Cross-View Graph Alignment for Mashup Recommendation

Chunyu Wei, Yushun Fan, Zhixuan Jia, and Jia Zhang, *Senior Member, IEEE*

***Abstract*—As the adoption of Service-Oriented Computing continues to grow, the number of web services has increased significantly, which makes service recommendation become an essential tool to assist users in selecting suitable services. However, a single service cannot satisfy the complex requirements of users, which has led to the emergence of a new technique known as Mashup, which combines services as reusable components to create value-added service compositions. Along with mashup, mashup recommendation has also become an indispensable and important component of service platforms. On service platforms, there are many heterogeneous entities and complex relationships between them. We divide these interaction into three different views: Mashup-Invocation view, Service-Consumption view, and Mashup-Composition view. As user preferences and characteristics of services and mashups are distributed across different views, their cooperation is crucial for accurate mashup recommendation. Therefore, we propose Cross-view Graph Alignment (CGA), a framework that captures the collaborative associations dispersed across different views and enhances the representation learning of users and mashups. This the first study to jointly tackle structure- and representation-level collaboration on the service platforms for better mashup recommendation. Experiments on two real-world service datasets show that CGA outperforms state-of-the-art methods and can better improve the mashup recommendation.**

***Index Terms*—Mashup recommendation, graph neural networks, graph alignment.**

I. Introduction

With rapid development and wide adoption of service-oriented architecture (SOA), a growing number of web services with diverse functions have been developed on the Internet [1]. To help users select the optimal services from the vast array of candidates, service recommendation has emerged as a crucial instrument, using various filtering techniques, such as collaborative [2], content-based [3], or hybrid [4] filtering. Despite the large number of diverse services available, situations still arise where a single service cannot satisfy the users' complex requirements. The characteristics of web services have led to a new application development technique, namely Mashup, which combines services as reusable components to create value-added service compositions, eliminating the need to build everything from scratch [5]. For instance, Youshu, a popular Chinese book review site, offers thousands of books for users to review and create lists. A perfect reading list for a user may involve a diverse range of books, such as mystery novels, science fictions, biographies, etc. These books are scattered across the platform, making it a cumbersome task for the user to create an optimal reading list. Therefore, the platform provides many book mashups as complete reading lists, combining suitable books for the user to choose from. This not only saves a lot of time and effort for the user but also helps the platform to present some high-quality but less popular books to the customer in the mashup, enhancing the value conversion of the platform. As a result, both the users and platforms would prefer mashups instead of single services. Therefore, both the academic and industrial sectors have shown a growing interest in developing effective mashup recommendation systems.

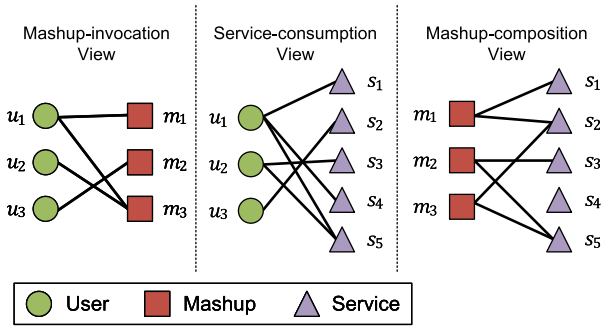


Fig. 1. Three distinct views of interactions on a service platform. User preferences are distributed across different views.

By examining previous research on service recommendations and mashup creation [3], [6], [7], we summarize the various interaction relationships on the service platform into three distinct views, which can be represented in the form of graphs as illustrated in Fig. 1: (1) Mashup-Invocation view, which illustrates user preferences through their interactions with mashups and can be organized as a User-Mashup (U-M) graph; (2) Service-Consumption view, which describes users' consuming behaviors at the granularity of services, i.e., user-service interactions in the form of a User-Service (U-S) graph; and (3) Mashup-Composition view, which delineates the detailed service composition of the mashups by connecting the mashup and its containing services as a Mashup-Service (M-S) graph.

Manuscript received 11 October 2023; revised 6 May 2024; accepted 10 May 2024. Date of publication 30 May 2024; date of current version 9 October 2024. This work was supported by the National Natural Science Foundation of China under Grant 62173199. *(Corresponding author: Yushun Fan.)*

Chunyu Wei, Yushun Fan, and Zhixuan Jia are with the Beijing National Research Center for Information Science and Technology (BNRist), Department of Automation, Tsinghua University, Beijing 100190, China (e-mail: cy-wei19@mails.tsinghua.edu.cn; fanys@tsinghua.edu.cn; jzx21@tsinghua.edu.cn).

Jia Zhang is with the Department of Computer Science, Southern Methodist University, Dallas, TX 75205 USA (e-mail: jiazhang@smu.edu).

Digital Object Identifier 10.1109/TSC.2024.3407524

These three perspectives enable us to comprehend and depict the complicated interactions within a service from different viewpoints. Intuitively, if we are able to effectively incorporate the information from these multiple views to model user preferences and mashup characteristic, we can more accurately recommend suitable mashups to the users. Nevertheless, prior studies have not accounted for the discrepancies between different views in the service platforms. For example, in Fig. 1, if we consider only the interaction relationships in the Mashup-Invocation view, we would recommend m_1 to u_2, as u_2 has a more similar service invocation history with u_1. However, upon closer examination of the Service-Consumption view and Mashup-Composition view at a finer-grained service level, we may recommend m_2 containing s_3 and s_5, as a more suitable option. Therefore, as user preferences and characteristic of services and mashups are distributed across different views, their cooperation is crucial for accurate mashup recommendation.

Based on the aforementioned observations, we propose a Cross-view Graph Alignment (CGA) framework for mashup recommendation, which captures the collaborative associations dispersed across different views and enhances the representation learning of users and mashups. The underlying intuition is that interactions in different forms (i.e., U-M, U-S, and M-S) on the service platform all reflect the user's inherent preferences and the characteristics of the mashup, even if they are dispersed across different views. If we can align and integrate user and mashup information from different views, we can more comprehensively capture user preferences for mashups and make favorable recommendations. However, to achieve this goal, we need to address the following two significant challenges:

- *Structure-level Collaboration:* In the Mashup-composition view, the inclusion relationships between mashups and services are modeled in the form of a graph. We believe that services with some similar characteristics are more likely to be included in the same mashup, which should be reflected in the topological structure of the M-S graph. Therefore, we need to align the service representations learned from other views on the platform with the inclusion relationships reflected in the Mashup-composition View.
- *Representation-level Collaboration:* In the Mashup-Invocation and Service-Consumption views, we can learn mashup representations and service representations based on the user preferences on different views. At a finer granularity, a mashup is composed of individual services, and thus, the representation of a mashup can also be aggregated from the representations of its constituent services using the M-S graph. However, the mashup representations learned from these two views are not in the same hidden space, which hinders our collaborative utilization of data from different views.

Recent advancements in graph neural network-based recommendation models, such as BGCN [8], have shown promise, yet they are not without limitations. For instance, BGCN, which first performs representation learning and preference prediction upon the views individually, only considers the cooperative signal at the level of predictions, not at the level of representations. This approach fails to guarantee the mutual enhancement of the two views. More recent methods like CrossCBR [9] attempt to model the cooperative association between two different views through cross-view contrastive learning. However, they neglect the crucial role of topological structure in graph-based recommendation models.

To address the structure-level collaboration, we propose a meta-mashup learner to learn "meta-mashups", which is used to be aligned with the existing mashup compositions under the constrain of our well-designed regularization loss. First, we use a graph-based representation learning method (GNN encoder) to learn service representation on the U-S graph. The resulting service representations are then fed into a differentiable lightweight multi-layer meta-mashup learner, which categorizes the services into different "meta-mashups". The differentiable learner can effectively capture the complex service correlations coupled with our downstream mashup recommendation task in a learnable manner. We utilize the Kullback-Leibler (KL) divergence as the regularization loss to align the reconstructed meta-mashup and the existing mashups. Taking one step further, predicting the same number of meta-mashups as existing mashups requires extensive resource costs, as the number of existing mashups is very large, reaching tens of thousands in magnitude, limiting the application to large-scale service platforms. To accelerate the computing, we cluster existing mashups into groups, and then classify services into these groups to implement our regularization loss.

To address the representation-level collaboration, we propose cross-view graph contrastive learning to align mashup representations from different views. The intuition behind this is that the Mashup-Invocation and Service-Consumption views represent two distinct but correlated perspectives on user-mashup preferences. Specifically, on the U-M graph, we use a graph-based representation learning method (GNN encoder) to learn the mashup collaborative representation in the Mashup-Invocation view. Analogously, on the U-S graph, we employ another GNN to generate the representations of services and aggregate the representations of compositional services as the mashup compositional representation based on the M-S graph. Through contrastive learning, we aim to maximize the mutual information of the mashup between the two views. This ensures that the model consistently aligns the representations from both views, enabling better graph-alignment of the mashup and capturing information from both the service and user side. Finally, we unify the mashup recommendation task and the cross-view graph alignment task under a primary and auxiliary learning framework. By jointly optimizing the two tasks and leveraging the interplay of all the components, we achieve significant gains in the performance of the mashup recommendation task.

This paper presents the following main contributions:

- We propose CGA, a framework that captures the collaborative associations dispersed across different views and enhances the representation learning of users and mashups. This is the first study to jointly tackle structure- and representation-level collaboration on service platforms for better mashup recommendation.

- We propose a meta-mashup learner constrained by a designed regularization loss for better aligning the service characteristics with the existing mashup composition.
- We also leverage advances in contrastive learning to better align user preferences on services and mashups.
- Our experiments on real-world service datasets show that CGA outperforms state-of-the-art methods and can better improve the mashup recommendation.

The remainder of this article is organized as follows. Section II formally defines the problem. Section III introduces our CGA model framework in detail. Section IV presents conducted experiments with analyses. Section V discusses related work. Finally, Section VI draws conclusions.

