High-Order Social Graph Neural Network for Service Recommendation

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

***Abstract*—Driven by proliferation of the Service-Oriented Architecture (SOA), the quantity of published software services and their users keeps increasing rapidly in the service ecosystem; thus, personalized service selection and recommendation has remained a hot topic. Recent studies have revealed that users' social connections may help better model their potential behaviors. Therefore, in this paper, we study how users' high-order social networks may help improve service recommendation as well as its explainability. Two observations are set forth. First, a user's service preference may be influenced by his trusted users, whom in turn influenced by their social connections. Second, such chained influences will not remain static and equally-weighted, as a user's confidence over his social relations may vary confronted with different targeted services. We thus introduce a novel High-order Social Graph Neural Network (HSGNN) to support social-aware service recommendation. The key idea of the model is a graph convolution-based, multi-hop propagation module devised to extract the high-order social similarity signals from users' local social networks, and encode them into the users' general representations. Afterwards, a neighbor-level attention module is constructed to adaptively select informative neighbors to model the users' specific preference. Extensive experiments in a real-world service dataset show that our HSGNN makes service recommendation more accurately, i.e., by 4.71% in terms of normalized discounted cumulative gain (NDCG), than state-of-the-art baseline methods.**

***Index Terms*—Web services, service recommendation, social network, high-order connectivity.**

I. Introduction

With the wide application of the Web services and Cloud Computing techniques, a burgeoning number of software services have been developed and published into the service ecosystem. Users can remotely leverage reusable services as components to quickly create their value-added new products and fulfill their demands, instead of building everything from scratch. Under such context, service recommendation technique remains to be a key instrument to help users select optimal software services among vast amounts of candidates.

Recent trends in social media and communication technologies have motivated a number of systems to synergistically integrate social features [1], [2]. Representative examples include Epinions, Yelp, Steam and Amazon. Inspired by their success, many traditional service repositories, *e.g., ProgrammableWeb.com*, also allow users to form and join various communities and establish friendships with others. Users on these service platforms usually spread their preference over services to their social connections. Thus, a user's preference can not only be inferred from his historical usage records, but also can be further deduced from his social connections. For instance, on the largest game service platform *Steam*, a user may post comments about some game services on their profile pages, or share game experiences with their friends through instant messages. Such social actions may cause potential effects on the game service selection of their friends. These observations imply that the pervasive use of the social media provides extra information about users, which may be leveraged to support interpretability and predictability of user behaviors [3], while alleviating the data sparsity issue that is a common critical challenge on service recommendation [4].

However, it is not a trivial task to effectively integrate social information to support service recommendation, especially if we want to consider the high-order influence from a user's indirect social neighbors. According to the social correlation theory [5], users' preference may not only be influenced by their first-order local neighbors (direct friends), but also result from their direct friends' social networks (indirect friends). In this article, the term **high-order social relation** refers to such a common phenomenon in most service systems, that is for a user in the social network, besides the preference propagation from the user's first-order neighbors to this user at one hop, the high-order social information propagates recursively to influence the user's behaviors (embedding). Figure 1 shows an example to illustrate how two aspects of the **high-order social relation** may influence a user's preference:

- *High-order social similarity (general preference):* Throughout this article, the term high-order social similarity refers to a common case that a user and the friends of his friends tend to share similar general preference. As illustrated in the left part of Figure 1, user A joins a group favoring sharing and discussing tourism videos on a service platform, which makes him become a social friend of two vloggers B and C. Intuitively users B and C may share some common characters both being

Manuscript received 12 December 2021; revised 6 April 2022; accepted 21 June 2022. Date of publication 30 June 2022; date of current version 31 January 2023. This research has been supported by the National Natural Science Foundation of China (No. 62173199). The associate editor coordinating the review of this article and approving it for publication was Y. Diao. *(Corresponding author: Chunyu Wei.)*

Chunyu Wei and Yushun Fan are with the Beijing National Research Center for Information Science and Technology (BNRist), Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: cy-wei19@mails.tsinghua.edu.cn; fanys@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/TNSM.2022.3186396

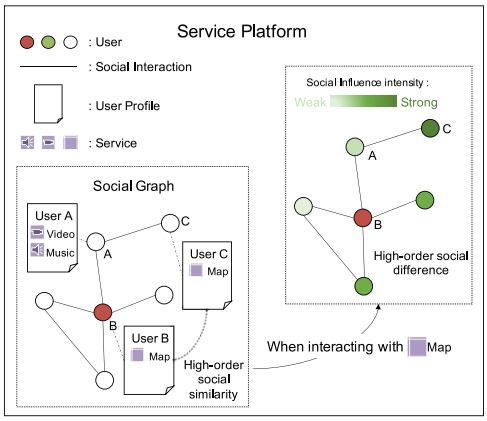

Fig. 1. **Users' high-order social relation.** In a service platform, the left part shows the high-order social similarity among users, in which users B and C show similar general preference between them. The right part describes the high-order social difference. When interacting with Map service, user B's specific preference receives influences of different intensities from his neighbors.

A's friends, which means they may have similar general preference, *e.g., both users like to use map service when travelling.* Such a preference correlation between users B and C cannot be fully reflected by only considering their first-order social relation. This example implies that when learning a user's representation (i.e., preference or embedding), the presence of the similarity between the user and his high-order neighbors (like the similarity between B and C) should be studied to help improve the accuracy of the user's representation construction.

- *High-order social difference (specific preference):* Throughout this article, the term high-order social difference reflects that a user's specific preference over a target service may receive different levels of influences from different users in his social networks. For a specific requirement, different neighbors in the user's social network may contribute different influence intensities. This can be illustrated briefly in the right part of Figure 1, where different neighbors' influence levels on user B are quantified by different grey scales, some stronger while others weaker. If the target is to select a map service, user B tends to act more like those neighbors who love traveling like user C, i.e., influenced more by them. In general, this means that when modeling a user's preference towards some particular service, it is necessary to treat the influence from his neighbors differently, which will help to distill useful preference signals among all the user's neighbors.

These motivating examples reveal that **high-order social relations** among users may be beneficial to improve the quality of service recommendation. To date, however, earlier studies of service recommendation approaches have not dealt with the high-order social relations. In some advanced machine learning fields, several studies have partially investigated the social relations for general recommendation. For example, SocialRec and SocialMF [6], [7] both utilize users' direct social connections to improve the recommendation accuracy. However, they take neither the similarity nor the difference of the high-order social relations into consideration, which may cause the incompleteness of user's representation. Other studies like Deepinf [8] explicitly encode the high-order social similarity, but with the high-order neighbors' weights set equal or relied on predefined static functions, which fail to account for the high-order social difference. Some recent studies like SAMN [9] consider the social difference of direct social friends; however, they do not exploit the high-order neighbors and integrate their features. In short, existing studies of social recommendation have not dealt with both the high-order similarity and the high-order difference simultaneously.

In this article, we propose a novel High-order Social Graph Neural Network (HSGNN) for service recommendation. Utilizing the recent advances in graph convolution networks [10], [11] and neural attention mechanisms [9], [12], [13], HSGNN simultaneously models both the general preference of a user and his specific preference over a given service, by considering the user's high-order social relations. Our prerequisites are that, a user's social network has been constructed over a service ecosystem, and the user-service interaction history has been recorded. HSGNN contains two major components: a social embedding propagation module and a neighbor-level attention module. To our best knowledge, this is the first attempt to study how to exploit deep learning techniques to leverage high-order social similarity and difference synergistically to enhance service recommendation. Meanwhile, our HSGNN model can be applied to other social-aware recommendation applications beyond service recommendation.

Applying Graph Neural Network (GNN), we first build a **social embedding propagation layer**, which refines a user's representation by aggregating the embeddings of his directly connected friends. Then we create the **social embedding propagation module** by stacking multiple propagation layers, which enables us to extract the high-order similarity signals from the user's social network and encode them explicitly into the user's general preference representation. Afterwards, a **neighbor-level attention module** is employed to obtain the user's specific preference on a particular service. Higher weights are placed on his neighbors who share similar preference on that kind of services, and then the weighted sum is used as the specific preference representation for the user. Finally, a **service ranking module** is adopted to apply linear interactions with the service embeddings on both the user's general and specific representations, to derive a ranking score for each service candidate. Over the real-world Steam game service dataset, a series of experiment results showed that our HSGNN model consistently outperforms the state-of-the-art methods in terms of prediction accuracy, and also verified the effectiveness of our designed propagation module and attention module.

Our main contributions are three-fold:

1) We investigate the effect and highlight the noteworthy significance of the **high-order social relations** in common service ecosystems, and classify them into **high-order social similarity** and **high-order social difference**. We expose that the two relations types will

affect a user's general preference and specific preference, respectively.

2) To distill the preference information hidden in the high-order social relations, we propose HSGNN, a novel service recommendation framework, which highlights two core modules: **a multi-hop propagation module** and **a neighbor-level attention module**. The former module explicitly aggregates the high-order social similarity into a user's representation, which enhances the capability of encoding his general preference. The latter module adaptively measures the dynamic social influence strengths in the context of different services, which enriches a user's representation with specific preferences.
3) Through extensive experiments conducted on the real-life Steam game service dataset, we show that HSGNN consistently outperforms the state-of-the-art models and justify the effectiveness of our proposed framework in capturing users' general and specific preferences.

II. Preliminaries

In this section, we will first formally define our problem, then describe how to construct social sub-graphs, followed by depicting a graph batching strategy.

A. Notations and Problem Definitions

We first stretch necessary definitions.

Definition 1 (Service Ecosystem): In a service system, we use $\mathbb{U} = \{u_1, u_2, \ldots, u_M\}$ and $\mathbb{S} = \{s_1, s_2, \ldots, s_N\}$ to denote its comprising set of **users** and the set of **services**, respectively. Recent literature has focused more on optimizing service ranks from implicit data than on predicting explicit service QoS [12], [14], [15], [16]. Most such methods assume that unobserved services are of less interest to users; thus, they are designed to discriminate observed services from unobserved services. In this work, we will follow this trend to record service usage history, yet to predict future service usage probabilities. Formally, we use matrix $\mathbb{T} = [\mathbf{T}_{u,s}]_{M\times N} \in \{0, 1\}$ to indicate the interaction records in the service ecosystem, each cell indicating whether the user $u \in \mathbb{U}$ used the service $s \in \mathbb{S}$ (recorded as 1) or not (as 0).

