of the k-th layer embeddings in the final embeddings. It can be regarded as a hyper-parameter that needs to be tuned manually. $\hat{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ is the symmetrically normalized matrix. We extract the last J rows from the result, which is the service part, and express them as $O \in \mathbb{R}^{J \times dim}$. The embedding of service s_j is represented as O_j.

E. Prediction Layer

After the operations conducted in the aforementioned layers, we are ready to predict whether a service may be invoked in a mashup query or not. We use z_{m_i} and O_j to estimate y_{ij}, which is the probability of mashup query m_i invokes service s_i. Since our work majorly focuses on the framework for service composition recommendation, we decide to apply a simple yet widely-used inner product model:

$$\hat{\mathrm{y}}_{ij}^t = \sigma\left(z_{m_i}^T O_j\right), \tag{11}$$

where σ is the sigmoid function. Note that the predictive function can be expanded to more complex models at will.

Through the above four components, we will finally obtain the desired recommendation results.

F. Optimization Strategy

For model training, we utilize the mashups created in the history as ground truths for parameter optimization. We employ the Bayesian Personalized Ranking (BPR) loss [11], which is a pairwise loss that prefers the prediction of invoked services to be higher than services not invoked. Let B be a batch of the compose of mashup, service included in this mashup and service not included. The loss function can be defined as follows:

$$\mathrm{L}_{BPR} = -\frac{1}{|B|} \sum_{(i,j,j')\in B} \ln \sigma\left(\hat{y}_{ij} - \hat{y}_{ij'}\right) + \lambda \|\Theta\|, \tag{12}$$

where λ is a hyper-parameter for regularization coefficient. Service s_j is included in mashup m_i and service $s_{j'}$ is not included. We adopt Adam as the optimizer to update the parameters.

IV. Experiments

In this section, we first introduce the dataset used to verify model effects, then present the evaluation metrics and baseline methods. Afterwards, we report the experimental results.

A. Experimental Settings

*1) **Dataset**:* We chose a dataset widely used in the field of service recommendation for composition creation, ProgrammableWeb, to verify our proposed method. This dataset is the world-largest portal of Web API service and mashup market place. Up to April 2021, it has accumulated 24,034 web APIs and 7,974 mashups (service compositions of Web APIs). The dataset provides the content information of services and mashups, including text descriptions, tags, and category information. The names of some popular services include Google maps, Twitter, Last.FM, Youtube, etc. In the process of model testing, the descriptions of the mashups in the test dataset are used as mashup queries, and the APIs invoked by those mashups are regarded as the correct results of service recommendation. We crawled the data of APIs and mashups from September 2005 to November 2020 from ProgrammableWeb.com. Information on the dataset is summarized in Table I. To evaluate the effect of compositions, the mashups we tested containing at least two services. At the same time, APIs that have never been used were removed when testing.

TABLE I
DATASET DETAILS

Item	Value
Number of mashups	7,974
Number of mashups invoking at least two services	3,194
Number of services	24,034
Number of services used in at least one mashup	1,154
Average number of services in one mashup	3.16
Sparsity of mashup-service matrix	0.274%

*2) **Evaluation Metrics**:* For the sake of simulating the service recommendation for composition creation in the real scene, in the evaluation, similar to work [1], the data before a certain time was selected as the training set, and the part of the data after that time was used as the test set. In our experiment, we set the test set size to 100, that is, each time 100 consecutive mashups were selected to form a test set. The performance of each algorithm from July 2010 to September 2020 was evaluated. To evaluate the performance of our recommendation technique, we adopted two widely accepted metrics, namely Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG). The former HR@K metric measures whether the test item is contained in the top-K recommendation ranking list. The latter NDCG@K metric extends HR by assigning higher scores to the hits at higher ranks. The adopted metrics are formulated as follows:

$$\begin{aligned} NDCG@K &= \frac{1}{R_N}\sum_{i=1}^{N}\frac{2^{rel_i-1}}{\log_2(1+i)}, \\ HR@K &= \frac{\sum_{i=1}^{K} rel_i}{|y_u^{test}|}, \end{aligned} \tag{13}$$

where K is the size of the recommendation list, $rel_i = 0$ or 1 denotes whether the service at the rank i is in the test set, and the R_N term indicates the maximum possible cumulative component through ideal ranking. $|y_u^{test}|$ is the number of services used by user u in the test set.