II. Problem Definition

The following section will provide a formal definition of the problem related to mashup recommendation on service platform. Additionally, we will introduce some important notations that will help us to clarify our concepts [10].

Definition 1. (Service Ecosystem): In a service ecosystem, $\mathbb{U} = \{u_1, u_2, \ldots, u_M\}$ and $\mathbb{S} = \{s_1, s_2, \ldots, s_N\}$ denote the sets of *users* and *services*, respectively.

Traditional service platforms that recommend individual services to users are no longer sufficient to meet the increasingly complex needs of users. Therefore, on service platforms, many developers use services as reusable components to create value-added service compositions, without building everything from scratch. We call these collections of services that work together collaboratively as a whole "mashups", which we formally define as follows:

Definition 2. (Mashup): A mashup m_t represents a bunch of services $\{s_t^1, s_t^2, \ldots\}, s_t^i \in \mathbb{S}$, which collaborate functionally to achieve one or more objectives. The entire set of mashups can be represented as $\mathbb{M} = \{m_1, m_2, \ldots, m_t\}$.

Based on the above definition, we can establish a Mashup-Composition View on the service platform to describe the relationship between mashups and services, which is presented in the form of an Mashup-Service (M-S) graph. The specific definition is as follows:

Definition 3. (Mashup Composition View): Let $\mathcal{G}_{MS} = (\mathbb{M} \cup \mathbb{S}, \mathcal{E}_{MS})$ represent a *M-S Graph*, where $\mathcal{E}_{MS} = \{e_{ms} | r_{ms} = 1, u \in \mathbb{U}, s \in \mathbb{S}\}$ is the edge set denoting the inclusion relationships between mashups and services if there exists an edges between them.

On service platforms, users have various forms of interactions, i.e., services and mashups, and all of these interactions can reflect users' personal preferences. Therefore, according to the different forms of user interactions, we define two additional views on the platform as follows:

Definition 4. (Service Consumption View): Following most existing works [11], [12], we represent Service-Consumption view as a user-service bipartite graph $\mathcal{G}_{US} = \{\mathbb{U} \cup \mathbb{S}, \mathcal{E}_{US}\}$, where the edge set $\mathcal{E}_{US} = \{e_{us} | r_{us} = 1, u \in \mathbb{U}, s \in \mathbb{S}\}$ represents the services consumed by the users.

Definition 5. (Mashup Invocation View): Similarly, the history of user-invoked mashups can be constructed as a Mashup-Invocation view, which can also be represented by a user-mashup bipartite graph $\mathcal{G}_{UM} = \{\mathbb{U} \cup \mathbb{M}, \mathcal{E}_{UM}\}$, where the edge set $\mathcal{E}_{UM} = \{e_{um} | r_{um} = 1, u \in \mathbb{U}, m \in \mathbb{M}\}$ represents the mashups invoked by the users.

The three views defined above are commonly present in service platforms, and encompass distinct information about user preferences and service characteristics, which heuristically enables the cooperative effect between the these different views and improve the mashup recommendation. We formally define this problem as follows:

Problem Formulation: Given the existing relation graphs $\{\mathcal{G}_{MS}, \mathcal{G}_{US}, \mathcal{G}_{UM}\}$, our task is to capture the collaborative associations dispersed across different views and learn the comprehensive representation of users and mashups. Then we predict the unseen user-mashup interactions in $\mathcal{G}_{UM}$.

III. Methodology

In this section, we first outline the overall architecture of our CGA framework and give detailed descriptions of its main components, then analyze its learning process including the design of different types of loss functions, followed by discussing the computation complexity of CGA.

A. Model Architecture

Fig. 2 presents the overall framework of our CGA. The framework is introduced from left to right. (1) The GNN encoder, which learns the representation based on interactions of the Mashup-Invocation view and Service-Composition view. It obtains user's invocation/consumption representations, mashup invocation representation and service consumption representation. (2) The graph aggregator incorporates the structure of the M-S graph and learn the compositional representation of the mashup. (3) The cross-view graph contrastive learning is performed to align both user representations from U-M and U-S graphs, as well as the invocation and compositional representations of the mashup. (4) The meta-mashup learner captures the complex relationships within the service consumption representation to map similar services into meta-mashups. And the regularization loss constrains the consistency between the meta-mashups and existing mashups for structure-level collaboration.

B. Graph-Based Representation Learning

To start with the primary element of CGA, our goal is to learn the representations of users, mashups and services from two different views: Mashup-Invocation view and Service-Consumption view. The raw U-M and U-S graph explicitly represents the interaction information as its links, and a node's local structure (i.e., the topology of its multi-hop neighbors) is shown to encode a user's preference or an service or mashup's characteristic [11]. To capture the collaborative signal alongside the local topology, we exploit the high-order connectivity following the recent advance in LightGCN [12]. Here, we elaborate in detail on the graph-based learning processes of various representations.

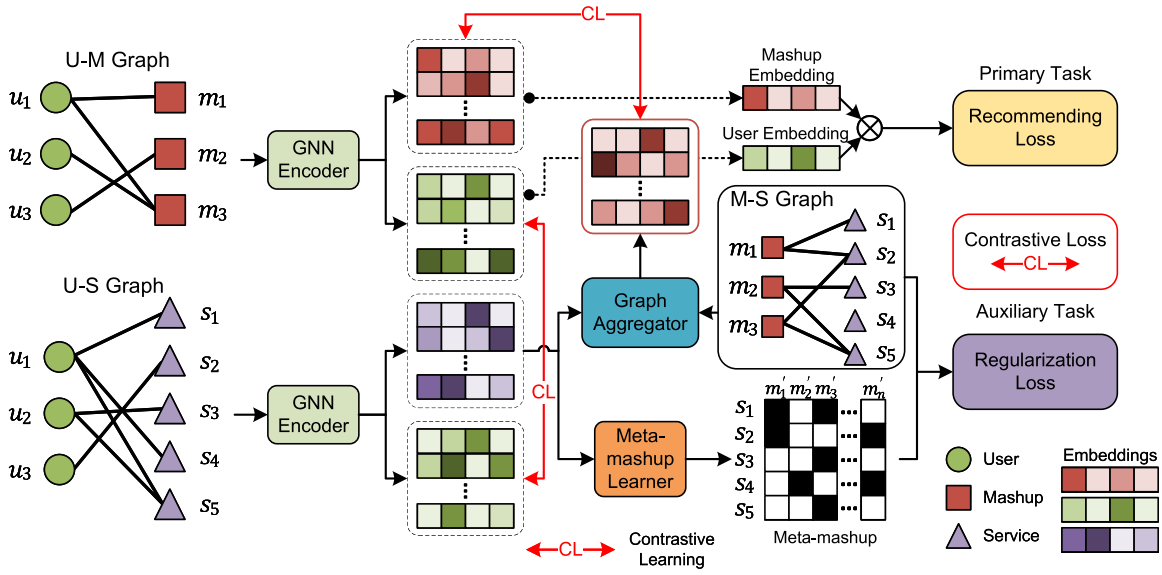


Fig. 2. Model Framework. The overview of the model structure of CGA. The regularization loss aligns the learned meta-mashup with the existing mashup composition for structure-level collaboration. The contrastive loss, indicated by the two Red bidirectional arrows in the diagram, is used for cross-view graph contrastive learning for representation-level collaboration.

1) Learning on Mashup-Invocation View: In order to better capture the user preferences and mashup characteristics exhibited in mashup invocation, we first establish the U-M graph based on the user-mashup interaction matrix, as defined in Definition 5. Afterwards, we design a GNN encoder to learn the user and mashup invocation representations by simulating information propagation on the interaction between users and mashups. More specifically, the information propagation process can be formulated as follows:

$$\mathbf{e}_{u,MI}^{(k)} = \sum_{m \in \mathcal{N}_u} \frac{1}{\sqrt{\mathcal{N}_m}\sqrt{\mathcal{N}_u}} \mathbf{e}_{m,MI}^{(k-1)}, \tag{1}$$

$$\mathbf{e}_{m,MI}^{(k)} = \sum_{u \in \mathcal{N}_m} \frac{1}{\sqrt{\mathcal{N}_u}\sqrt{\mathcal{N}_m}} \mathbf{e}_{u,MI}^{(k-1)}, \tag{2}$$

where $\mathbf{e}_{u,MI}^{(k)} \in \mathbb{R}^d$ and $\mathbf{e}_{m,MI}^{(k)} \in \mathbb{R}^d$ represent the k-th layer's representation learned from the Mashup Invocation (MI) view, which are also referred to as user invocation representation and mashup invocation representation, respectively. Additionally, $\mathbf{e}_{u,MI}^{(0)}$ and $\mathbf{e}_{m,MI}^{(0)}$ are randomly initialized at the beginning of the training. Also, $\mathcal{N}_u$ and $\mathcal{N}_m$ are the neighbors of user u and mashup m on the U-M graph $\mathcal{G}_{UM}$, respectively. And, d is the embedding size.