Definition 2 (Social Graph): Let $\mathbb{G} = (\mathbb{U}, \mathbb{E})$ represents a static **social graph**, where $\mathbb{E} \subseteq (\mathbb{U}, \mathbb{U})$ is the edge set in the graph that denotes the social interactions between the comprising users. Note that we do not differentiate between directed graphs and undirected graphs in this work.

Definition 3 (r-Order Neighbors and Social Sub-Graph): For a user vertex $v \in \mathbb{U}$, its ***r*-order neighbors** are defined as $\mathbf{N}_v^r = \{u : d(u, v) = r\}$ where $d(u, v)$ is the number of edges in a shortest path connecting vertex u and v in the social graph $\mathbb{G}$. As an illustration, z denotes the directly connected social friends of user v on the service platform, i.e., 1-order neighbors. The **social sub-graph** of user vertex v denoted by $\mathbf{G}_v$ is the sub-graph induced by the neighbor set $\mathbf{S}_v^r = \{u : u \in N_v^s \cup v, 0 < s \leq r\}$. Similar to the social graph, the social sub-graph can also be represented as $\mathbf{G}_v = (\mathbf{S}_v^r, \mathbf{A}_{\mathbf{G}_v})$, where $\mathbf{A}_{\mathbf{G}_v}$ denotes the adjacency matrix of $\mathbf{G}_v$.

TABLE I
Notations and Explanations

Notation	Explanation
$\mathbb{U}$	user set
$\mathbb{S}$	service set
$\mathbb{T}$	user-service interaction matrix
$\mathbb{G}$	social graph
$\mathbb{E}$	edge set of the social graph
$\mathbf{N}_v^r$	r-order neighbors of user vertex v
$\mathbf{S}_v^r$	user vertex v's neighbor set considering at most r-order neighbors
$\hat{\mathbf{S}}_v^r$	user vertex v's RWR-based neighbor set considering at most r-order neighbors
$\mathbf{G}_v$	social sub-graph of user vertex v
$\hat{\mathbf{G}}_v$	RWR-based social sub-graph of user vertex v
$\mathbf{A}_{\mathbf{G}_v}$	the adjacency matrix of user vertex v's social sub-graph
$\mathbf{T}_{u,s}$	interaction record of user u and service s
$\hat{\mathbf{Y}}_{u,s}$	matching score of the interaction between user u and service s
$\mathbf{e}_u^{(0)}$	initial embedding of user u
$\mathbf{e}_u^{(l)}$	embedding of user u after l-hop propagation
$\mathbf{e}_s$	embedding of service s
$\mathcal{L}$	the Laplacian matrix of a sub-graph
$\mathbf{W}_{self}, \mathbf{W}_{inter}$	weight matrices of a propagation layer
$\mathbf{W}_{dst}, \mathbf{W}_s, \mathbf{W}_{src}$	weight matrices of the attention module

Problem Formulation: The social relation-based service recommendation problem (interest of user u against service s) can be defined as: given observed interaction records in $\mathbb{T}$ and social graph $\mathbb{G}$, we aim to estimate the score $\hat{\mathbf{Y}}_{u,s}$ of the unobserved interaction $\mathbf{T}_{u,s}$ in $\mathbb{T}$, which is used to rank services.

The notations that we will use throughout the article are summarized in Table I. Next, we will explain how to construct a social sub-graph for each user.

B. Construction of Social Sub-Graph

For a specific user v, to represent his social features, one intuitive way is to extract his social sub-graph G_v. However, for different users, the sizes of their social sub-graphs may vary significantly from each other, which makes it unsuited to the learning process of traditional deep learning methods. A straightforward solution is to perform the Breadth-First-Search (BFS) strategy, starting from the user v to sample a fixed number of neighbors as in our previous work [17]. However, the BFS strategy may lead to an unbalanced distribution of neighbors from different orders in our sampled sub-graphs. In some case, if a user has a lot of directly connected social friends (1-order neighbors), few high-order neighbors will be included in his sampled social sub-graph. In another case, if a user has few close social friends, most vertices in his sampled social sub-graph will be high-order neighbors. To prevent such an imbalance of different order neighbors, some methods like GraphSAGE [10] proposed to sample uniform neighbors of each nodes at different order recursively to form a fixed-size sub-graph. However, since the numbers of neighbors at each iteration grow exponentially, we may not reach the neighbors far from the target user vertex within limited iterations.

To remedy the above issues, we combine the merits of the above methods and perform a random walk with restart

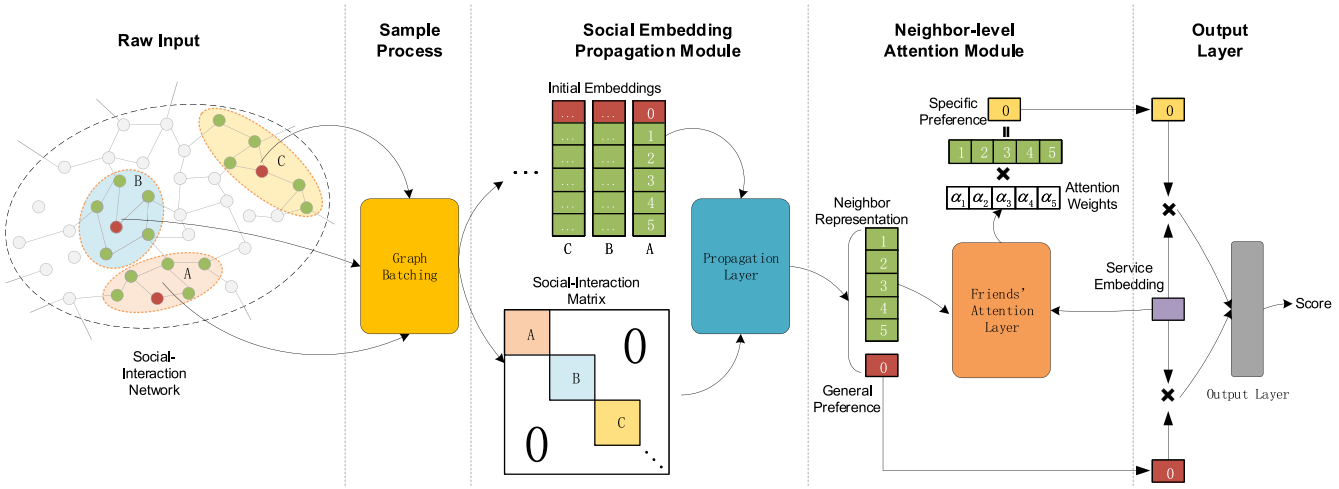

Fig. 2. **Model Framework.** Prior to the calculation part, our model firstly performs random walk with restart (RWR) to sample neighbors from the social network $\mathbb{G}$ to form a social sub-graph for every target user in the batch. After the graph batching operation, we feed the large social graph to the main part comprising two main components: (1) a multi-hop propagation module to enforce the sampled users' embeddings to capture the high-order social similarity; and (2) a neighbor-level attention module to assign non-uniform weights to the users in the social sub-graph.

(RWR) [18] to derive representative sub-graph. We start a random walk on the social graph $\mathbb{G}$ from the user vertex v. The walk iteratively travels to its neighborhood and with a positive probability, the walk returns to the starting vertex v at each step. The RWR runs until it successfully collects a fixed number of vertices denoted by $\hat{\mathbf{S}}_v^r$ with $|\hat{\mathbf{S}}_v^r| = L + 1$, where L is the number of vertices to collect except the target user. The resulted sub-graph $\hat{\mathbf{G}}_v = (\hat{\mathbf{S}}_v^r, \mathbf{A}_{\hat{\mathbf{G}}_v})$ induced by $\hat{\mathbf{S}}_v^r$ can be regarded as an approximation of user v's social sub-graph $\mathbf{G}_v = (\mathbf{S}_v^r, \mathbf{A}_{\mathbf{G}_v})$.

The way how we construct users' social sub-graphs matches their learning patterns from neighbors. Intuitively, a user typically will not check and learn from every of his neighbors, even for his directly connected ones. Meanwhile, social friends may be associated from various types of social relations, and not always share the same preference over objects (i.e., services in our study). Therefore, we do not have to exhaust all neighbors in each hop, like in GraphSAGE [10], not only to control performance but also to avoid from overfitting. Meanwhile, adopting the RWR strategy will favor learning from closer neighbors, which aligns with common practice. Furthermore, applying RWR will allow navigation to high-order neighbors, which aligns with the social correlation theory [5] and the purpose of this study. For each user, after his social sub-graph is constructed, we will build a neural network on top of each such graph and use it for later training and prediction tasks. Therefore, throughout the rest of the article, we will use $\hat{\mathbf{S}}_v^r$ and $\mathbf{S}_v^r$, $\hat{\mathbf{G}}_v$ and $\mathbf{G}_v$, interchangeably.

By implementing the RWR-based sampling strategy, we can easily handle large-scale social graph with millions of users in real world. Wang *et al.* [19] proposed a matrix-form method, without sampling to simultaneously update the representations of all the nodes in a graph. However, it will consume considerable computing resources, since the scale of the social graph increases with a quadratic function of the number of users. Therefore, in this work, we adopt the RWR-based sampling strategy to make our framework runnable and cost-saving on real-world service platforms.