*3) **Baselines**:* To evaluate the performance of the Top-K service recommendation for composition creation, we compared our coACN with the following four baseline methods:

- **MF** [12]: This method optimizes the matrix factorization by the Bayesian personalized ranking (BPR) loss, which exploits the mashup-service invoking interactions for service recommendation.
- **NeurMF** [5]: The method is a state-of-the-art neural Collaborative Filtering (CF) model, which uses multiple hidden layers above the element-wise and concatenation of user and item embeddings to capture their nonlinear feature interactions. Especially, we employed two-layered plain architecture, where the dimensions of each hidden layer remain the same.
- **DeepCoNN** [22]: This method models mashups and services via excavating the descriptions by a convolutional neural network.
- **LightGCN** [4]: This method consists of light graph convolution and layer combination to complicate the design of GCNs, for collaborative filtering service recommendation.

*4) **Experiment Setup**:* We preprocessed the descriptions of both mashups and services, firstly removing stop words, and then truncating (or completing) the long (or short) descriptions at the same length of 80 words. During the training phase, we set the negative sampling proportion as 4 and the default mini-batch size of 256. The embedding size was set to 64 for both coACN and baselines. We used the model parameters trained by coACN-A and coACN-co (variants of our coACN for comparison) to initialize the corresponding model parameters in coACN, so that the model can learn in the right direction. We employed the Adam [8] optimizer and used the default learning rate of 0.0001, the default L2 regularization parameter of 0.0005. Same as in LightGCN [4], the layer combination coefficient a_k was uniformly set to $\frac{1}{1+K}$ where K is the number of layers. We tested K in the range of 1 to 4, and finally, chose 2. We investigated the top-K recommendation performance with K setting to $10, 20$ both in our method and baseline methods. As aforementioned, we tested the performance of all services that exist in systems. We implemented coACN based on Pytorch, a widely-used open-source machine learning framework.

TABLE II
PERFORMANCE COMPARISON

	HR@10	**NDCG@10**	**HR@20**	**NDCG@20**
MF	0.3721	0.4797	0.4686	0.5238
NCF	0.4129	0.5141	0.4878	0.5511
DeepCoNN	0.4023	0.5120	0.5096	0.5689
LightGCN	0.4396	0.5396	0.5006	0.5792
coACN-N	0.3996	0.5096	0.4806	0.5492
coACN-co	0.4481	0.5326	0.5050	0.5660
coACN-A	0.4627	0.5797	0.5328	0.6118
coACN	**0.4828**	**0.6050**	**0.5447**	**0.6368**

B. Performance Comparison

Table II summarizes the experimental results of our coACN and baseline methods on the ProgrammableWeb dataset. From the results, we can draw the following four conclusions:

First, increasing the nonlinearity and complexity of the model could improve the recommendation effect of service composition. As shown in Table II, NCF is better than MF in each metric. Among them, there is an 11% improvement in HR@10 but only 4.2% in HR@20. Through the analysis of the ProgrammableWeb dataset, we guess that since the record number of popular services is much larger than that of long-tail services, the popular services are more likely to be accurately recommended, ranking at the top of the recommendation sequence. The length of 20 is sufficient for finding popular services, but not enough for finding long-tail services. Even if the length of the prediction list is increased from 10 to 20, it is difficult to cover the long-tail services. Thus, the improvement of HR@10 is more obvious than that of HR@20.

Second, extracting structured information of graph can improve the effectiveness of service composition. In most indicators of Table II, LightGCN performs better than MF and NCF. LightGCN can transfer the high-order collaborative relationships of services through the service-composition graph network.

Third, enhancing the understanding of descriptive content information will improve the effectiveness of service composi-

tion recommendations. On the basis of using the trained look-up table to construct word embeddings, DeepCoNN adds a convolution network to capture more detailed text information. It can also be seen from Table II that this method shows a better service composition recommendation effect than MF and NCF in various metrics. At the same time, LightGCN and DeepCoNN have similar performance on HR@20 and NDCG@20, while LightGCN is better than DeepCoNN on HR@10 and NDCG@10. We guess that the method with a relational network could better recommend popular services.

Fourth, introducing domain information into service composition can significantly improve the recommendation effect of service composition. We notice that our coACN method has a significant improvement in HR and NDCG, compared with other baseline methods. In the HR@20 index, coACN is 6.9% higher than the best baseline method. In the NDCG@20 index, coACN is 9.9% higher than the best baseline method.

C. Effect of Components

After quantitative analysis, we examined the role of model components, the domain-level attention unit in *Service Domain Enhancement* and the service-composition graph in *Structured Information Extraction*. To be persuasive, we explored the effects of coACN and its three variants:

- coACN-N: a variant of coACN, which contains neither service-composition graph nor domain-level attention unit.
- coACN-co: a variant of coACN, which only contains service-composition graph, but not the domain-level attention unit.
- coACN-A: a variant of coACN, which only contains the domain-level attention unit, but not the service-composition graph.