To carry out the propagation process for all users and mashups on $\mathcal{G}_{UM}$ simultaneously, we transform (1) and (2) into a matrix form:

$$\begin{aligned}\mathbf{E}_{MI}^{(0)} &= [\mathbf{E}_{u,MI}^{(0)}, \mathbf{E}_{m,MI}^{(0)}] \\ &= [\mathbf{e}_{u_1,MI}^{(0)}, \ldots, \mathbf{e}_{u_M,MI}^{(0)}, \mathbf{e}_{m_1,MI}^{(0)}, \ldots, \mathbf{e}_{m_t,MI}^{(0)}],\end{aligned} \tag{3}$$

$$\mathbf{E}_{MI}^{(k)} = \mathcal{L}_{\mathcal{G}_{UM}} \mathbf{E}_{MI}^{(k-1)}, k \in \mathbb{N}^+, \tag{4}$$

$$\mathcal{L}_{\mathcal{G}_{UM}} = \mathbf{D}_{\mathcal{G}_{UM}}^{-\frac{1}{2}} \mathbf{A}_{\mathcal{G}_{UM}} \mathbf{D}_{\mathcal{G}_{UM}}^{-\frac{1}{2}} \tag{5}$$

where $\mathbf{E}_{MI}^{(k)}$ is the new embedding matrix after the k-th propagation, and $\mathcal{L}_{\mathcal{G}_{UM}}$ is the Laplacian matrix of the U-M graph $\mathcal{G}_{UM}$. Formally, $\mathbf{A}_{\mathcal{G}_{UM}}$ is the adjacency matrix of the U-M graph $\mathcal{G}_{UM}$, and $\mathbf{D}_{\mathcal{G}_{UM}}$ is the diagonal degree matrix of the U-M graph $\mathcal{G}_{UM}$. Each of $\mathbf{D}_{\mathcal{G}_{UM}}$'s diagonal elements $\mathbf{D}_{\mathcal{G}_{UM}}[i,i] = |\mathcal{N}_i|$ represent the degree of the vertex u_i or m_i in the $\mathcal{G}_{UM}$. We can get that element $\mathcal{L}_{ik} = \frac{1}{\sqrt{|\mathcal{N}_i||\mathcal{N}_k|}}$.

By utilizing the matrix form for propagation, we are able to not only update all user and mashup representations within the U-B graph simultaneously, but also facilitate the batch calculations. Additionally, the use of the matrix format allows for seamless integration of multiple propagation processes as a united module, allowing for extraction of high-order invocation signals in the Mashup-Invocation view. Different from the traditional GCN propagation rule, we follow LightGCN to abandon the use of feature transformation and nonlinear activation, which has been shown to greatly reduce computational complexity while improving the effectiveness of representation learning on bipartite graphs.

After K iterations, each node encodes the information of the farthest K-order node after information propagation in the K-th layer. Therefore, we perform a weighted sum on all K layers' embeddings to obtain a comprehensive representation of each node, where the weight of each layer k is set as $\frac{1}{k+1}$. As a result, the innovation representation of the user and mashup is formally represented as follows:

$$\mathbf{E}_{MI}^{(K)} = [\mathbf{e}_{u_1,MI}^{(K)}, \ldots, \mathbf{e}_{u_M,MI}^{(K)}, \mathbf{e}_{m_1,MI}^{(K)}, \ldots, \mathbf{e}_{m_t,MI}^{(K)}], \tag{6}$$

$$\mathbf{e}_{u,MI}^{*} = \sum_{k=0}^{K} \frac{1}{k+1} \mathbf{e}_{u,MI}^{(k)}, \mathbf{e}_{m,MI}^{*} = \sum_{k=0}^{K} \frac{1}{k+1} \mathbf{e}_{m,MI}^{(k)}. \tag{7}$$

2) Learning on Service-Consumption View: From the perspective of Service-Consumption view, the interaction information between users and services encodes users' preferences and service characteristics. In order to capture this information, we also model the user-service interaction relationship as a User-Service (U-S) bipartite graph. Similar to the learning process of MI view, we also propose a simple but efficient GNN encoder for consumption representation learning. The detailed formulas are defined as follows:

$$\mathbf{e}_{u,SC}^{(k)} = \sum_{s \in \mathcal{N}_u'} \frac{1}{\sqrt{\mathcal{N}_s'}\sqrt{\mathcal{N}_u'}} \mathbf{e}_{SC}^{(k-1)}, \tag{8}$$

$$\mathbf{e}_{s,SC}^{(k)} = \sum_{u \in \mathcal{N}_s'} \frac{1}{\sqrt{\mathcal{N}_u'}\sqrt{\mathcal{N}_s'}} \mathbf{e}_{u,SC}^{(k-1)}, \tag{9}$$

where $\mathbf{e}_{u,SC}^{(k)} \in \mathbb{R}^d$ and $\mathbf{e}_{s,SC}^{(k)} \in \mathbb{R}^d$ represent the k-th layer's representation learned from the Service Consumption (SC) view, which are also referred to as user consumption representation and service consumption representation, respectively. Similarly, $\mathbf{e}_{u,SC}^{(0)}$ and $\mathbf{e}_{s,SC}^{(0)}$ are also randomly initialized at the beginning of the training. Also, $\mathcal{N}_u'$ and $\mathcal{N}_s'$ are the neighbors of user u and service s on the U-S bipartite graph $\mathcal{G}_{US}$, respectively.

Therefore, the matrix form of the service consumption can be formalized as follows:

$$\begin{aligned}\mathbf{E}_{SC}^{(0)} &= [\mathbf{E}_{u,SC}^{(0)}, \mathbf{E}_{s,SC}^{(0)}] \\ &= [\mathbf{e}_{u_1,SC}^{(0)}, \ldots, \mathbf{e}_{u_M,SC}^{(0)}, \mathbf{e}_{s_1,SC}^{(0)}, \ldots, \mathbf{e}_{s_N,SC}^{(0)}],\end{aligned} \tag{10}$$

$$\mathbf{E}_{SC}^{(k)} = \mathcal{L}_{\mathcal{G}_{US}} \mathbf{E}_{SC}^{(k-1)}, k \in \mathbb{N}^+, \tag{11}$$

$$\mathcal{L}_{\mathcal{G}_{US}} = \mathbf{D}_{\mathcal{G}_{US}}^{-\frac{1}{2}} \mathbf{A}_{\mathcal{G}_{US}} \mathbf{D}_{\mathcal{G}_{US}}^{-\frac{1}{2}}, \tag{12}$$

where $\mathbf{E}_{SC}^{(k)}$ is the updated embedding matrix after the k-th propagation, and $\mathcal{L}_{\mathcal{G}_{US}}$ is the Laplacian matrix of the U-S bipartite graph $\mathcal{G}_{US}$. Similarly, $\mathbf{A}_{\mathcal{G}_{US}}$ is the adjacency matrix of the U-S graph $\mathcal{G}_{US}$, and $\mathbf{D}_{\mathcal{G}_{US}}$ is the diagonal degree matrix of the U-S bipartite graph $\mathcal{G}_{US}$.

Afterwards, we also perform a weighted sum on all K layers' embeddings to obtain a comprehensive representation of each node on U-S graph, where the weight of each layer k is set as $\frac{1}{k+1}$. Therefore, the consumption representation of the user and service is formally represented as follows:

$$\mathbf{E}_{SC}^{(K)} = [\mathbf{e}_{u_1,SC}^{(K)}, \ldots, \mathbf{e}_{u_M,SC}^{(K)}, \mathbf{e}_{m_1,SC}^{(K)}, \ldots, \mathbf{e}_{m_t,SC}^{(K)}], \tag{13}$$

$$\mathbf{e}_{u,SC}^{*} = \sum_{k=0}^{K} \frac{1}{k+1} \mathbf{e}_{u,SC}^{(k)}, \mathbf{e}_{s,SC}^{*} = \sum_{k=0}^{K} \frac{1}{k+1} \mathbf{e}_{s,SC}^{(k)}. \tag{14}$$

3) Mashup Compositional Representation Learning: Based on the consumption representation of services $\mathbf{e}_{u,SC}^{*}$ and the Mashup-Service (M-S) graph $\mathcal{G}_{MS}$ that records the composition information of the mashup, we can obtain the compositional representation of the mashups on the SC view $\mathbf{e}_{m,SC}$ by performing average pooling on the containing services' consumption representations. We formalize the process as follows:

$$\mathbf{e}_{m,SC}^{*} = \frac{1}{|\mathcal{N}_m''|} \sum_{s \in \mathcal{N}_m''} \mathbf{e}_{s,SC}^{*}. \tag{15}$$

Here, $\mathcal{N}_m''$ denotes the mashup m's neighbors on the Mashup-Service (M-S) graph, which also means the services contained in the mashup m.

To summarize, we propose a simple but efficient graph encoder to learn the invocation representation of users and mashups, denoted as $\mathbf{e}_{u,MI}^{*}$ and $\mathbf{e}_{m,MI}^{*}$ respectively, on the MI view. Similarly, we utilize a graph encoder with the same structure to learn the consumption representations of users and services, denoted as $\mathbf{e}_{u,SC}^{*}$ and $\mathbf{e}_{s,SC}^{*}$ respectively, on the SC view. Then, based on the mashup compositional information on the Mashup-Service (M-S) graph, we aggregate the service consumption representations $\mathbf{e}_{s,SC}^{*}$ to obtain the mashup representation $\mathbf{e}_{m,SC}^{*}$ on the SC view. In this way, we obtain the user's invocation representation $\mathbf{e}_{u,MI}^{*}$ and consumption representation $\mathbf{e}_{u,SC}^{*}$, as well as the mashup's invocation representation $\mathbf{e}_{m,MI}^{*}$ and compositional representation $\mathbf{e}_{m,SC}^{*}$.

C. Meta-Mashup Learning for Structure Regularization

We have a potential assumption that the characteristics of services are reflected in their consumption histories, which are encoded in their consumption representations $\mathbf{e}_{s,SC}^{*}$. In addition, many previous studies [7], [13], [14] have found that services within the same mashup often have some similar characteristics. Based on the above two conclusions, we believe that, through the consumption representation of services, we are able to reconstruct existing mashups to a certain extent. Therefore, we designed a lightweight and learnable meta-mashup learner, which learns to map all services to a certain number of meta-mashups using the learned service consumption representation. Furthermore, we designed a novel regularization loss to constrain the learned meta-mashup structure to be as similar as possible to the existing mashup structure, which is the Mashup-Service bipartite (M-S) graph, in order to achieve structure-level collaboration across different views.