C. Graph Batching

Since our framework needs to sample a social sub-graph for each user to make recommendation, to train neural networks efficiently, a common practice is to batch multiple samples together to form a mini-batch. Thus, how to batch multiple sub-graphs to apply our embedding propagation process is a key issue to accelerate the calculation. Applying the idea from [20], we view a batch of graphs as a larger graph with a collection of disjointed components. The social interaction matrix in Figure 2 shows an example of such a graph batching concept. For a batch of users $\{a, b, c \ldots\}$, after we obtain their social sub-graphs $\{\hat{\mathbf{G}}_a = (\hat{\mathbf{S}}_a^r, \mathbf{A}_{\hat{\mathbf{G}}_a}), \hat{\mathbf{G}}_b = (\hat{\mathbf{S}}_b^r, \mathbf{A}_{\hat{\mathbf{G}}_b}), \hat{\mathbf{G}}_c = (\hat{\mathbf{S}}_c^r, \mathbf{A}_{\hat{\mathbf{G}}_c}), \ldots\}$, we align their adjacency matrices $\{\mathbf{A}_{\hat{\mathbf{G}}_a}, \mathbf{A}_{\hat{\mathbf{G}}_b}, \mathbf{A}_{\hat{\mathbf{G}}_c} \ldots\}$ along the diagonal of a new adjacency matrix (i.e., social-interaction matrix), which enables us to treat it as a mini-batch and perform our method simultaneously.

III. HSGNN Model Framework

In this section, we will first introduce our HSGNN framework, then analyze its learning process including the design of the loss function, followed by discussing the complexity and the generalization of HSGNN.

Figure 2 illustrates the overall architecture of HSGNN that contains two core consecutive modules: a social embedding propagation module, and a neighbor-level attention module. For the input of the HSGNN, we use RWR-based sampling to construct his social sub-graph (See Section II-B) and batch multiple samples together to form a mini-batch (See Section II-C). The social embedding propagation module initializes the user representation and stacks multiple propagation layers to learn the user's general preference encoding the high-order social similarity (See Section III-A). The attention module assigns different weights to the user's high-order social neighbors to learn user's specific preference w.r.t. a particular service (See Section III-B). The output layer of HSGNN fuses the user-service interactions of both general and specific

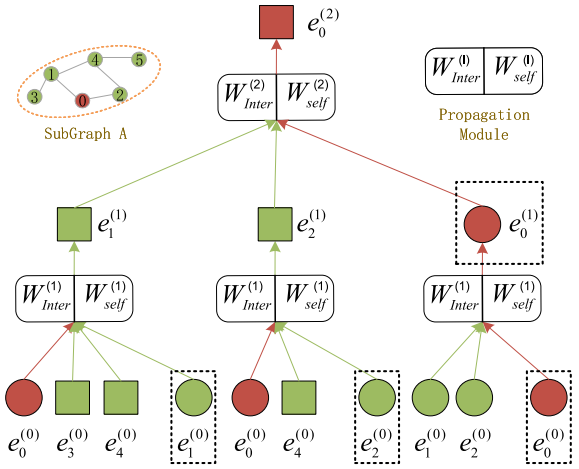

Fig. 3. **Illustration of Embedding Propagation**. Here we show a two-hop propagation process of the social subgraph A in Figure 2. Each node denotes the embedding of the corresponding user. A square node represents its embedding has integrated some of u_0's 2nd-order neighbor signals (u_3 and u_4). After the 1st-hop propagation layer, u_0 and its direct neighbors (u_1 and u_2) all integrate the embeddings of their own 1st-order neighbors into their representations. Consequently after the 2nd-hop propagation layer, u_0 integrates its neighbors' new embeddings and thus indirectly encodes the social similarity from his high-order neighbors (u_3 and u_4). By propagating recurrently, the 1st and the 2nd order social similarity information diffuses to u_0 in $e_0^{(2)}$.

preference to obtain a final matching score for the user-service pair (See Section III-C).

A. Social Embedding Propagation Module

Distributed representation techniques [21] have exhibited great potential in many fields. We encode a user v into a low-dimension latent space with an embedding vector $\mathbf{e}_v \in \mathbb{R}^{d_0}$, where d_0 is the initial embedding size. Initial user features to be encoded may come from the user's carried on features such as his descriptions or characteristics. Besides v, the embedding layer learns the embedding of the service s and we denote it as e_s. For convenience, we use the matrix $\mathbf{E} \in \mathbb{R}^{|\mathbf{S}_v^r| \times d_0}$ to denote the overall initial embeddings of the users in the sampled neighbor set $\mathbf{S}_v^r$:

$$\mathbf{E} = \left[\mathbf{e}_{u_{i0}}, \mathbf{e}_{u_{i1}}, \mathbf{e}_{u_{i2}}, \ldots, \mathbf{e}_{u_{i(|\mathbf{S}_v^r|-1)}}\right], \tag{1}$$

where $\mathbf{e}_{u_{ik}}$ represents the embedding of the user $u_{ik} \in \mathbf{S}_v^r$ and specially u_{i0} denotes user v himself.

According to the social correlation theory [5], a user's general preference is not only influenced by his direct friends, but also affected by his high-order neighbors. We thus devise a multi-hop propagation module to simulate the social information message flow. In the module, a propagation layer simulates how we get social influence from our direct friends, which aggregates the 1-order neighbors' information to update the user's embedding. As this process iterates, the user vertex incrementally gains more and more information from further reaches of the high-order neighbors. Figure 3 shows a two-hop propagation process.

In the following sub-sections, we will first formulate the information propagation of one single user in his social sub-graph. Then we will extend the single-user propagation to a matrix-form, which can operate layer-wise propagation on all users simultaneously over the entire social sub-graph. Finally, the layer-wise propagation will be generalized to multiple stacked layers.

1) Propagation Process of a Single User: For a user u_i and one of his connected friends u_k, we define the social information flow from u_k to u_i as:

$$\mathbf{m}_{u_i \leftarrow u_k} = f(\mathbf{e}_{u_k}, \mathbf{e}_{u_i}, p_{ik}), \tag{2}$$

where p_{ik} is the decay factor of the propagation. Intuitively, the more direct friends u_i or u_k has, the smaller p_{ik} should be. The rationale is that, if u_i has more friends, u_k will have less influence on u_i because u_i will receive more information thus making u_k less important; similarly, if u_k has more friends, u_k may not have much time with u_i thus will have less influence on u_i as well. Note that such a social influence propagation is dependent on both senders and recipients, opposite to the Web page impact propagation in PageRank [22] dependent on senders only. Leveraging the idea of neural graph collaborative filtering [19], we set $f(\cdot)$ as:

$$\mathbf{m}_{u_i \leftarrow u_k} = \frac{1}{\sqrt{|\mathbf{N}_i^1||\mathbf{N}_k^1|}}\left(\mathbf{W}_{self}\mathbf{e}_{u_k} + \mathbf{W}_{inter}(\mathbf{e}_{u_i} \odot \mathbf{e}_{u_k})\right), \tag{3}$$

where $\mathbf{W}_{self}, \mathbf{W}_{inter} \in \mathbb{R}^{d_0 \times d_1}$ are independent weight matrices of this propagation layer, d_1 is the transformation size, and $\frac{1}{\sqrt{|\mathbf{N}_i^1||\mathbf{N}_k^1|}}$ is the coefficient p_{ik}. $|\mathbf{N}_i^1|$ and $|\mathbf{N}_k^1|$ denote the numbers of the first-order neighbors or their "degrees" in the social sub-graphs in other words. We use $\mathbf{e}_{u_i} \odot \mathbf{e}_{u_k}$ to represent the interaction between both u_i and u_k, where $\odot$ denotes the element-wise product. This ensures that the connected friends with higher similarity to user u_i will pass more information to u_i.

Aggregating the social information flows from user u_i's all connected friends, we can update the representation of u_i after one round of propagation as:

$$\mathbf{e}_{u_i}^{(1)} = ReLU\left(\mathbf{m}_{u_i \leftarrow u_i} + \sum_{u_k \in \mathbf{N}_i^1} \mathbf{m}_{u_i \leftarrow u_k}\right), \tag{4}$$

where $\mathbf{e}_{u_i}^{(1)}$ denotes the representation of user u_i after the first embedding propagation layer and *ReLU* is a common nonlinear activation function. Note that in order to keep the original information of u_i, we insert the term $\mathbf{m}_{u_i \leftarrow u_i}$ in the function, which can be represented as:

$$\mathbf{m}_{u_i \leftarrow u_i} = \mathbf{W}_{self}\mathbf{e}_{u_i}. \tag{5}$$

Here $\mathbf{W}_{self} \in \mathbb{R}^{d_0 \times d_1}$ shares the same values as the weight matrix in Equation (3).

2) Matrix Form of the Propagation Process: To carry out the propagation process for all users in the same social sub-graph, we transform Equations (3), (4), and (5) into a matrix form as:

$$\mathbf{E}^{(1)} = ReLU\Big((\mathcal{L}+\mathbf{I})\mathbf{E}\mathbf{W}_{self}^{(0)} + \mathcal{L}\mathbf{E} \odot \mathbf{E}\mathbf{W}_{inter}^{(0)}\Big) \quad (6)$$

where $\mathbf{E}^{(1)}$ is the new embedding matrix after the first propagation, and $\mathcal{L}$ is the Laplacian matrix of the sub-graph $\mathbf{G}_v$. Formally, suppose $\mathbf{G}_v$ has adjacency matrix $\mathbf{A}$ and diagonal degree matrix $\mathbf{D}$, the Laplacian matrix $\mathcal{L}$ can be calculated as follows:

$$\mathcal{L} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}. \quad (7)$$

Each of $\mathbf{D}$'s diagonal elements $\mathbf{D}_{ii} = |\mathbf{N_i^1}|$ represents the degree of the user vertex u_i in the social sub-graph. We can deduce that the diagonal element $\mathcal{L}_{ii} = 0$ and the off-diagonal element $\mathcal{L}_{ik} = \frac{1}{\sqrt{|\mathbf{N_i^1}||\mathbf{N_k^1}|}}$.

The matrix form of the propagation process helps us not only update all the user representations in the same sub-graph simultaneously, but also facilitate the batch calculation. Furthermore, the matrix form makes it easy to stack multiple propagation processes as a united module to extract the high-order social signals in the user's social sub-graph, which will be discussed in the next section.