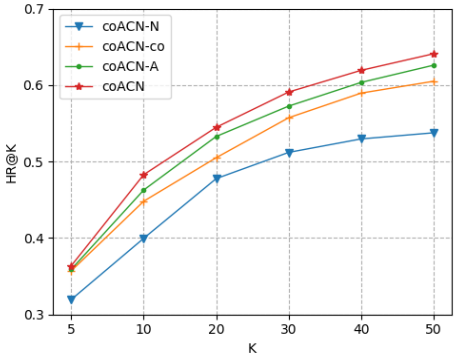

Fig. 3. Performance of variants of coACN

Table II and Figure 3 show the effect of the coACN and its three variants. From them, we can draw the following conclusions:

The coACN-N removes two key components and basically degenerates into the MF model with fully connected networks added, which is the worst compared with the other three models. Compared with the results in Table II, the coACN-N is slightly better than the MF but still worse than the NCF.

The coACN-co adds the service-composition graph on the basis of the coACN-N. By using a graph convolution network on this graph, both the latest functions of services and the collaborative relationships of services can be mined to supplement the service embeddings. It is 12% higher than the coACN-N in HR@10.

The coACN-A adds a domain-level attention unit on the basis of the coACN-N. Based on the description of the mashup query, the coACN-A enhances more domain information according to the domain similarity of mashup. Compared with the coACN-N, it has been improved greatly.

D. Impact of Hyper-parameters

We further investigated the impact of four hyper-parameters in the coACN thoroughly. When we studied the impact of one hyper-parameter, the other three were set with the default values. Fig 4 shows the effects of the four hyper-parameters used in our coACN.

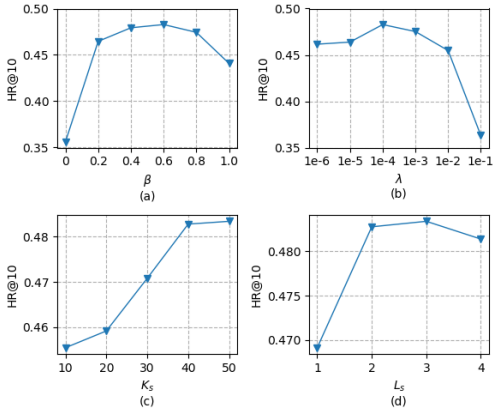

Fig. 4. Influence of four hyper-parameters on the coACN effect.

1) β: As a hyper-parameter controlling the proportion of service domain information learned from the domain-level attention unit to mashup representation, β value affects the effect of our coACN. We set $\beta = \{0, 0.2, 0.4, 0.6, 0.8, 1\}$ in turn to observe the corresponding service composition recommendation effect. As shown in Fig 4, when $\beta = 0$, which means domain information is not included, the effect is basically the same as coACN-co. When $\beta = 0.6$, the best effect is achieved. When $\beta = 1$, the final representation of the mashup is completely dependent on the domain information, while its own description information is discarded, so the effect is poor.

2) λ: This parameter controls the proportion of $L2$ regularization in the optimization function of the model. According to experience, adding $L2$ regularization could avoid overfitting. In our experiment, we set $\lambda = \{1e^{-6}, 1e^{-5}, 1e^{-4}, 1e^{-3}, 1e^{-2}, 1e^{-1}\}$. It can be found that HR-index increases first and then decreases. The maximum value of HR is obtained at $\lambda = 0.0001$.

3) K_s: The number of service domains K_s determines how many categories to retain. An appropriate K_s can ensure the effect of the service composition recommendation. Fig 4(c) shows the effect of hyper-parameter K_s on the HR@10 of coACN recommendation. The results show that 40 is a reasonable choice of K_s. When K_s is less than 40, increasing K_s can significantly improve the recommendation effect. While when K_s is more than 40, the recommendation effect of coACN tends to be stable.

4) L_s: We also investigated the number of layers of graph convolution networks on the service side and found that the model has reached the upper limit of capacity when the layer number is 2. A deeper graph network only increased the training time, but did not improve the recommendation effect. Our analysis shows that the mashups in ProgrammableWeb dataset are mostly composed of a small number of services, which leads to the lack of high-order neighbors. In other words, service nodes have only first-order neighbors in the cooperative network. Considering other datasets with complex high-order relations in the future, we may consider increasing the number of layers to achieve better results.

V. Related work

In this paper, we propose a solution to solve the problem of service recommendation for composition creation. Nowadays, service recommendation plays a decisive role in relieving the information overload problem.

Some existing service recommendation approaches use semantic-based models and significantly improve recommendation performance. Qiu et al. [14] extract content-based features from both WSDL documents and user queries, then calculates the semantic similarity between them. Bai et al. [1] extend the CTR and proposes a framework that considers the procedure of service selection as a generative process.