Formally, we compute the meta-mashup assignment of services using a multi-layer perceptron (MLP) with softmax operation on the learned service consumption representation $\mathbf{e}_{s,SC}^{*}$, which outputs t binary predictions as a meta-mashup incidence vector $\hat{\mathbf{p}}_s$:

$$\hat{\mathbf{p}}_s = Softmax(ReLU(\mathbf{E}_{s_i,SC}^{*}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2), \tag{16}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d \times d}$, $\mathbf{W}_2 \in \mathbb{R}^{d \times t}$, $\mathbf{b}_1 \in \mathbb{R}^d$, and $\mathbf{b}_2 \in \mathbb{R}^t$ are trainable weight matrices, in which t is the number of total mashups to learn. And $\hat{\mathbf{p}}_{s,m}$ denotes the m-th entry of vector $\hat{\mathbf{p}}_s$, which represents the probability of which the service is belong to the meta-mashup m after the projection.

Recall that our goal is to align the meta-mashup structure learned through consumption representation with the existing mashup-service composition view. As illustrated in Fig. 3, we consider the reconstructed meta-mashup as a probability distribution of services across different categories, and we also convert the presence of services in the original mashup into a probability distribution. In this way, we can transform the alignment of the two structures into the alignment of two probability distributions. Based on this setting, it is natural to think that

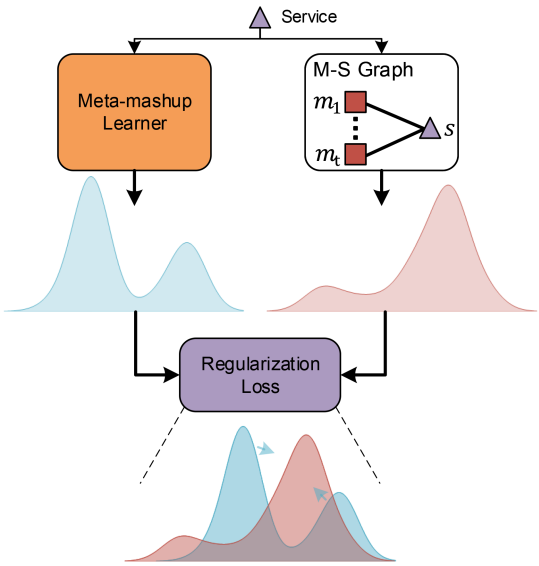


Fig. 3. Structure-level Collaboration by KL-divergence. Blue distribution is predicted by the meta-mashup learner, while the red distribution is transformed from the original M-S graph structure. Regularization loss is implemented by the KL-divergence, which forces the blue distribution to align with the original red distribution.

the similarity between two probability distributions can be measured by Kullback-Leibler divergence (KL-divergence) [15]. By forcing the KL-divergence between them to decrease, we can promote the alignment of the two structures.

Formally, we first transform the the existing mashup-service composition view into the category distribution of corresponding services.

$$\mathbf{p}_{s,m} = \frac{r_{ms}}{\sum_{k=1}^{t} r_{ks}}, \tag{17}$$

where r_{ms} and r_{ks} denote whether there exists interaction between mashup s or k and service s. Thus we have two distribution $\mathbf{p}_s = [\mathbf{p}_{s,1}, \mathbf{p}_{s,2}, \ldots, \mathbf{p}_{s,t}]$ and the reconstructed distribution $\hat{\mathbf{p}}_s$. The KL-divergence between $\mathbf{p}_s$ and $\hat{\mathbf{p}}_s$ can be formulated as:

$$KL(\hat{\mathbf{p}}_s || \mathbf{p}_s) = \sum_{m=1}^{t} \hat{\mathbf{p}}_{s,m} \log \frac{\hat{\mathbf{p}}_{s,m}}{\mathbf{p}_{s,m}}. \tag{18}$$

1) Mashup Cluster for Compression: Computing the meta-mashups to classify services into overall existing mashups requires extensive resources and limits large-scale mashup recommendation in the industry. To reduce the cost, we propose to cluster the existing mashups into multiple groups and construct the same number of meta-mashups as the groups to achieve alignment. Specifically, given the mashup set $\mathbb{M}$, we adopts k-means to cluster mashups into C categories $\{\mathbb{M}_1, \mathbb{M}_2, \ldots, \mathbb{M}_C\}$ based on the corresponding invocation representation $\mathbf{e}^*_{m,MI}$. Afterwards, the goal of the learned meta-mashups has become classifying services into the group containing the mashups they belong to. And the KL-divergence in (18) has become a measure of the difference between the probability distribution of the original service categories and the learned service categories.

D. Cross-View Graph Contrastive Learning

Recall that we obtain the user and mashup representations in different views, denoted as $\mathbf{e}^*_{u,MI}$ and $\mathbf{e}^*_{u,SC}$, as well as $\mathbf{e}^*_{m,MI}$ and $\mathbf{e}^*_{m,SC}$. We believe that each view captures a distinctive aspect of the user's preferences or the mashup characteristic, and the two views need to work cooperatively to maximize the overall modeling capacity. Thus, we employ a cross-view contrastive loss to model the cross-view cooperative association. Formally, we follow SimCLR [16] and adopt the contrastive loss, InfoNCE [17], to maximize the agreement of the same users/mashups representation from different views and minimize that of different users/mashups:

$$\mathcal{L}^U_{CL} = \frac{1}{|\mathbb{U}|} \sum_{u_i \in \mathbb{U}} -\log \frac{\exp\left(s(\mathbf{e}^*_{u_i,MI}, \mathbf{e}^*_{u_i,SC})/\tau\right)}{\sum_{u_j \in \mathbb{U}} \exp\left(s(\mathbf{e}^*_{u_i,MI}, \mathbf{e}^*_{u_j,SC})/\tau\right)} \tag{19}$$

$$\mathcal{L}^M_{CL} = \frac{1}{|\mathbb{M}|} \sum_{m_i \in \mathbb{M}} -\log \frac{\exp\left(s(\mathbf{e}^*_{m_i,MI}, \mathbf{e}^*_{m_i,MC})/\tau\right)}{\sum_{m_j \in \mathbb{M}} \exp\left(s(\mathbf{e}^*_{m_i,MI}, \mathbf{e}^*_{m_j,MC})/\tau\right)} \tag{20}$$

where $\mathcal{L}^U_{CL}$ and $\mathcal{L}^M_{CL}$ represent the cross-view graph contrastive losses for users and mashups, respectively. $s(\cdot)$ measures the similarity between two vectors, which is set as cosine similarity function. τ is the hyper-parameter, known as the *temperature* in softmax.

E. Mashup Prediction

After obtaining the user and mashup representations from different views in Section III-B, which are $\mathbf{e}^*_{u,MI}$ and $\mathbf{e}^*_{u,SC}$, as well as $\mathbf{e}^*_{m,MI}$ and $\mathbf{e}^*_{m,SC}$, we first utilize the classical inner product to calculate the user-mashup match score in different views. And for a more comprehensive prediction, we combine the scores in different views with a weighted sum:

$$\hat{s}_{um} = \alpha * \mathbf{e}^{*T}_{u,MI} \mathbf{e}^*_{m,MI} + (1-\alpha) * \mathbf{e}^{*T}_{u,SC} \mathbf{e}^*_{m,SC}, \tag{21}$$

where $0 < \alpha < 1$ controls the weight of the user-mashup match score in the Mashup Invocation view.

1) Optimization: To optimize model parameters, we adopt the pairwise Bayesian Personalized Ranking (BPR) loss [18], which has been extensively used in most implicit recommendation processes. BPR operates on the assumption that mashups that have been consumed should be given higher predicted values than those that have not been observed, as they better reflect a user's preferences. For each positive user-mashup pair $< u_i, m_j >$, we randomly sample multiple negative mashups from the unobserved mashups of the user, which is denoted as m_k. The *BPR* pairwise ranking loss is defined as follows:

$$\mathcal{L}_{BPR} = \sum_{(u_i, m_j, m_k) \in \mathfrak{D}} -ln\sigma(\hat{s}_{u_i m_j} - \hat{s}_{u_i m_k}), \tag{22}$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$ is the logistic sigmoid function and $\mathfrak{D}$ represents the set of pairwise training instances.

Finally, we unify the mashup recommendation task and the cross-view graph alignment in structure- and representation-level in a joint learning framework:

$$\mathcal{L} = \mathcal{L}_{BPR} + \lambda_1 \sum_{s \in \mathbb{S}} KL(\hat{\mathbf{p}}_s || \mathbf{p}_s) + \lambda_2 (\mathcal{L}_{CL}^{U} + \mathcal{L}_{CL}^{M}) + \lambda_3 ||\Theta||_2^2, \tag{23}$$

where λ_1, λ_2, and λ_3 are the paramters to balance different tasks. The last term of (23) is a regularization term, which is a L_2 norm to prevent overfitting.

2) Time Complexity Analysis: The time complexity of our CGA is mainly composed of three parts, which we analyze separately. 1) Graph-based representation learning: The time complexity of graph convolution in each iteration on Mashup-Invocation and Service-Consumption view is $O(2|\mathcal{E}_{\mathcal{UM}}|Kd)$ and $O(2|\mathcal{E}_{\mathcal{US}}|Kd)$, respectively, where d is the embedding size. Thus the overall complexity for the whole training phase is $O(2|\mathcal{E}_{\mathcal{UM}} + \mathcal{E}_{\mathcal{US}}|Kds\frac{|\mathcal{E}_{\mathcal{UM}}|}{B})$, where $|\mathcal{E}_{\mathcal{UM}}|$ is the number of edges in the user-mashup bipartite graph, which is also the number of the training samples. Analogously, $|\mathcal{E}_{\mathcal{US}}|$ is the number of edges in the user-service bipartite graph. B is the batch size and s is the epoch number. 2) Meta-Mashup learning: It involves a MLP projection module, which has a complexity of $O(|\mathbb{S}|d^2)$ for all services in each iteration. So the complexity will be $O(|\mathbb{S}|d^2 s\frac{|\mathcal{E}_{\mathcal{UM}}|}{B})$ for the whole training phase. 3) Cross-view Graph Contrastive Learning: As defined in (19) and (20), we treat all user nodes and mashup nodes as negative samples when calculating the InfoNCE loss. For the user-side contrastive loss (19), the numerator and denominator within a batch have complexities of $O(Bd)$ and $O(B|\mathbb{U}|d)$, respectively, where $|\mathbb{U}|$ represents the number of users. The total complexity per epoch for user side is $O(B(|\mathbb{U}|+1)d)$. Similarly, the total complexity per epoch for mashup side is $O(B(|\mathbb{M}|+1)d)$. As a result, the time complexity for the entire training phase is $O(B(|\mathbb{M}| + |\mathbb{U}| + 2)d\frac{|\mathcal{E}_{\mathcal{UM}}|}{B}s) = O((|\mathbb{M}| + |\mathbb{U}| + 2)d|\mathcal{E}_{\mathcal{UM}}|s)$. One way to reduce time complexity is to only consider users (or mashups) within the same batch as negative samples [16], [19], resulting in a total time complexity of $O((2B+2)d|\mathcal{E}_{\mathcal{UM}}|s)$.