3) Multi-Hop Propagation: Stacking multiple propagation layers can explore the high-order social similarity information. In the l-th step of propagation, Equation (6) can be reformulated as:

$$\mathbf{E}^{(l)} = ReLU\Big((\mathcal{L}+\mathbf{I})\mathbf{E}^{(l-1)}\mathbf{W}_{self}^{(l)} + \mathcal{L}\mathbf{E}^{(l-1)} \odot \mathbf{E}^{(l-1)}\mathbf{W}_{inter}^{(l)}\Big), \quad (8)$$

where $\mathbf{E}^{(l)} \in \mathbb{R}^{(L+1)\times d_l}$ is the representation of the users in $\mathbf{S}_i^r$ after l times of propagation steps, and $\mathbf{E}^{(l-1)}$ is the representation from the previous step. $\mathbf{W}_{self}^{(l)}, \mathbf{W}_{inter}^{(l)} \in \mathbb{R}^{d_{l-1}\times d_l}$ are the transformation matrices for the l-th step of propagation. Different weight matrices are assigned for the transformation in each step, so that the size of $\mathbf{E}^{(l)}$ is updated after every propagation. $\mathbf{E}^{(0)}$ is set as $\mathbf{E}$ in Equation (1).

After iterating l times, each user vertex v in the social sub-graph aggregates information from its neighbor set $\mathbf{S}_v^l$ with the farthest reach of his l-order neighbors $\mathbf{N}_v^l$. Hence, the user's final embedding (z_v) explicitly encodes the high-order social information, which can be treated as the representation of the user's general preference towards services. In a matrix form, the general preference of all the users in $\mathbf{S}_v^r$ can be represented as:

$$\mathbf{Z} = [\mathbf{z}_{u_{i0}}, \mathbf{z}_{u_{i1}}, \mathbf{z}_{u_{i2}}, \ldots, \mathbf{z}_{u_{iL}}] = \mathbf{E}^{(l)}. \quad (9)$$

The pseudo algorithm framework of our social embedding propagation module is shown in Algorithm 1.

Figure 3 shows a portion of a computation graph constructed for one user (u_0) based on his social sub-graph, showing only two hops for the interest of space. For each user, such a neural network will be constructed. All such small neural networks will be trained simultaneously, instead of training a huge social network. Note that $\mathbf{W}_{self}^l$ and $\mathbf{W}_{inter}^l$ in each layer of the constructed computation graph will be shared by all neural networks for all nodes.

Algorithm 1 Social Embedding Propagation Module

Input: social sub-graph $\mathbf{G}_v(\mathbf{S}_v^r, \mathbf{A}_{\mathbf{G}_v})$;
Initial Embedding matrix $\mathbf{E}^{(0)}$; hop time l;
self-weight matrices $\mathbf{W}_{self}^k, \forall k \in \{1, \ldots l\}$; inter-weight matrices $\mathbf{W}_{inter}^k, \forall k \in \{1, \ldots l\}$.
Output: General preference representation $\mathbf{Z} = \{\mathbf{z}_v, \forall v \in \mathbf{S}_v^r\}$.

$\mathbf{D} \leftarrow Calculate\_degree\_matrix(\mathbf{A}_{\mathbf{G}_v})$;
$\mathcal{L} \leftarrow \mathbf{D}^{-\frac{1}{2}}\mathbf{A}_{\mathbf{G}_v}\mathbf{D}^{-\frac{1}{2}}$;
for $k = 1, \ldots l$ **do**
  $\mathbf{E}^{(k)} \leftarrow ReLU((\mathcal{L}+\mathbf{I})\mathbf{E}^{(k-1)}\mathbf{W}_{self}^{(k)} + \mathcal{L}\mathbf{E}^{(k-1)} \odot \mathbf{E}^{(k-1)}\mathbf{W}_{inter}^{(k)})$;
end for
$[\mathbf{e}_v^{(l)}, \ldots] \leftarrow \mathbf{E}^{(l)}, \forall v \in \mathbf{S}_v^r$;
$\mathbf{z}_v \leftarrow \mathbf{e}_v^{(l)}, \forall v \in \mathbf{S}_v^r$;
Return $\mathbf{Z}$

B. Neighbor-Level Attention Mechanism

As illustrated in the right part of Figure 1, if one of user v's friends has more expertise or experience over a service or similar services, his opinions will be more convincing thus may have higher influence on the user v's choice on the service. The main purpose of neighbor-level attention is to assign different importance to a user v's sampled neighbors $\mathbf{S}_v^r$, when the user is considering a specific service s and thus help model the user's specific preference over the service. In order to simulate such interactions among the user ($\mathbf{z}_v$), his neighbor ($\mathbf{z}_j$) and the service ($\mathbf{e}_s$), we first project all of their embeddings into a shared latent space. Note that the embeddings of the user and his neighbors result from multi-hop propagation layers as described in earlier sections. The embedding of a service can be calculated from its inherent features, such as the methods leveraging its functionality [23], non-functional features (i.e., QoS) [24], and service descriptions [25] etc. Then a fully-connected layer is applied to compute an attention score $\alpha_{(j)}^*$.

$$\alpha_{(j)}^* = \mathbf{h}^T ReLU\big(\mathbf{W}_{dst}\mathbf{z}_v + \mathbf{W}_s\mathbf{e}_s + \mathbf{W}_{src}\mathbf{z}_j\big), \quad (10)$$

where $\mathbf{W}_{dst} \in \mathbb{R}^{d_l\times d_k}$, $\mathbf{W}_s \in \mathbb{R}^{d_l\times d_k}$, $\mathbf{W}_{src} \in \mathbb{R}^{d_l\times d_k}$, $\mathbf{h} \in \mathbb{R}^{d_k}$ are parameters. d_l represents the final embedding size of both users and services, and k denotes the dimensions of the attention network.

Afterwards, we normalize the neighbor-level attention scores with a softmax function, which can make the attention network a probabilistic interpretation:

$$\alpha_{(j)} = \frac{\exp\Big(\alpha_{(j)}^*\Big)}{\sum_{i\in\{\mathbf{S}_v^r\setminus v\}}\exp\Big(\alpha_{(i)}^*\Big)}. \quad (11)$$

After we obtain the attention weight of each neighbor, the final representation of the user's specific preference towards

Algorithm 2 Neighbor-Level Attention Mechanism

Input: General preference set $\mathbf{Z}$;
service embedding $\mathbf{e}_s, \forall s \in \mathbb{S}$;
weight matrices $\mathbf{W}_{dst}$, $\mathbf{W}_s$, $\mathbf{W}_{src}$; parameter $\mathbf{h}$;
Output: Specific preference representation $\mathbf{y}_v$

1: **for** neighbor j in $\{\mathbf{S}_v^r \backslash v\}$ **do**
2: $\quad \alpha^*_{(j)} \leftarrow \mathbf{h}^T ReLU(\mathbf{W}_1\mathbf{z}_v + \mathbf{W}_2\mathbf{e}_s + \mathbf{W}_3\mathbf{z}_j)$
3: $\quad \alpha_{(j)} \leftarrow Softmax(\alpha^*_{(j)})$
4: **end for**
5: $\mathbf{y}_v \leftarrow \sum_{i \in \{\mathbf{S}_v^r \backslash v\}} \alpha_{(i)} \mathbf{z}_i$
6: Return $\mathbf{y}_v$

the specific service is as follows:

$$\mathbf{y}_v = \sum_{k \in \{\mathbf{S}_v^r \backslash v\}} \alpha_{(k)} \mathbf{z}_k, \tag{12}$$

which is the weighted sum of the general preference of the user's neighbors in the social sub-graph.

The pseudo code of the neighbor-level attention mechanism is shown in Algorithm 2.

C. Service Prediction

After obtaining the representation of user v's general preference $\mathbf{z}_v$ and specific preference $\mathbf{y}_v$, one question arises: how can we fuse both of them under our HSGNN framework, so that they can mutually reinforce each other to better model the user's preference influenced by his high-order social relations?

We first apply element-wise product to calculate the interaction feedback of the above two preferences over the target service s, following [26], [27]:

$$\phi_z = \mathbf{z}_v \odot \mathbf{e}_s, \ \ \phi_y = \mathbf{y}_v \odot \mathbf{e}_s. \tag{13}$$

Then we combine their concatenating vectors to the output layer, as shown in Figure 2:

$$\hat{\mathbf{Y}}_{v,s} = \mathbf{h}_{out}^T \begin{bmatrix} \phi_z \\ \phi_y \end{bmatrix}, \tag{14}$$

where $\mathbf{h}_{out}$ is the edge weights of the output layer and $\hat{\mathbf{Y}}_{v,s}$ is the predicted score for the unused service s. We will calculate such predicted scores for all unused services, which will then be ranked in descending order to provide a Top-K service recommendation list.

D. Parameter Learning

In this section, we will carefully analyze the parameter training aspects regarding the optimization and training efficiency of HSGNN.

1) Optimization: The core of the optimization process is to learn the relevance-based ranking position of services. To this end, we optimize the pairwise Bayesian Personalized Ranking (BPR) loss [28], which has been extensively used in most implicit recommendation processes. BPR assumes that consumed services should be assigned higher prediction values than unobserved ones, since they are more reflective of a user's preference. For each positive user-service pair $< u_i, s_j >$, we randomly sample multiple negative services from the unobserved services of the user, which is denoted as s_k. The **BPR** pairwise ranking loss is defined as follows:

$$L_{BPR} = \sum_{(u_i, s_j, s_k) \in \mathfrak{D}} -ln\sigma\left(\hat{\mathbf{Y}}_{u_i,s_j} - \hat{\mathbf{Y}}_{u_i,s_k}\right) + \lambda ||\Theta||_2^2 \tag{15}$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$ is the logistic sigmoid function and $\mathfrak{D}$ represents the set of pairwise training instances. The second term of equation (15) is a regularization term, which is a L_2 norm to prevent overfitting. The pseudo code of the whole algorithm framework is shown in Algorithm 3.