Rating-based methods mainly focus on the interaction rating matrix of users and services. Collaborative filtering [12], [20] models user preferences on services based on the user-service historical rating matrix. MF [9] projects users and services into a shared latent space. With the wide application of deep learning in recommendation systems, many researchers provide a neural network-based recommendation. Zheng et al. [22] present the DeepCoNN which exploits the feedback information existing in the reviews by coupled deep neural networks. Another type of classic service recommendation is rating-based. NCF [5] explores neural network architectures for collaborative filtering. To deepen the use of graph structure with high-order neighbors, He et al. [4] recently propose LightGCN to mine high-level implicit information and achieves state-of-the-art performance.

Some methods also take the evolution characteristics of services into consideration [10], so as to better predict the tendency of services invocation and update service function information.

Apart from these function-related recommendation methods, some researches focus on non-functional properties of services. Strunk et al. [6], [13], [16] provide QoS-aware service recommendation methods which give recommendations by analyzing the characteristics of services (such as reliability, availability, response time, and so on).

In contrast to related work, our coACN effectively learns and enriches the bilateral information toward service recommendation, from both user needs and services.

VI. Conclusions

This paper has presented a novel deep learning-based service recommendation framework for composition creation, named coACN, short for Collaborative Attention Convolutional Network. This framework can exhaustively understand the two sides' information on service recommendations. One is user needs, and the other is service functions. By using a domain-level attention unit to learn the similarity between user needs and related service domains, the service domain information is reasonably integrated into the embeddings of user needs. By using a graph convolution network on the service-composition graph, the structured information can be imported into service embeddings. Extensive experiments on real-world dataset ProgrammableWeb have demonstrated the effectiveness of our proposed method. The overall HR@20 of coACN is 6.9% higher than the state-of-the-art methods.

We plan to continue our research in the following two aspects. First, we plan to integrated additional information to further enrich the descriptions of user needs and services, such as user profiles and service usage data. Second, we plan to consider datasets with complex high-order relations, to further study the effectiveness of our coACN framework.

Acknowledgment

This research has been partially supported by the National Key Research and Development Program of China (No.2018YFB1402500). Yushun Fan is the corresponding author.

References

[1] B. Bai, Y. Fan, K. Huang, W. Tan, B. Xia, and S. Chen. Service recommendation for mashup creation based on time-aware collaborative domain regression. In *2015 IEEE International Conference on Web Services*, pages 209–216. IEEE, 2015.

[2] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, and T.-S. Chua. Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 335–344, 2017.

[3] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537, 2011.

[4] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 639–648, 2020.

[5] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.

[6] Y. Hu, Q. Peng, and X. Hu. A time-aware and data sparsity tolerant approach for web service recommendation. In *2014 IEEE International Conference on Web Services*, pages 33–40. IEEE, 2014.

[7] K. Huang, Y. Fan, and W. Tan. Recommendation in an evolving service ecosystem based on network prediction. *IEEE Transactions on Automation Science and Engineering*, 11(3):906–920, 2014.

[8] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434, 2008.

[10] H. Lin, Y. Fan, J. Zhang, and B. Bai. Msp-rnn: Multi-step piecewise recurrent neural network for predicting the tendency of services invocation. *IEEE Transactions on Services Computing*, 2020.

[11] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*, 2012.

[12] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295, 2001.

[13] A. Strunk. Qos-aware service composition: A survey. In *2010 Eighth IEEE European Conference on Web Services*, pages 67–74. IEEE, 2010.

[14] Q. Tian, L. Lei, and P. Lin. Web service discovery with uddi based on semantic similarity of service properties. In *International Conference on Semantics*, 2007.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[16] Z. Wang and J. Crowcroft. Quality-of-service routing for supporting multimedia applications. *IEEE Journal on selected areas in communications*, 14(7):1228–1234, 1996.

[17] B. Xia, Y. Fan, C. Wu, K. Huang, W. Tan, J. Zhang, and B. Bai. Domain-aware service recommendation for service composition. In *2014 IEEE International Conference on Web Services*, pages 439–446. IEEE, 2014.

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

[19] H. Yan, C. Yang, D. Yu, Y. Li, D. Jin, and D. M. Chiu. Multi-site user behavior modeling and its application in video recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 33(1):180–193, 2019.

[20] H. Zhang, F. Shen, W. Liu, X. He, H. Luan, and T.-S. Chua. Discrete collaborative filtering. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 325–334, 2016.

[21] J. Zhang, W. Tan, J. Alexander, I. Foster, and R. Madduri. Recommend-as-you-go: A novel approach supporting services-oriented scientific workflow reuse. In *2011 IEEE International Conference on Services Computing*, pages 48–55. IEEE, 2011.

[22] L. Zheng, V. Noroozi, and P. S. Yu. Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the tenth ACM international conference on web search and data mining*, pages 425–434, 2017.