To summarize, we can find the bottleneck of the overall time complexity lies in the Graph-based representation learning, which is inevitable in most graph-based recommendation tasks. Moreover, the time complexity of this part is greatly reduced, thanks to the removal of feature transformation and non-linear activation in each graph convolutional layer as illustrated in Section III-B. In practice, we usually have $K \leq 3$. Moreover, clustering the mashups into groups effectively reduces the dimensionality of the problem. Instead of having to deal with each individual mashup separately, you can deal with a smaller number of clusters, each representing a group of similar mashups. This simplification can significantly reduce the computational load and thereby the time complexity. In conclusion, our algorithm is computationally feasible in practice and thus support real-time query in real-world mashup recommendation system.

To help the readers better understand the parameters and the corresponding symbols mentioned above, we summarize the notations involved in the time complexity analysis in Table I.

TABLE I
NOTATIONS AND EXPLANATIONS IN TIME COMPLEXITY ANALYSIS

Notation	Explanation
$\lvert\mathcal{E}_{\mathcal{UM}}\rvert$	Number of edges in the U-M graph
$\lvert\mathcal{E}_{\mathcal{US}}\rvert$	Number of edges in the U-S graph
$\lvert\mathbb{U}\rvert$	Number of users
$\lvert\mathbb{S}\rvert$	Number of services
$\lvert\mathbb{M}\rvert$	Number of mashups
K	Layer number
d	Embedding size
s	Epoch number
B	Batch size

TABLE II
STATISTICS OF THE DATASETS

Dataset	Netease	Youshu
Users	18,528	8,039
Services	123,628	32,770
Mashups	22,864	4,771
User-Mashups Invocations	302,303	51,377
User-Service Consumptions	1,128,065	138,515
Mashup-Service Compositions	1,778,838	176,667
Avg. invocations per user	16.32	6.39
Avg. consumptions per user	60.88	17.23
Avg. services per mashup	77.80	37.03
User-Service Density	0.05%	0.05%
User-mashup Density	0.07%	0.13%

IV. EXPERIMENTS

A. Dataset Description

To verify the effectiveness of our proposed CGA, we evaluate our proposed CGA on a well-known online service datasets, namely Netease [20]. Netease Cloud Music is a prominent music service platform that allows users to discover their preferred songs or user-created playlists by searching with keywords or exploring different genres. Each song is considered an independent service, while user-generated playlists are viewed as unique mashups. Following [20], we filter the dataset by keeping only playlists with a minimum of 10 songs, songs that appeared in at least 5 playlists, and users who consumed at least 10 songs and 10 playlists. In addition to the NetEase dataset, we further validated our proposed CGA method on another benchmark dataset, Youshu [21]. This dataset, sourced from a Chinese book review site, allowed users to create lists of books, similar to the song lists in NetEase. We treated each book as a separate service and each user-created book-list as a mashup. Similar to Netease, we also filter the Youshu dataset by keeping only booklists with a minimum of 10 books, books that appeared in at least 5 booklists, and users who reviewed at least 10 books and 10 booklists.

The detailed statistics of the Netease dataset and Youshu dataset are shown in Table II.

In our study, we chose two general recommendation datasets, Youshu and Netease, to validate the effectiveness of our CGA method. These datasets, when abstracted, share several similarities with web services, making them suitable for our research. First, like web service platforms, they have a vast number of users, services, and mashups composed of services. Second,

within their ecosystems, numerous elements have a massive and heterogeneous connection, forming many different graph structures in multiple views. Third, these different graph structures reflect the preferences of users and the characteristics of services, making them ripe for comprehensive mining to aid in better representation learning. These commonalities demonstrate that our CGA method can be effectively applied on web service platforms and generalized to other recommendation tasks that meet these criteria.

B. Experimental Settings

1) Baselines: To evaluate the performance of mashup recommendation, we selected three classes of baseline methods for comparison. These include: (1) *Mashup Creation methods*, which aim to recommend related services to existing mashups to form new mashups and enhance their functionality. For comparison with our methods, we adopted the mashup modeling methods and calculated the matching score between users and mashups for recommendation; (2) *Service Recommendation methods*, which aim to recommend suitable single services to users. To adapt these methods to our experiment, we treated mashups as a special type of service, using only user-mashup interactions without considering the included services within the mashups; and (3) *Mashup Recommendation methods*, which are consistent with our problem definition and aim to recommend a bundle of services to users in the form of mashups to better meet their needs. The baseline models are as follows and the class index of each model is indicated beside its name.

- *DySR* [22] (1) An method for solving mashup creation problem by jointly addressing evolving service social and semantic gap issues.
- *coACN* [23] (1) A service recommendation framework for composition creation, which designs a domain-level attention unit to integrate domain information and construct a graph network to capture holistic information.
- *T2L2* [24] (1) A method to eliminate representation heterogeneity between services and mashups by aligning them to the same representation space.
- *CSBR* [7] (1) A method to learn the compositional semantics of a mashup, which discover potential reusable service packages and learn their compositional semantics from existing mashups and save them into a repository.
- *LightGCN* [12] (2) A state-of-the-art GCN-based general recommendation model that leverages the user-service proximity to learn node representations and generate recommendations.
- *NCF* [25] (2) A deep learning based framework combining matrix factorization (MF) with a multilayer perceptron model to learn the embedding for recommendations.
- *BundleNet* [26] (3) A method that constructs a user-bundle-item tripartite graph and employs multi-task learning to optimize the model, allowing it to effectively handle complex interactions.
- *BGCN* [8] (3) A method for mashup recommendation which decomposes the user-mashup-service relations into two separate views and uses GCN to learn representations separately.
- *CrossCBR* [9] (3) A method formulates the cross-view cooperative association in bundle recommendation, which propose a simple yet model the cooperative association between two views via cross-view contrastive learning.

TABLE III
OVERALL PERFORMANCE COMPARISON ON NETEASE

	HR@20	NDCG@20	HR@40	NDCG@40
DySR	0.0643	0.0336	0.0948	0.0423
coACN	0.0656	0.0356	0.1008	0.0443
T2L2	0.0382	0.0221	0.0656	0.0327
CSBR	0.0656	0.0348	0.0921	0.0413
LightGCN	0.0496	0.0254	0.0795	0.0334
NCF	0.0463	0.0241	0.0736	0.0312
BundleNet	0.0391	0.0201	0.0661	0.0271
BGCN	0.0491	0.0258	0.0829	0.0346
CrossCBR	0.0842	0.0457	0.1264	0.0569
CGA	**0.0887**	**0.0481**	**0.1306**	**0.0588**

2) Evaluation Metrics: To evaluate the effectiveness of mashup recommendation for all algorithms, we employed two commonly used metrics: Normalized Discounted Cumulative Gain@K (NDCG@K) and Hit Ratio@K (HR@K). Both metrics are better when higher. NDCG@K is position-aware and assigns higher scores to hits at top ranks, while HR@K determines if the test mashup is included in the recommendation list. For each user, the metrics are defined as follows:

$$NDCG@K = \frac{1}{R_N} \sum_{i=1}^{N} \frac{2^{rel_i - 1}}{\log_2(1+i)} \tag{24}$$

$$HR@K = \frac{\sum_{i=1}^{K} rel_i}{|y_u^{test}|} \tag{25}$$

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

C. Comparative Analysis on Overall Performance

We summarize the performance of different algorithms in terms of HR@K and NDCG@K with recommendation size $K = 20, 40$ in Tables III and IV. The best performing methods are bold, while the strongest baselines are underlined. From the the empirical results, we find our CGA outperforms other methods in all evaluation metrics on both datasets with varying difficulty levels, and one-sample t-tests show that the improvements of CGA over the strongest baseline are statistically significant (p-value < 0.05). From the results, we drew three conclusions:

- Graph-based methods have better performance than other methods. For example, DySR and coACN outperform T2L2 and CSBR, and LightGCN performs better than NCF. This indicates that the graph learning module is more

TABLE IV
OVERALL PERFORMANCE COMPARISON ON YOUSHU

	HR@20	NDCG@20	HR@40	NDCG@40
DySR	0.2324	0.1469	0.3405	0.1526
coACN	0.2389	0.1440	0.3302	0.1680
T2L2	0.1822	0.1123	0.2620	0.1402
CSBR	0.2252	0.1384	0.3178	0.1477
LightGCN	0.2286	0.1344	0.3190	0.1592
NCF	0.2136	0.1273	0.2952	0.1488
BundleNet	0.1895	0.1125	0.2675	0.1335
BGCN	0.2347	0.1345	0.3248	0.1593
CrossCBR	0.2813	0.1668	0.3785	0.1938
CGA	**0.2936**	**0.1747**	**0.3928**	**0.2024**

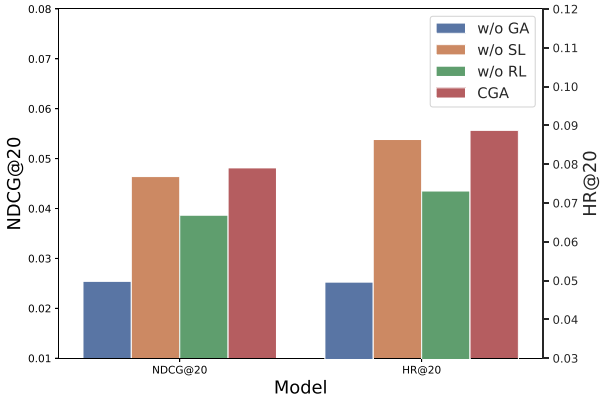


Fig. 4. Performance of CGA and its variants.

effective in modeling the user-mashup and user-service collaborative signals.