Algorithm 3 The Overall Learning Algorithm of HSGNN

Input: Interaction matrix $\mathbb{R}$; social graph $\mathbb{G}$; hop time l;
Output: Parameter set Θ;

1: Initialize model parameter set Θ with small random values;
2: **while** not converged **do**
3: $\quad$ **for** $< u_i, s_j, s_k >$ in the pairwise training set $\mathfrak{D}$ **do**
4: $\quad\quad$ Compute general preference $\mathbf{Z}$; (Algorithm 1)
5: $\quad\quad$ Compute specific preference $\mathbf{y}_{u_{i0}}$; (Algorithm 2)
6: $\quad\quad$ $\phi_{z_j} \leftarrow \mathbf{z}_{u_i} \odot \mathbf{e}_{s_j}; \phi_{y_j} \leftarrow \mathbf{y}_{u_i} \odot \mathbf{e}_{s_j}$
$\quad\quad$ $\phi_{z_k} \leftarrow \mathbf{z}_{u_i} \odot \mathbf{e}_{s_k}; \phi_{y_k} \leftarrow \mathbf{y}_{u_i} \odot \mathbf{e}_{s_k}$
7: $\quad\quad$ $\hat{\mathbf{Y}}_{u_i,s_j} \leftarrow \mathbf{h}_{out}^T \begin{bmatrix} \phi_{z_j} \\ \phi_{y_j} \end{bmatrix}; \hat{\mathbf{Y}}_{u_i,s_k} \leftarrow \mathbf{h}_{out}^T \begin{bmatrix} \phi_{z_k} \\ \phi_{y_k} \end{bmatrix}$
8: $\quad$ **end for**
9: $\quad$ Calculate BPR loss L; (Equation. 15)
10: $\quad$ **for** Each parameter θ in Θ **do**
11: $\quad\quad$ Update $\theta = \theta - \eta \frac{\partial L}{\partial \theta}$
12: $\quad$ **end for**
13: **end while**
14: Return parameter set Θ

2) Complexity: For our proposed learning algorithm, we consider its space complexity and time complexity.

Space Complexity: As shown in Algorithm 3, our HSGNN framework contains two sets of parameters: the initial embeddings of both users and services $\Theta_1 = [\mathbf{E}, \mathbf{S}]$ and the parameter set $\Theta_2 = [\mathbf{W}^k_{self}, \mathbf{W}^k_{inter} (\forall k \in \{1, \ldots, l\}), \mathbf{W}_{dst}, \mathbf{W}_s, \mathbf{W}_{src}, \mathbf{h}]$. Since Θ_1 is mainly based on user and service embeddings, which all need space to store the representation of users and services, the space complexity of parameter Θ_1 is the same as that of most baseline models and it grows linearly with the numbers of users and services. As for the parameters in Θ_2, they are shared among all the users and services with very low dimensions compared with the embedding matrices in Θ_1. Thus the space complexity of Θ_2 can be treated as a constant and neglected, which makes the space cost of our framework nearly as much as the classical embedding models (e.g., BPR [28], NCF [29]) and significantly lower than some memory-based methods (e.g., SAMN [9]).

Time Complexity: Since the BPR loss function for implicit feedback has been adopted by most of the baseline methods, we only need to compare the time cost of the social embedding propagation module and the neighbor-level attention mechanism in our framework. The embedding propagation process costs $O(Mln)$, where M is the number of users, l represents

TABLE II
STATISTICS OF STEAM DATASETS

Item	Number
Users	9,392
Services	2,368
Invocations	341,594
Density(Invocations)	1.536%
Social Connections	19,752
Density(Social Connections)	0.022%

the propagation layers and n denotes the fixed sample neighbor numbers. Actually, in Section IV, our experimental results show that with $l = 3$, our model consistently shows better performance than other cases. Furthermore, the sample neighbor numbers per user $n \ll M$ is relatively small. Thus, the time complexity $O(Mln)$ of propagation module is acceptable. From Algorithm 2, we know that the time complexity of the attention mechanism is $O(Mn)$. Therefore, the overall time complexity is $O(Mln)$, which is computationally feasible in practice and thus able to support real-time query in real-world service recommendation systems.

IV. EXPERIMENTS

A. Experimental Settings

1) Dataset Description: Steam is a platform providing game and software services, which includes services from over 1,200 publishers and over 75 million active users. In addition to its service providing function, Steam provides numerous social networking features such as profile pages, friends, groups, instant messaging, voice chat, and news feeds. According to [30] and [31], the game services show a high level of similarity to Web services with three unique features:

- The friendship connections are sparse compared to other social networks;
- There exist long tail behaviors in their distributions;
- Both game services and Web services are massive and heterogeneous.

Based on their common characteristics, it is reasonable to use the Steam platform to test and verify our proposed HSGNN framework on service recommendation.

To test and verify our HSGNN, we randomly crawled 9,392 Steam user accounts, along with their friends lists and 341,594 service usage records, as well as a total of 2,368 game services to construct a real-world dataset. The associated social network is built according to the method described in Section II-B. Table II summarizes the numerical properties of the dataset.

For each user in the dataset, we left his latest two invocations for validation and test, and utilized the remaining data for training. To evaluate the results more efficiently, we randomly sampled 99 services that have no interaction with the target user as hard negative samples, and ranked the validation and test services with respect to these 99 services. For each observed user-service interaction in the training set, we conducted the negative sampling strategy to pair it with 19 negative services that the user did not consume before. Such an evaluation scheme has been widely used in many other works on recommendation evaluation [9], [19], [32].

2) Evaluation Metrics: To evaluate the performance of all algorithms, we adopted two popular metrics, namely Normalized Discounted Cumulative Gain (NDCG) and Hit Ratio (HR). The NDCG@K metric accounts for the position of the hits by assigning higher scores to hits at top ranks and thus is position-aware. The HR@K metric measures whether the test item is present on the recommendation list or not. Both of the adopted metrics can be formulated as follows:

$$NDCG@K = \frac{1}{R_N} \sum_{i=1}^{N} \frac{2^{rel_i - 1}}{\log_2(1+i)} \tag{16}$$

$$HR@K = \frac{\sum_{i=1}^{K} rel_i}{|y_u^{test}|} \tag{17}$$

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 or not, and the R_N term indicates the maximum possible cumulative component through ideal ranking. $|y_u^{test}|$ is the number of services used by the user u in the test set.

3) Baselines: To evaluate the performance of the Top-K recommendation, we compared our HSGNN with the following five baseline methods. All models are learned by optimizing the same pairwise ranking loss of BPR defined in Equation (15) for a fair comparison, which means some methods are adjusted to evaluate on implicit feedback.

- **BPR** [28]: This method optimizes the matrix factorization (MF) model with the BPR objective function for implicit feedback-based service recommendation.
- **Sorec** [6]: This is a social recommendation method performing co-factorization on the user-item rating matrix and user-user social relations matrix.
- **SocialMF** [7]: This is a classical model considering the trust information and its propagation into the matrix factorization model for recommender systems.
- **NCF** [29]: This method is a state-of-the-art deep learning based framework combining matrix factorization (MF) with a multilayer perceptron model.
- **SAMN** [9]: This is a state-of-the-art deep learning method, which unifies the strengths of memory networks and attention mechanisms to address the problems in social-aware recommendations.

Our method aims to model the social relationship in the service recommendation. However, few research has explored the effect of the social influence in service recommendation. Thus we adjusted the above general social recommendation methods to compare with our HSGNN on the Steam game service dataset.

4) Experiment Implementation: We implemented HSGNN on the basis of Pytorch [33], a widely used Python library for neural networks. During the tuning process, we found that 0.005 can be a good initial learning rate with an embedding size of 64, respectively. Borrowing the idea of autoencoder, the transformation size sequence should be non-increasing. By shrinking the transformation size, each propagation process can learn more abstract features. Empirically, we halved the transformation size for each successive propagation layer.

TABLE III
OVERALL PERFORMANCE COMPARISON

	NDCG@10	HR@10	NDCG@20	HR@20
SocialMF	0.4311	0.6291	0.4581	0.7105
BPR	0.4029	0.5855	0.4269	0.6739
SocialRec	0.4248	0.6021	0.4502	0.6966
NCF	0.4286	0.6204	0.4546	0.7220
SAMN	0.4392	0.6460	0.4626	0.7366
HSGNN	**0.4599**	**0.6584**	**0.4830**	**0.7494**

To maintain the space consistency of the representations of both users and services, we also project a service s into a low dimension latent space with an embedding vector e_s, which makes it easy to calculate the interaction between users and services. There are many embedding methods that can be utilized for the service representation learning in our work. For example, some methods learn service embeddings based on their functionality [23] or their non-functional features (i.e., QoS) [24]. Other methods like [25] analyze service descriptions (i.e., WSDL file) to obtain the service embeddings. Since the service embedding methods are not the focus of this article, we only use the ID embedding of the services in our experiments to illustrate the effectiveness of our approach.

B. Comparative Analysis on Overall Performance

The empirical results of our HSGNN and the baselines on the Steam game service dataset are summarized in Table III, which shows the NDCG@K and HR@K with recommendation size $K = 10, 20$. We conducted one-sample t-tests and *p-value* < 0.05 indicates that the improvements of HSGNN over the strongest baseline are statistically significant. From the results, we drew three conclusions:

First, the methods leveraging social network information perform better than the ones without it. For example, in Table III for most metrics, SocialRec shows a better performance than BPR, while the state-of-the-art SAMN, SocialMF and our HSGNN all outperform BPR and NCF. The experiment results provide trustworthy evidence for introducing the social information into service recommendation.

Second, we noticed that the methods assigning different weights to different neighbors bear better performances than those do not. Compared to SocialRec and SocialMF, the performance of SAMN and our HSGNN proves that the attention mechanisms on the friend level improve the user representation learning. It might because the influence strengths of a user's friends are actually different and dynamic when interacting with different services.

Third, our HSGNN consistently yields the best performance on the Steam dataset, which improves over the strongest baseline SAMN by 4.71% with respect to NDCG@10 and 1.74% to HR@20, respectively. In our model, we stack multiple propagation layers to extract the high-order social similarity in an explicit way, while the state-of-the-art SAMN only considers directed friends. This result strongly supports that the high-order social similarity among users can improve the user representation learning in service ecosystems.

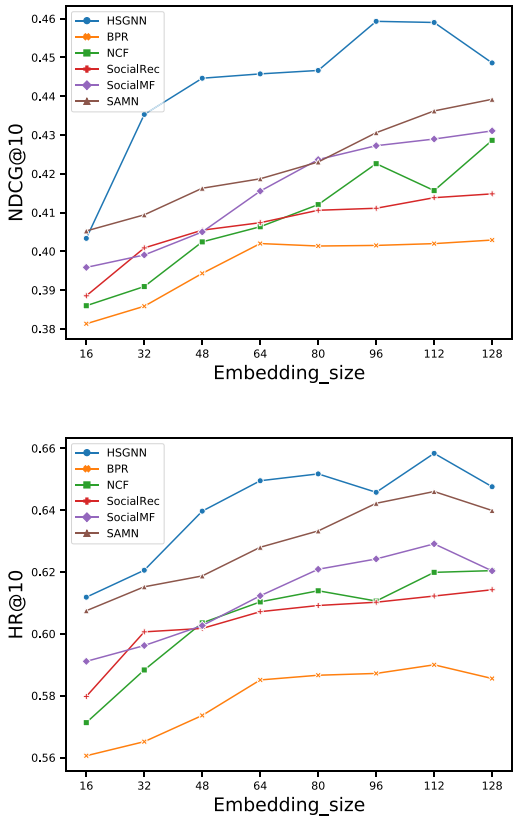

Fig. 4. Performance comparison on test set w.r.t embedding sizes.

C. Study of Embedding Propagation

As the embedding propagation module plays a pivotal role in our HSGNN, we specially investigated its impact on the recommendation performance. We started by exploring the influence of different embedding sizes. We then studied how the size of a social sub-graph affects the performance. Furthermore, we analyzed the influence of the hop time (i.e., the number of the propagation layers).

1) Effect of Embedding Size: We conducted experiments to test the influence of the latent factor size d. The results are shown in Figure 4, which contains the results on HR@10 and NDCG@10 metrics, respectively. The result of HR@10 is similar to that of NDCG@10, which shows our method outperforms all the other models under different values of embedding size. Moreover, the performance of all the models increase, when the embedding size d increases. This means that a larger dimension of the latent factors can encompass more information about both users and services and thus increase the model capability.

2) Effect of Sub-Graph Size: Since we perform the random walk with restart (RWR) strategy to construct a social sub-graph, the sampled neighbor number n can not only control the size of the sample sub-graph, but also can directly determine the range of the social influence of each order ($|\mathbf{N}_v^r|, r = 1, 2, \ldots$). Figure 5 shows the performance (in terms of NDCG and HR) by varying the sampled neighbor number from 10 to 100. We can observe a slow increase of prediction performance when we sample more close-by neighbors. This

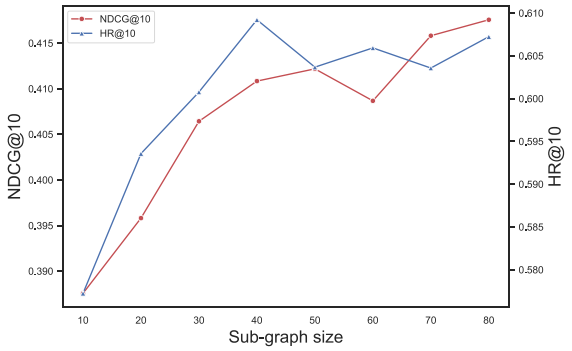

Fig. 5. Sub-graph size sensitivity of HSGNN.

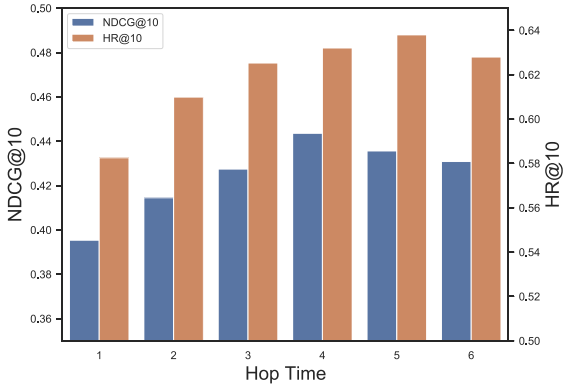

Fig. 6. Performance of HSGNN under different hop time l.

is not surprising, because we catch more information as the size of the sampled network increases.

3) Effect of Hop Time: The multi-hop propagation module is one of the centrepieces in our HSGNN, so we varied the model depth l (i.e., hop time) to examine how HSGNN may benefit from multiple propagation layers. The experiment results are shown in Figure 6 and next we will use a term HSGNN-# to indicate the HSGNN method with # propagation layers. From Figure 6, we can have the following two observations:

First, the general trend shows that the increase of the hop time can improve the service recommendation performance. From Figure 6, we can see that our HSGNN with more than one single propagation layer consistently exhibits better performance than HSGNN-1. Combining the aforementioned analysis in Section IV-B, we attribute this improvement to the increased level of the ability to extract the high-order social similarity.

Second, when we added more propagation layers on HSGNN-3, the NDCG@10 of the model started to decrease, which might be caused in two aspects. On the one hand, too deep architecture introduces noises into the learning process. On the other hand, the neighbors far away from the user have little influence on the user. Thus the marginal improvement brought by the far neighbors is insufficient to offset the noises.

Based on the above observation, we try two techniques commonly used in the deep learning to enable a deeper architecture and stabilize the training process, which can help the model to extract more information from further reaches of the neighbors. They are as follows:

1) **HSGNN-Dropoup**: We randomly dropout 50% of the neighbors in the user's social sub-graph at each propagation layer.
2) **HSGNN-Skip**: we use skip connections to add the initial $\mathbf{E}$ and the $\mathbf{E}^{(l)}$ from the last layer together as the output $\mathbf{Z}$, that is $\mathbf{Z} = \mathbf{E} + \mathbf{E}^{(l)}$.
3) **HSGNN-Dropout&Skip**: We train HSGNN with both Dropoup and skip connections.

We show the performance of HSGNN and the above variants in Table IV. Both the dropout and the skip connection can stabilize the training process and improve the performance when we stacking more than 3 layers. Specifically, when we adopt both tricks simultaneously, the model achieves the best performance and the shows the best robustness. In our game service dataset, three or four hops provide sufficient capacity to model the complex general preference of users and further neighbors have very weak influence to the target user. However, we believe in other service scenarios like video streaming or mobile app stores, the information from higher-order neighbors may be valuable and these techniques provide our model with more scalability.

TABLE IV
NDCG@10 OF HSGNN WITH DIFFERENT TRICKS

	4 Layers	5 Layers	6 Layers
HSGNN	0.4436	0.4356	0.4309
HSGNN-Dropoup	0.4524	0.4487	0.4481
HSGNN-Skip	0.4591	0.4552	0.4524
HSGNN-Dropout&Skip	0.4599	0.4604	0.4593

TABLE V
COMPARISON OF THE VARIANT MODELS OF HSGNN

Varients	Representation of user's specific preference	Social information	Social difference
HSGNN-S	0	\	\
HSGNN-A	$\mathbf{y}_{u_i} = \sum_{v \in \{\mathbf{S}^r_{u_i}\}} \frac{1}{\lvert\mathbf{S}^r_{u_i}\rvert} \mathbf{z}_v$	✓	\
HSGNN	$\mathbf{y}_{u_i} = \sum_{v \in \{\mathbf{S}^r_{u_i}\}} \alpha_{(v)} \mathbf{z}_v$	✓	✓

D. Study of Neighbor-Level Attention

One of the key components in the proposed HSGNN is the neighbor-level attention module. Thus, we also conducted experiments to analyze its performance.

1) Effect of the Neighbor-Level Attention: To evaluate the overall effect of our neighbor-level attention mechanism, we compared our model with the following variants of HSGNN:

1) **HSGNN-S**: A variant model of HSGNN without considering the specific preference of the user himself.
2) **HSGNN-A**: A variant model of HSGNN in which the weights of the neighbors in Equation 12 are set equal.

The characteristics of the variant models are listed in Table V. The main difference of them is about the representation of the user's specific representation $\mathbf{y}_{u_i}$. $\frac{1}{\lvert\mathbf{S}^r_{u_i}\rvert - 1}$ is the number of user u_i's social neighbors, $\mathbf{z}_v$ is the general

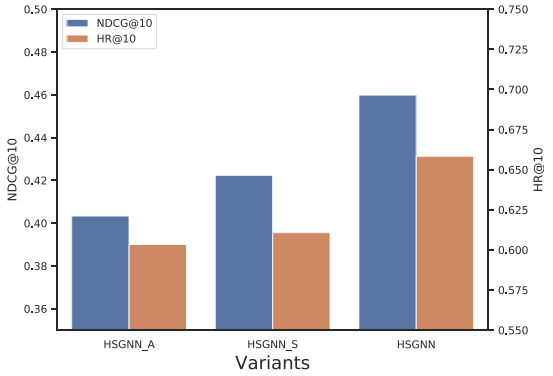

Fig. 7. Performance of variants of HSGNN.

preference vector generated by the social embedding propagation module (cf. Section III-A3) and $\alpha_{(v)}$ is the neighbor-level attention score (cf. Equation 11).

Figure 7 shows the results of HSGNN and its two variants. For the interest of space, here we show only the results of the NDCG@10 and HR@10 on the Steam dataset. From Figure 7, two observations can be made.

First, when the neighbor-level attention mechanism is applied, the performance achieves significant improvement compared to HSGNN-S and the constant-weight method HSGNN-A. This may because the user representation from the propagation module only explicitly encodes the user's general preference without considering the user's specific preference towards the target service. The result also shows that the attention mechanism can learn adaptive weights to make sure that when interacting with different services, the neighbors who share common preference can be more prominent in the user's final representation.

Second, HSGNN-A performs the worst among the variant models since it does not consider the high-order social difference. Even worse, by setting all weights equal fails to get the correct representation of the user's specific preference, which disturbs the calculation of the user's final score towards a given service.