- Considering the composition of services within a mashup at a finer granularity, rather than simply considering the interaction between the mashup as an entity and the user, is more effective for mashup recommendation. For instance, in Mashup Creation methods, the best performing methods DySR and coCAN have better performance than the best performing method LightGCN in service recommendation methods. This proves that the compositional information of mashups can help us better learn the representation of mashups, in addition to user-mashup interaction information. At the same time, users' preferences for mashups can be better captured through finer-grained service preferences. This also better supports the necessity of performing cross-view graph alignment.
- Methods that take into account service-consumption information, such as BGCN and CrossCBR, perform better than methods that only consider mashup-invocation information, such as LightGCN and NCF. This proves that the information from service consumption indeed brings additional information gain and enhances our model's ability to model user preferences.
- Multi-view cooperation is crucial for improving the effectiveness of mashup recommendation. For example, CrossCBR and our CGA perform better than BGCN, which independently performs presentation learning in different views and fuses view-specific decisions at the final prediction stage. We believe that this approach does not fully exploit the potential of multi-view cooperation. Instead, multi-view cooperation during the representation learning stage forces the two views to interact and exchange information, allowing us to learn a more comprehensive representation that reflects user preferences and mashup characteristics.
- Our CGA consistently yields the best performance on both the Netease dataset and the Youshu dataset, which improves over the strongest baseline CrossCBR by 5.26% with respect to NDCG@20 and 3.36% to HR@40 on Netease, respectively. This proves that when performing multi-view cooperation, especially in graph-based recommendation methods, it is crucial to comprehensively consider both structure-level and representation-level collaboration for aligning graphs distributed in multiple views. One possible reason is that in the data structure like graphs, in addition to the features of the nodes, the topological structure between the nodes is also a very important component. Therefore, considering structure-level collaboration is very important for the effectiveness of cross-view graph alignment.

D. Study of the Multi-View Cooperation

In order to better explore the respective roles of structure- and representation-level collaboration in graph alignment, we designed several variants of CGA and conducted ablation experiments. The design of the variants is as follows.

1) *w/o GA* A variant model of CGA without all the graph alignment. In this case, the variant degenerates into LightGCN, so we directly adopted the results in Table III.
2) *w/o SL* A variant model of CGA without structure-level collaboration.
3) *w/o RL* A variant model of CGA without representation-level collaboration.

We show the results over NDCG@20 and HR@20 of CGA and its variants in Fig. 4. From the results, we can see that both structure-level and representation-level collaboration can significantly improve the performance of graph-based mashup recommendation. Among all these variants, our CGA achieved the best performance, indicating that when we consider both structure- and representation-level collaboration at the same time, the model performs best, and removing either level of collaboration will impair the model's capability. What's more, when we compare the performance of the w/o SL and w/o RL variants, we find that w/o SL performs better than w/o RL, indicating that representation-level collaboration is more effective for multi-view graph alignment in service ecosystems. This conclusion also guides us that for the selection of weights for $\sum_{s\in\mathbb{S}} KL(\hat{\mathbf{p}}_s||\mathbf{p}_s)$ and $(\mathcal{L}_{CL}^{U}+\mathcal{L}_{CL}^{M})$ in the loss function (23).

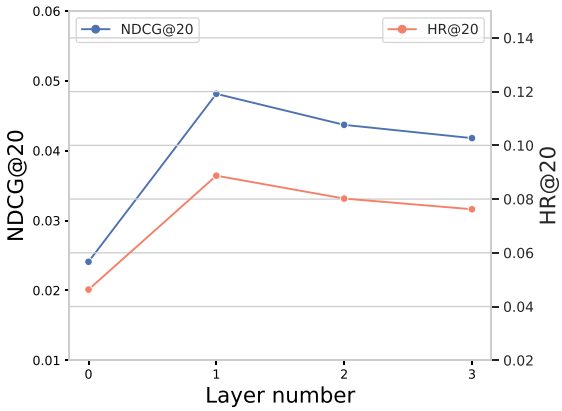


Fig. 5. Performance of CGA w.r.t. different layer number.

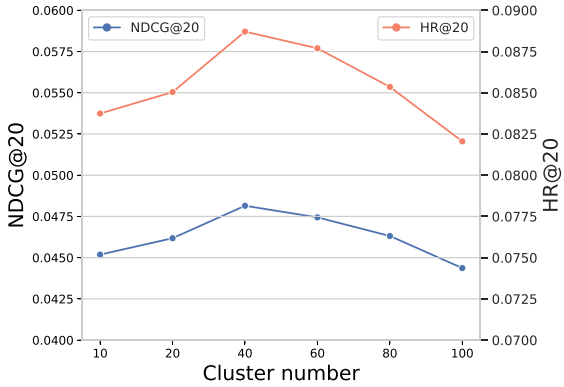


Fig. 6. Performance of CGA w.r.t. different cluster number.

E. Parameter Sensitive Studies

1) Effect of Layer Number K: In order to determine the layer number K of the graph-based representation learning in our CGA model and to ascertain if multiple stacked layers can enhance its performance, we conducted experiments on Netease datasets by varying the layer number K from 0 to 6. The results are illustrated in Fig. 5, where CGA-k denotes the model with k layers. We have the following observation:

- CGA-1 exhibits a marked enhancement in comparison to its predecessor, CGA-0. It is important to note that when the number of layers is set to 0, the model reverts to NCF [25], and the results of our experiments align with those presented in Table III. The observed improvement can be attributed to the fact that, unlike CGA-0 which solely encodes implicit collaborative signals, CGA-1 explicitly incorporates the connectivity of user-mashup and user-service into its modeling.
- Another noteworthy finding is that our CGA achieved optimal performance after just one layer of graph convolution, and further deepening the model depth does not bring additional improvement. This is different from many existing graph models, such as LightGCN [12], which achieved optimal performance after reaching three layers. The main reason for this is that we introduced a graph alignment mechanism in multiple different views in the service ecosystem, which requires us to have a more accurate modeling of the local topological structure of each node on the graph in order to achieve better alignment. Graph neural networks, essentially as a low-pass filter, suffer from the problem of over-smoothing [27], which is more prominent in our case of alignment focusing on local topological information. Therefore, our method uses one layer of graph convolution in the graph-based representation learning. By effectively reducing the number of convolution layers and surpassing multi-layer convolution by graph alignment, our method also greatly reduces the computational burden and is conducive to deployment when dealing with millions of services and mashups in real-world industry scenarios.

2) Effect of Cluster Number C: We further delve into the exploration of the parameter C, which signifies the number of clusters in mashups and governs the degree of compression applied to our meta-mashup learning quantity. We vary C from 10 to 100 while keeping other parameters fixed. We present the experimental results of NDCG@20 and HR@20 on *Netease* in Fig. 6. Fig. 6 reveals that the optimal cluster number for CGA is between 40 and 60. Further increasing the number of clusters results in a decline in recommendation performance, likely due to the fact that too many clusters can cause semantically similar mashups to be divided into different clusters, thereby making the original distribution of a certain service more dispersed, which is not conducive to our meta-mashup learner learning the correct distribution. In addition, too few cluster numbers will also affect the model's ability, mainly because too few clusters will force unrelated mashups to be classified together, resulting in semantic confusion.

TABLE V
COMPUTATION COST ON DIFFERENT CLUSTER NUMBERS

Cluster Number	Training Time (s/epoch)	
	Youshu	**Netease**
10	5.80	38.06
20	6.83	40.22
50	9.06	44.06
100	10.89	49.28
200	15.40	58.73
500	26.27	97.62

We also analyze the computation time of our algorithm under different cluster numbers. As mentioned in Section III-C1, we use a clustering algorithm to group the mashups into clusters based on their invocation patterns. This strategy can help speed up the recommendation process and reduce the overall response time. To evaluate the impact of cluster number on the computation time, we conduct experiments on both Youshu and NetEase datasets with different cluster numbers ranging from 10 to 500. We measure the average computation time of our algorithm for each cluster number, and report the results in Table V.

From Table V, we can see that the computation time increases rapidly as the cluster number increases. This is because a larger cluster number means a smaller cluster size, which leads to more clusters and more alignment operations in structure regularization. The increase of computation time is linearly correlated with

the increase of cluster number. These results show that mashup clustering can effectively reduce the computation time of our algorithm, and thus improve the efficiency and scalability of our recommendation system. However, there is a trade-off between computation time and recommendation accuracy, as a smaller cluster number may also result in a lower quality of alignment and prediction according Fig. 6. Therefore, we need to choose an appropriate cluster number according to the specific application scenario and the available computational resources.

We also provide some general guidelines on how to choose an appropriate cluster number for different datasets. In general, we can perform a grid search in the range of [10, 200] to find the best cluster number for a given dataset. However, the optimal cluster number may also depend on the diversity of the service categories in the dataset. For datasets with high service diversity, such as online shopping platforms, a larger cluster number may be more suitable, as it can capture the finer-grained preferences of the users. For datasets with low service diversity, such as book or music platforms, a smaller cluster number may be more suitable, as it can avoid overfitting and reduce the computation cost.

V. Related Work

A. Service Recommendation

The academic and industrial communities have recently shown significant interest in utilizing service recommendations for fulfilling the needs of users. The approaches for service recommendation can be categorized based on the type of information utilized in the model. These categories include content-based approaches that rely on content information, graph-based approaches that make use of historical invocation information, and hybrid approaches that combine both content and historical invocation information.

Content-based approaches are commonly used to recommend services by matching candidate services to the requirements. This involves using techniques such as keywords [28], [29], TF-IDF [13], and ontologies [30], [31] to measure the similarity between descriptions of services and requirements. However, these methods suffer from poor performance and cannot fully understand semantic meaning. Some studies have attempted to use topic models and latent semantics to improve recommendation performance [7], [32], but small data volumes and high noise in data have limited their effectiveness. Recently, researchers have turned to pre-trained language models and deep learning models to solve the service recommendation problem. For example, Bai et al. [33] designed a stacked denoising autoencoder (SADE) to extract features for the recommendation. However, these approaches have had limited success due to the difficulty of adapting existing models to service recommendations and negative transfer.

In contrast to content-based methodologies, graph-based approaches offer recommendations by examining data extracted from past interactions involving services or users. Graph-based techniques generally work together with collaborative filtering (CF). For example, Zheng et al. [34] introduces a neighborhood integrated matrix factorization approach for predicting the quality of service (QoS) of potential services. Qi et al. [35] employs a hybrid random walk to calculate the similarities between users or services, with a CF model used for service recommendation. Xie et al. [36] and Liang et al. [37] build a heterogeneous information network by incorporating various details of services to determine the similarity between services, and then employ user-based CF to rank potential services.

Given the complementarity of textual and graph data, several hybrid service recommendation methods that combine both content and historical invocation information have emerged recently. Li et al. [38] augment a latent Dirichlet allocation (LDA) model with invocation relations between requirements and services to enable the topic model to capture the association between services and requirements. Jain et al. [39] and Samanta et al. [40] utilize topic models and neighbor interaction probabilities to compute similarity scores between services and requirements, and then multiply these scores to rank candidate services. Deep learning-based methods are gradually becoming the predominant hybrid approaches. For instance, Xiong et al. [41] integrate the invocation relations between services and requirements, as well as their description similarity, into a deep neural network (DNN). Chen et al. [42] propose a preference-based neural collaborative filtering recommendation model that uses a multi-layer perceptron to capture non-linear user-service relationships and obtain abstract data representations from sparse vectors.

B. Mashup Recommendation

Rather than recommending services individually based on their contents or historical interactions, which has been explored in previous works [2], [3], mashup recommendation methods recommend a bundle of compatible services to satisfy the functional requirements of the mashup as completely as possible. Our proposed method follows this approach. Initial works just ignore the affiliated services of the mashup and just use an id to represent a mashup. Following works recognize the importance of affiliated services and develop various models to capture the additional user-service interaction and mashup-service affiliated relations. For example, Maaradji et al. [43] puts forth a frequent pair mining method for mashup development. Wu et al. [44] introduce a neural framework based on multi-model fusion and multi-task learning, which leverages a semantic component to generate mashup representations and introduces a feature interaction component to model the interaction between mashups and services. Ma et al. [45] leverage the strong feature extraction capabilities of deep learning to extract textual and interaction-based features between mashups and services. Another line of research similar to mashup recommendation is bundle recommendation [20], [46]. Liu et al. [47] build a bridge between bundle recommendation and mashup recommendation by modeling the mashup creation problem as a service bundle recommendation task. With the emergence of graph neural network-based recommendation models, Deng et al. [26] propose BundleNet and Chang et al. [8] propose BGCN. BundleNet mixups the three types of relations between

users, bundles, and items, while BGCN decomposed user preferences into item view and bundle view, which effectively captured the two types of preferences and resulted in better performance. More recently, CrossCBR [9] proposes to model the cooperative association between the two different views through cross-view contrastive learning. Despite substantial progress made in the aforementioned studies, they all only consider collaboration at the representation-level across different views. We believe these approaches is somewhat lacking, especially for graph-based recommendation models. This is because, for graphs, the topological structure is an extremely crucial element. Therefore, our cross-view graph alignment work proposed in this paper takes into consideration not only the representation-level, but also the structure-level collaboration.

C. Contrastive Learning

Contrastive Learning (CL) [48], [49] was initially introduced for training convolutional neural networks (CNNs) to learn image representations and has recently shown great success. CL has also been applied to graph data, such as DGI [50] and InfoGraph [51], which learn node representations based on the mutual information between nodes and the entire graph. Unsupervised learning models, such as that developed by Peng et al. [52], have been trained by maximizing mutual information of nodes between the input and output of a graph neural encoder. Similarly, Hu et al. [53] extended this idea to learn GCN for graph representation by building contrastive pairs between nodes and subgraphs. Additionally, GCC [54] designed the pre-training task as subgraph instance discrimination in and across networks and leveraged CL to enhance graph neural networks. However, there are limited works that utilize CL to enhance graph-based recommendation. S^3-Rec [55] employs the mutual information maximization principle to learn the correlations among attribute, item, subsequence, and sequence for sequential recommendation. A recent work, SGL [56], supplements the supervised task of recommendation with an auxiliary graph CL task that generates multiple views of a node and maximizes the agreement between different views of the same node compared to that of other nodes.

VI. Conclusion

With the development and adoption of service-oriented architecture, a large number of web services have been published on the Internet. The characteristics of these services make it possible to combine services as reusable components to create value-added service compositions, known as mashups. To avoid information overload and help users select suitable mashups, mashup recommendation has received increasing attention. Due to the diversity of interaction relationships between various types of entities on service platforms, Mashup-Invocation view, Service-Consumption view, and Mashup-Composition view can be constructed on service platforms. User preferences and characteristics of services and mashups are distributed across different views, so their cooperation is crucial for accurate mashup recommendation. In this work, we propose Cross-view Graph Alignment (CGA), a framework that captures the collaborative associations dispersed across different views and enhances the representation learning of users and mashups. This is the first study to jointly tackle structure- and representation-level collaboration on service platforms for better mashup recommendation. Extensive experiments on two real-world service datasets demonstrate that CGA outperforms state-of-the-art methods and can better improve the mashup recommendation.

References

[1] Y. Zhong, Y. Fan, K. Huang, W. Tan, and J. Zhang, "Time-aware service recommendation for mashup creation," *IEEE Trans. Serv. Comput.*, vol. 8, no. 3, pp. 356–368, May/Jun. 2015.

[2] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "QoS-aware web service recommendation by collaborative filtering," *IEEE Trans. Serv. Comput.*, vol. 4, no. 2, pp. 140–152, Second Quarter, 2011.

[3] S. Wang, Y. Ma, B. Cheng, F. Yang, and R. N. Chang, "Multi-dimensional QoS prediction for service recommendations," *IEEE Trans. Serv. Comput.*, vol. 12, no. 1, pp. 47–57, Jan./Feb. 2019.

[4] H. Mezni and M. Fayala, "Time-aware service recommendation: Taxonomy, review, and challenges," *Softw. Pract. Experience*, vol. 48, no. 11, pp. 2080–2108, 2018.

[5] J. Yu, B. Benatallah, F. Casati, and F. Daniel, "Understanding mashup development," *IEEE Internet Comput.*, vol. 12, no. 5, pp. 44–52, Sep./Oct. 2008.

[6] H. Mezni, "Temporal knowledge graph embedding for effective service recommendation," *IEEE Trans. Serv. Comput.*, vol. 15, no. 5, pp. 3077–3088, Sep./Oct. 2022.

[7] Q. Gu, J. Cao, and Y. Liu, "CSBR: A compositional semantics-based service bundle recommendation approach for mashup development," *IEEE Trans. Serv. Comput.*, vol. 15, no. 6, pp. 3170–3183, Nov./Dec. 2022.

[8] J. Chang, C. Gao, X. He, D. Jin, and Y. Li, "Bundle recommendation and generation with graph neural networks," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 3, pp. 2326–2340, Mar. 2023.

[9] Y. Ma, Y. He, A. Zhang, X. Wang, and T. Chua, "CrossCBR: Cross-view contrastive learning for bundle recommendation," in *Proc. 28th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2022, pp. 1233–1241.

[10] C. Wei, Y. Fan, and J. Zhang, "Time-aware service recommendation with social-powered graph hierarchical attention network," *IEEE Trans. Serv. Comput.*, vol. 16, no. 3, pp. 2229–2240, May/Jun. 2023.

[11] X. Wang, X. He, M. Wang, F. Feng, and T. Chua, "Neural graph collaborative filtering," in *Proc. 42nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2019, pp. 165–174.

[12] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "LightGCN: Simplifying and powering graph convolution network for recommendation," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 639–648.

[13] B. Xia, Y. Fan, W. Tan, K. Huang, J. Zhang, and C. Wu, "Category-aware API clustering and distributed recommendation for automatic mashup creation," *IEEE Trans. Serv. Comput.*, vol. 8, no. 5, pp. 674–687, Sep./Oct. 2015.

[14] L. Yao, X. Wang, Q. Z. Sheng, B. Benatallah, and C. Huang, "Mashup recommendation by regularizing matrix factorization with API co-invocations," *IEEE Trans. Serv. Comput.*, vol. 14, no. 2, pp. 502–515, Mar./Apr. 2021. [Online]. Available: https://doi.org/10.1109/TSC.2018.2803171

[15] S. Ghimire, A. Masoomi, and J. G. Dy, "Reliable estimation of KL divergence using a discriminator in reproducing Kernel Hilbert space," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2021, pp. 10 221–10 233.