2) Visualization on Attention Mechanism: The attention weights of the neighbors in the social sub-graph reflect how our proposed model learns and recommends according to the user's specific preference. We conducted a visualization experiment to verify the effectiveness of the attention module. We randomly chose a user along with his social sub-graph containing 29 neighbors to see their contributions in the user's specific preference, when the user is interacting with two random services (#50 and #380). Then we visualized the attention weights of different neighbors over these two services, which are shown in Figure 8, respectively.

We can have the following three observations. First, the attention weights of a user's high-order neighbors are dynamic when interacting with different services. For example, when we learn the user representation over service #50, we find the intensity of neighbor #4 in Figure 8(a) is relatively higher compared with that over service #380 in Figure 8(b). Second, the attention weights of neighbors may vary significantly. For example, when interacting with service #50 as shown in

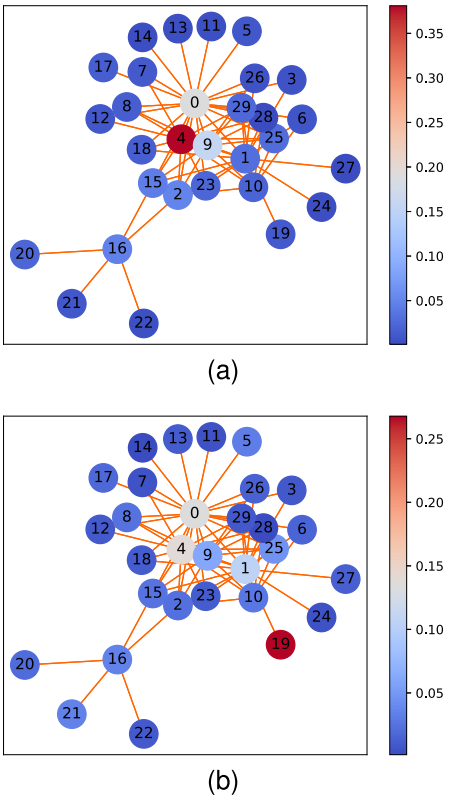

Fig. 8. **Neighbor-level Attention distributions of a social sub-graph**. The color scale indicates the intensities of the influence weights, where a lighter color means higher score and a darker color means a lower score. Note that Vertex 0 is the target user. (a) Some user interacting with service #50. (b) Some User interacting with service #380.

Figure 8(a), the user gives more attention to neighbor #4. The reason of both of the observations may be that when a neighbor has consumed some service, he will have a larger influence strength for the user over this service. Third, the neighbors with more invocation histories tend to have higher attention weights. For example, when interacting with service #50 as shown in Figure 8(a), both neighbors #4 and #9 have consumed service #50, but the attention weight of neighbor #4 is much higher than that of neighbor #9. This may be because the attention weight reflects the richness of a neighbor's feedback information, that is his experience level. In other words, neighbor #9 consumed much fewer services than neighbor #4 and thus cannot provide significant influence. This experiment shows that our attention mechanism may help to explain a recommendation result.

V. Related Work

Our study is closely related to literature on service recommendation and graph representation learning.

A. Service Recommendation

With the explosive development of service ecosystems, service recommendation utilizing Big Data technologies remains to be one of the key research topics in the field of Services Management, since Big data are widely recognized as being one of the most powerful drivers to promote productivity,

improve efficiency, and support innovation [34], [35]. Here we review the related work on service recommendation in terms of the following three categories.

Semantics-Aware Recommendation: Semantics-aware methods mainly focus on information retrieval and similarity calculation. Existing methods usually extract semantic information, e.g., keywords and labels, and calculate relevance scores represented by semantic distance. Dong *et al.* [36] represented users and services as vectors of words and calculated the cosine similarity between corresponding vectors. Meng *et al.* [37] represented a user's preference by keywords and proposed a user-based collaborative filtering (CF) algorithm for service recommendation. To alleviate the dependence on the quality of domain thesaurus, Li *et al.* [38] revealed a correlation between services and words extracted from related WSDL documents based on latent dirichlet allocation (LDA). Chen *et al.* [39] integrated both the WSDL documents of services and tagging data and proposed a user tagging augmented LDA (TA-LDA) model. In contrast to their semantics-based service recommendation methods, our work complements their work and focuses on leveraging social relations among users to enhance service recommendation performance.

QoS-Aware Recommendation: Quality-of-Service (QoS) is widely employed to represent the nonfunctional performance of Web services and has been adopted as a key factor in service selection. Zheng *et al.* [24] proposed a user-collaborative mechanism for collecting historical QoS data of Web services from different service users. Hu *et al.* [40] integrated time information into both similarity measurement and QoS prediction for high-quality Web service recommendation. Liu *et al.* [41] proposed an enhanced measurement for computing QoS similarity and a location-aware CF-based Web service QoS prediction method for service recommendation. Ahmed *et al.* [26] proposed a model predicting a Web service's behaviors, by predicting the status of underlying hidden states in terms of response time. Some researchers refer to the context feature of services [42], or utilize hybrid factorization machine models to capture the non-linear and complex feature interactions [43]. In contrast to their QoS-based service recommendation methods, our work complements their work and focuses on leveraging social relations among users to enhance service recommendation performance.

Social Relation-Based Recommendation: The aforementioned works on service recommendation assume that all the service users are independent. They mainly focus on modeling the feedback order by using users' positive and negative feedback, but do not investigate how the feedback from users' friends can be used to model users' preference on services. With the prevalence of social media, social influence is pervasive, not only in our daily physical life but also on the virtual Web space. Some researchers proposed to enhance the reliability of service recommendation by integrating users' social connections. Tang *et al.* [27] adapted the conventional CF technique by choosing to recommend users for the target user in regard to both similarity and trust between them. Li *et al.* [44] proposed to measure interest similarity between friends in video service systems and to study friend recommendation. Kalai *et al.* [45] proposed a Web service discovery process, by taking into account the best social friendships of the current user and the past invocation histories with satisfactory Web services of one's friends. Zhang *et al.* [46] define a "social plane" that relies on recommended measurements to enable network performance expectation management. Lu *et al.* [47] propose a service recommendation model based on data compensation and dynamic user interest grouping in social networks.

TABLE VI
COMPARISON OF HSGNN AND ITS RELATED WORK

Methods	Social Information	High-order Social Relation	Learnable Social Influence
[24], [26], [36]–[43]	\	\	\
[27], [44]–[46]	✓	\	\
[9], [47]	✓	\	✓
HSGNN	✓	✓	✓

The characteristics of HSGNN and its related works are listed in Table VI. We can observe that all of those previously published studies, though inspiring, are limited to users' direct friends without considering the **high-order connectivity** from user-user interactions. Moreover, the social influence strengths in most of the works are usually set equal or relied on a predefined static function, which should be dynamic and learnable [9].

B. Graph Representation Learning

Research on graph representation learning has rapidly gained significant attention in recent years, since many real-world data can be represented by graphs conveniently. Perozzi *et al.* [8] developed *DeepWalk* that learns social representations of a graph's vertices, by modeling a stream of short random walks. Tang *et al.* [48] introduced a network embedding method called *LINE*, which can be applied to arbitrary types of large-scale information networks. Grover and Leskovec [49] proposed *node2vec*, which can learn representations that organize nodes based on their network roles and/or communities to which they belong.

In the last a couple of years, graph convolution networks (GCNs) [50] have shown strong capability of learning on graph representations. GCMC [51] utilized GCN on a user-item graph, with only one convolutional layer to extract the direct relationships between users and items. Thus it failed to encode the high-order signals of the graph. GraphSAGE [11] applies GNNs to borrow signals from the neighbors of a node to enrich the embedding of the node in a graph. In contrast to its covering all neighbors nodes, we applied the RWR strategy to select a fixed number of neighbor nodes to ensure performance, while simulating real-world social relations. PinSAGE [52] applied GCN layers to learn embeddings for nodes in item-item graphs, which does not take account of users' social relationships. Wang *et al.* [19] proposed NGCF based on graph neural network, which explicitly encodes the collaborative signals in the form of high-order connectivity by performing embedding propagation. However, it simultaneously updates the presentations for all users and items in a same graph, which fails to run on large-scale graph, due to its

high space complexity. In contrast to existing work, we differentiate the two aspects of social impacts from high-order social relations, and applied to service recommendation. Despite the inspiring success achieved by previous work, little attention has been paid to social-based service recommendation with GCNs. In this paper, we proposed a social attentional graph convolutional network for the service recommendation to fill this gap.

When GNNs aggregate signals from neighbors, there are many aggregation operators proposed such as max pooling, sum, and average operators. In this work, we adopted the neural graph collaborative filtering [19] that considers the degrees of both message sender and recipient. However, their neural networks share the same pair of weight matrices in all layers. In contrast, we learn separate pairs of weight matrices at each layer of neural networks.

The recent graph attention networks [11], [13] introduced an attention mechanism to differentiate user-user impact in a social network. In contrast, our attention mechanism adds in the target service as the third item, aiming to study a finer-grained, context-aware user-user impact.

VI. Conclusion

With the prevalence of online social networks, increasingly more service platforms tend to leverage the social networks among users to alleviate the data sparsity problem and to enhance service recommendation quality. However, one main deficiency affecting the accuracy of existing social relation-based recommending approaches, is that high-order social relations are not considered in a comprehensive manner.

In this article, we have introduced a novel model HSGNN that unifies graph convolutional techniques and attention mechanisms, to seamlessly integrate the high-order social similarity and difference into user representations simultaneously, for more accurate service recommendation. On the one hand, we have applied graph convolution to model the multi-hop propagation process of social information. On the other hand, we have developed a neighbor-level attention instrument to learn context-aware (i.e., target service oriented) social impact. Extensive experiments have proved that HSGNN outperforms the baseline methods in the prediction accuracy on the real-world game service dataset. The neighbor-level attention mechanism also enables certain explainability of service recommendation results.

In this work, to focus on learning from social relations, we intentionally simplify the embedding initialization step (i.e., using trivial identifiers). In our future work, we plan to extend HSGNN to incorporate the content and context information of services to deal with the service-side cold-start problem. In addition, we plan to investigate and improve the explainability of our model. This is feasible since our social embedding propagation technique can help to explain the influence diffusion process in social networks. Furthermore, we plan to investigate parallel and efficient techniques to handle training of many constructed graph neural networks when the social network becomes very large scale.