[16] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 1597–1607. [Online]. Available: http://proceedings.mlr.press/v119/chen20j.html

[17] M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, Y. W. Teh and D. M. Titterington, Eds., 2010, pp. 297–304. [Online]. Available: http://proceedings.mlr.press/v9/gutmann10a.html

[18] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," in *Proc. 25th Conf. Uncertainty Artif. Intell.*, 2009, pp. 452–461.

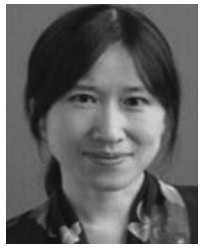
[19] T. Yao et al., "Self-supervised learning for deep models in recommendations," 2020, *arXiv: 2007.12865*.

[20] D. Cao, L. Nie, X. He, X. Wei, S. Zhu, and T. Chua, "Embedding factorization models for jointly recommending items and user generated lists," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2017, pp. 585–594.
[21] L. Chen, Y. Liu, X. He, L. Gao, and Z. Zheng, "Matching user with item set: Collaborative bundle recommendation with deep attention network," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 2095–2101.
[22] M. Liu, Z. Tu, H. Xu, X. Xu, and Z. Wang, "DySR: A dynamic graph neural network based service bundle recommendation model for mashup creation," *IEEE Trans. Serv. Comput.*, vol. 16, no. 4, pp. 2592–2605, Jul./Aug. 2023.
[23] R. Yan, Y. Fan, J. Zhang, J. Zhang, and H. Lin, "Service recommendation for composition creation based on collaborative attention convolutional network," in *Proc. IEEE Int. Conf. Web Serv.*, 2021, pp. 397–405.
[24] M. Liu, Y. Zhu, H. Xu, Z. Tu, and Z. Wang, "T2L2: A tiny three linear layers model for service mashup creation," in *Proc. 19th Int. Conf. Serv.-Oriented Comput.*, Springer, 2021, pp. 317–331.
[25] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 173–182.
[26] Q. Deng et al., "Personalized bundle recommendation in online games," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, 2020, pp. 2381–2388.
[27] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun, "Measuring and relieving the over-smoothing problem for graph neural networks from the topological view," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 3438–3445.
[28] Q. He et al., "Keyword search for building service-based systems," *IEEE Trans. Softw. Eng.*, vol. 43, no. 7, pp. 658–674, Jul. 2017.
[29] Q. He et al., "Efficient keyword search for building service-based systems based on dynamic programming," in *Proc. 15th Int. Conf. Serv. Oriented Comput.*, Springer, 2017, pp. 462–470.
[30] M. Al-Hassan, H. Lu, and J. Lu, "A semantic enhanced hybrid recommendation approach: A case study of e-government tourism service recommendation system," *Decis. Support Syst.*, vol. 72, pp. 97–109, 2015.
[31] S. Balaji B, K. N K, and R. K. R S, "Fuzzy service conceptual ontology system for cloud service recommendation," *Comput. Elect. Eng.*, vol. 69, pp. 435–446, 2018.
[32] C. Lin, A. K. Kalia, J. Xiao, M. Vukovic, and N. Anerousis, "NL2API: A framework for bootstrapping service recommendation using natural language queries," in *Proc. IEEE Int. Conf. Web Serv.*, 2018, pp. 235–242.
[33] B. Bai, Y. Fan, W. Tan, and J. Zhang, "DLTSR: A deep learning framework for recommendations of long-tail web services," *IEEE Trans. Serv. Comput.*, vol. 13, no. 1, pp. 73–85, Jan./Feb. 2020.
[34] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Collaborative web service QoS prediction via neighborhood integrated matrix factorization," *IEEE Trans. Serv. Comput.*, vol. 6, no. 3, pp. 289–299, Third Quarter, 2013.
[35] L. Qi, Z. Zhou, J. Yu, and Q. Liu, "Data-sparsity tolerant web service recommendation approach based on improved collaborative filtering," *IEICE Trans. Inf. Syst.*, vol. 100-D, no. 9, pp. 2092–2099, 2017.
[36] F. Xie, J. Wang, R. Xiong, N. Zhang, Y. Ma, and K. He, "An integrated service recommendation approach for service-based system development," *Expert Syst. Appl.*, vol. 123, pp. 178–194, 2019.
[37] T. Liang, L. Chen, J. Wu, H. Dong, and A. Bouguettaya, "Meta-path based service recommendation in heterogeneous information networks," in *Proc. 14th Int. Conf. Serv. Oriented Comput.*, Springer, 2016, pp. 371–386.
[38] C. Li, R. Zhang, J. Huai, and H. Sun, "A novel approach for API recommendation in mashup development," in *Proc. IEEE Int. Conf. Web Serv.*, 2014, pp. 289–296.
[39] A. Jain, X. Liu, and Q. Yu, "Aggregating functionality, use history, and popularity of APIs to recommend mashup creation," in *Proc. 13th Int. Conf. Serv. Oriented Comput.*, A. Barros, D. Grigori, N. C. Narendra, and H. K. Dam, Eds., Springer, 2015, pp. 188–202.
[40] P. Samanta and X. Liu, "Recommending services for new mashups through service factors and top-K neighbors," in *Proc. IEEE Int. Conf. Web Serv.*, I. Altintas and S. Chen, Eds., 2017, pp. 381–388.
[41] R. Xiong, J. Wang, N. Zhang, and Y. Ma, "Deep hybrid collaborative filtering for web service recommendation," *Expert Syst. Appl.*, vol. 110, pp. 191–205, 2018.
[42] L. Chen, A. Zheng, Y. Feng, F. Xie, and Z. Zheng, "Software service recommendation base on collaborative filtering neural network model," in *Proc. 16th Int. Conf. Serv. Oriented Comput.*, Springer, 2018, pp. 388–403.
[43] A. Maaradji, H. Hacid, R. Skraba, and A. Vakali, "Social web mashups full completion via frequent sequence mining," in *Proc. IEEE World Congr. Serv.*, 2011, pp. 9–16.
[44] H. Wu, Y. Duan, K. Yue, and L. Zhang, "Mashup-oriented web API recommendation via multi-model fusion and multi-task learning," *IEEE Trans. Serv. Comput.*, vol. 15, no. 6, pp. 3330–3343, Nov./Dec. 2022.
[45] Y. Ma, X. Geng, and J. Wang, "A deep neural network with multiplex interactions for cold-start service recommendation," *IEEE Trans. Eng. Manage.*, vol. 68, no. 1, pp. 105–119, Feb. 2021.
[46] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized Markov chains for next-basket recommendation," in *Proc. 19th Int. Conf. World Wide Web*, 2010, pp. 811–820.
[47] M. Liu, Z. Tu, H. Xu, X. Xu, and Z. Wang, "DySR: A dynamic graph neural network based service bundle recommendation model for mashup creation," *IEEE Trans. Serv. Comput.*, vol. 16, no. 4, pp. 2592–2605, Jul./Aug. 2023.
[48] Y. Tian, D. Krishnan, and P. Isola, "Contrastive multiview coding," in *Proc. 16th Eur. Conf. Comput. Vis.*, 2020, pp. 776–794.
[49] R. D. Hjelm et al., "Learning deep representations by mutual information estimation and maximization," in *Proc. 7th Int. Conf. Learn. Representations*, 2019.
[50] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," in *Proc. 7th Int. Conf. Learn. Representations*, 2019.
[51] F. Sun, J. Hoffmann, V. Verma, and J. Tang, "InfoGraph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization," in *Proc. 8th Int. Conf. Learn. Representations*, 2020.
[52] Z. Peng et al., "Graph representation learning via graphical mutual information maximization," in *Proc. Web Conf.*, Taipei, Taiwan, 2020, pp. 259–270.
[53] W. Hu et al., "Strategies for pre-training graph neural networks," in *Proc. 8th Int. Conf. Learn. Representations*, Addis Ababa, Ethiopia, 2020.
[54] J. Qiu et al., "GCC: Graph contrastive coding for graph neural network pre-training," in *Proc. 26th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2020, pp. 1150–1160.
[55] K. Zhou et al., "S3-Rec: Self-supervised learning for sequential recommendation with mutual information maximization," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, 2020, pp. 1893–1902.
[56] J. Wu et al., "Self-supervised graph learning for recommendation," in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2021, pp. 726–735.

Chunyu Wei received the BS degree in control theory and application from Tsinghua University, China, in 2019. He is currently working toward the PhD degree with the Department of Automation, Tsinghua University. His research interests include services computing, service recommendation, and social computing.

Yushun Fan received the PhD degree in control theory and application from Tsinghua University, China, in 1990. He is currently a professor with the Department of Automation, director of the System Integration Institute, and director of the Networking Manufacturing Laboratory, Tsinghua University. From September 1993 to 1995, he was a visiting scientist, supported by Alexander von Humboldt Stiftung, with the Fraunhofer Institute for Production System and Design Technology (FHG/IPK), Germany. He has authored 10 books and published more than 300 research papers in journals and conferences. His research interests include enterprise modeling methods and optimization analysis, business process reengineering, workflow management, system integration, object-oriented technologies and flexible software systems, petri nets modeling and analysis, and workshop management and control.

Zhixuan Jia is currently working toward the PhD degree with the Department of Automation, Tsinghua University. His research interests include services computing, service recommendation, and spatial-temporal data mining.



Jia Zhang (Senior Member, IEEE) received the BS and MS degrees in computer science from Nanjing University, China, and the PhD degree in computer science from the University of Illinois at Chicago. She is currently the Cruse C. and Marjorie F. Calahan Centennial chair in engineering, professor with the Department of Computer Science, Southern Methodist University. Her research interests emphasize the application of machine learning and information retrieval methods to tackle data science infrastructure problems, with a recent focus on scientific workflows, provenance mining, software discovery, knowledge graph, and interdisciplinary applications of all of these interests in the area of earth science. She has published more than 170 refereed journal papers, book chapters, and conference papers. She is currently an associate editor of *IEEE Transactions on Services Computing* (TSC).