References

[1] J. Wu, S. Guo, H. Huang, W. Liu, and Y. Xiang, "Information and communications technologies for sustainable development goals: State-of-the-art, needs and perspectives," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2389–2406, 3rd Quart., 2018.

[2] H. Song, J. Bai, Y. Yi, J. Wu, and L. Liu, "Artificial intelligence enabled Internet of Things: Network architecture and spectrum access," *IEEE Comput. Intell. Mag.*, vol. 15, no. 1, pp. 44–51, Feb. 2020.

[3] K.-Y. Goh, C.-S. Heng, and Z. Lin, "Social media brand community and consumer behavior: Quantifying the relative impact of user- and marketer-generated content," *Inf. Syst. Res.*, vol. 24, no. 1, pp. 88–107, 2013.

[4] B. Bai, Y. Fan, W. Tan, and J. Zhang, "DLTSR: A deep learning framework for recommendations of long-tail Web services," *IEEE Trans. Services Comput.*, vol. 13, no. 1, pp. 73–85, Jan./Feb. 2020.

[5] P. V. Marsden and N. E. Friedkin, "Network studies of social influence," *Soc. Methods Res.*, vol. 22, no. 1, pp. 127–151, 1993.

[6] H. Ma, H. Yang, M. R. Lyu, and I. King, "SoRec: Social recommendation using probabilistic matrix factorization," in *Proc. 17th ACM Conf. Inf. Knowl. Manag. (CIKM)*, 2008, pp. 931–940.

[7] M. Jamali and M. Ester, "A matrix factorization technique with trust propagation for recommendation in social networks," in *Proc. 4th ACM Conf. Recommender Syst.*, 2010, pp. 135–142.

[8] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (KDD)*, 2014, pp. 701–710.

[9] C. Chen, M. Zhang, Y. Liu, and S. Ma, "Social attentional memory network: Modeling aspect-and friend-level differences in recommendation," in *Proc. 12th ACM Int. Conf. Web Search Data Min. (WSDM)*, 2019, pp. 177–185.

[10] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2017, pp. 1024–1034.

[11] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018, pp. 1–9.

[12] Y. Yin, L. Chen, Y. Xu, and J. Wan, "Location-aware service recommendation with enhanced probabilistic matrix factorization," *IEEE Access*, vol. 6, pp. 62815–62825, 2018.

[13] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2017, pp. 5998–6008.

[14] Y. Zhang, T. Lei, and Y. Wang, "A service recommendation algorithm based on modeling of implicit demands," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, 2016, pp. 17–24.

[15] K. A. Botangen, J. Yu, S. Yongchareon, L. Yang, and Q. Z. Sheng, "Integrating geographical and functional relevance to implicit data for Web service recommendation," in *Proc. Int. Conf. Service Orient. Comput. (ICSOC)*, 2019, pp. 53–57.

[16] K. Fletcher, "Regularizing matrix factorization with implicit user preference embeddings for Web API recommendation," in *Proc. IEEE Int. Conf. Services Comput. (SCC)*, 2019, pp. 1–8.

[17] C. Wei, Y. Fan, J. Zhang, and H. Lin, "A-HSG: Neural attentive service recommendation based on high-order social graph," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, 2020, pp. 338–346.

[18] H. Tong, C. Faloutsos, and J.-Y. Pan, "Fast random walk with restart and its applications," in *Proc. IEEE 6th Int. Conf. Data Min. (ICDM)*, 2006, pp. 613–622.

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

[20] M. Wang *et al.*, "Deep graph library: Towards efficient and scalable deep learning on graphs," in *Proc. ICLR Workshop Represent. Learn. Graphs Manifolds*, 2019, pp. 1–9.

[21] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2013, pp. 3111–3119.

[22] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the Web," Stanford InfoLab, Stanford, CA, USA, Rep. 1999-66, Nov. 1999. [Online]. Available: http://ilpubs.stanford.edu:8090/422/

[23] A. V. Paliwal, B. Shafiq, J. Vaidya, H. Xiong, and N. Adam, "Semantics-based automated service discovery," *IEEE Trans. Services Comput.*, vol. 5, no. 2, pp. 260–275, Apr.–Jun. 2011.

[24] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "QoS-aware Web service recommendation by collaborative filtering," *IEEE Trans. Services Comput.*, vol. 4, no. 2, pp. 140–152, Apr.–Jun. 2010.
[25] G. Lam and D. Rossiter, "A Web service framework supporting multimedia streaming," *IEEE Trans. Services Comput.*, vol. 6, no. 3, pp. 400–413, Jul.–Sep. 2013.
[26] W. Ahmed, Y. Wu, and W. Zheng, "Response time based optimal Web service selection," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 2, pp. 551–561, Feb. 2015.
[27] M. Tang, Y. Xu, J. Liu, Z. Zheng, and X. F. Liu, "Trust-aware service recommendation via exploiting social networks," in *Proc. IEEE Int. Conf. Services Comput. (SCC)*, 2013, pp. 376–383.
[28] 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.
[29] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web (WWW)*, 2017, pp. 173–182.
[30] K. Huang, Y. Fan, and W. Tan, "An empirical study of programmable Web: A network analysis on a service-mashup system," in *Proc. 19th IEEE Int. Conf. Web Services (ICWS)*, 2012, pp. 552–559.
[31] M. O'Neill, E. Vaziripour, J. Wu, and D. Zappala, "Condensing steam: Distilling the diversity of gamer behavior," in *Proc. ACM Internet Meas. Conf.*, 2016, pp. 81–95.
[32] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," 2012, *arXiv:1205.2618*.
[33] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS*, 2019, pp. 8026–8037.
[34] J. Wu, S. Guo, J. Li, and D. Zeng, "Big data meet green challenges: Big data toward green applications," *IEEE Syst. J.*, vol. 10, no. 3, pp. 888–900, Sep. 2016.
[35] R. Atat, L. Liu, J. Wu, G. Li, C. Ye, and Y. Yi, "Big data meet cyber-physical systems: A panoramic survey," *IEEE Access*, vol. 6, pp. 73603–73636, 2018.
[36] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang, "Similarity search for Web services," in *Proc. 30th Int. Conf. Very Large Data Bases (VLDB)*, vol. 30, 2004, pp. 372–383.
[37] S. Meng, W. Dou, X. Zhang, and J. Chen, "KASR: A keyword-aware service recommendation method on MapReduce for big data applications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 12, pp. 3221–3231, Dec. 2014.
[38] C. Li, R. Zhang, J. Huai, X. Guo, and H. Sun, "A probabilistic approach for Web service discovery," in *Proc. IEEE Int. Conf. Services Comput. (SCC)*, 2013, pp. 49–56.
[39] L. Chen, Y. Wang, Q. Yu, Z. Zheng, and J. Wu, "WT-LDA: User tagging augmented LDA for Web service clustering," in *Proc. Int. Conf. Service Orient. Comput. (ICSOC)*, 2013, pp. 162–176.
[40] Y. Hu, Q. Peng, X. Hu, and R. Yang, "Time aware and data sparsity tolerant Web service recommendation based on improved collaborative filtering," *IEEE Trans. Services Comput.*, vol. 8, no. 5, pp. 782–794, Sep./Oct. 2015.
[41] J. Liu, M. Tang, Z. Zheng, X. Liu, and S. Lyu, "Location-aware and personalized collaborative filtering for Web service recommendation," *IEEE Trans. Services Comput.*, vol. 9, no. 5, pp. 686–699, Sep./Oct. 2016.
[42] S. Li, J. Wen, F. Luo, M. Gao, J. Zeng, and Z. Y. Dong, "A new QoS-aware Web service recommendation system based on contextual feature recognition at server-side," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 2, pp. 332–342, Jun. 2017.
[43] G. Kang, J. Liu, Y. Xiao, B. Cao, Y. Xu, and M. Cao, "Neural and attentional factorization machine-based Web API recommendation for mashup development," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 4, pp. 4183–4196, Dec. 2021.
[44] Z. Li, J. Lin, K. Salamatian, and G. Xie, "Social connections in user-generated content video systems: Analysis and recommendation," *IEEE Trans. Netw. Service Manag.*, vol. 10, no. 1, pp. 70–83, Mar. 2013.
[45] A. Kalaï, C. A. Zayani, and I. Amous, "User's social profile-based Web services discovery," in *Proc. IEEE 8th Int. Conf. Service Orient. Comput. Appl. (SOCA)*, 2015, pp. 2–9.
[46] Y. Zhang, P. Calyam, S. Debroy, and S. S. Nuguri, "Social plane for recommenders in network performance expectation management," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 1, pp. 97–111, Mar. 2018.
[47] X. Lu, J. Liu, S. Gan, T. Li, Y. Xiao, and Y. B. Liu, "Recommendation model based on dynamic interest group identification and data compensation," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 1, pp. 89–99, Mar. 2022.
[48] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding," in *Proc. 24th Int. Conf. World Wide Web (WWW)*, 2015, pp. 1067–1077.
[49] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (KDD)*, 2016, pp. 855–864.
[50] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.
[51] R. V. D. Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," 2017, *arXiv:1706.02263*.
[52] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for Web-scale recommender systems," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (KDD)*, 2018, pp. 974–983.

Chunyu Wei received the B.S. degree in control theory and application from Tsinghua University, China, in 2019, where he is currently pursuing the Ph.D. degree with the Department of Automation. His research interests include services management, service recommendation, and social computing.

Yushun Fan received the Ph.D. degree in control theory and application from Tsinghua University, China, in 1990, where he is currently a Professor with the Department of Automation and the Director of the System Integration Institute and the Networking Manufacturing Laboratory. 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 ten 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.

Jia Zhang (Senior Member, IEEE) received the B.S. and M.S. degrees in computer science from Nanjing University, China, and the Ph.D. degree in computer science from the University of Illinois at Chicago. She is currently the Cruse C. and Marjorie F. Calahan Centennial Chair of Engineering and the 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.