GSL4Rec: Session-based Recommendations with Collective Graph Structure Learning and Next Interaction Prediction

Chunyu Wei[1*], Bing Bai[1*], Kun Bai[1], Fei Wang[2]
[1]Tencent Security Big Data Lab, Tencent Inc., China
[2]Department of Population Health Sciences, Weill Cornell Medicine, USA
{steviewei,icebai,kunbai}@tencent.com
few2001@med.cornell.edu

ABSTRACT

Users' social connections have recently shown significant benefits to session-based recommendations, and graph neural networks have demonstrated great success in learning the pattern of information flow among users. However, the current paradigm presumes a given social network, which is not necessarily consistent with the fast-evolving shared interests and is expensive to collect. We propose a novel idea to learn the graph structure among users and make recommendations collectively in a coupled framework. This idea raises two challenges, i.e., scalability and effectiveness. We introduce a novel graph-structure learning framework for session-based recommendations (GSL4Rec) for solving both challenges simultaneously. Our framework has a two-stage strategy, i.e., the coarse neighbor screening and the self-adaptive graph structure learning, to enable the exploration of potential links among all users while maintaining a tractable amount of computation for scalability. We also propose a phased heuristic learning strategy to sequentially and synergistically train the graph learning part and recommendation part of GSL4Rec, thus improving the effectiveness by making the model easier to achieve good local optima. Experiments on five public datasets demonstrate that our proposed model significantly outperforms strong baselines, including state-of-the-art social network-based methods.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; *Social recommendation.*

KEYWORDS

session-based recommendations, graph neural networks, graph structure learning

ACM Reference Format:
Chunyu Wei, Bing Bai, Kun Bai, and Fei Wang. 2022. GSL4Rec: Session-based Recommendations with Collective Graph Structure Learning and Next Interaction Prediction. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3485447.3512085

∗ Equal contributions from both authors. This work is done when Chunyu Wei works as an intern at Tencent.

WWW '22, April 25–29, 2022, Virtual Event, Lyon, France

ACM ISBN 978-1-4503-9096-5/22/04…$15.00
https://doi.org/10.1145/3485447.3512085

1 INTRODUCTION

Personalization has become one of the key factors of today's online experience. Recommender systems aim to predict users' future interactions, such as the next click or consumption and display the personalized items of interest at the most conspicuous position to encourage more transactions. In order to respond to user needs in the most timely manner, *session-based recommendations* [4, 9, 25] are proposed. A session refers to multiple user-item interactions that happen together continuously. Session-based recommendations focus on user behaviors within a session and predict users' preference at the point that a session is being generated during the interaction process [25]. Since users' behaviors within a session are in close temporal proximity and often share the same focused objective, session-based recommendations can better capture this immediate interest and make more accurate recommendations. Recently, to incorporate more helpful information for better recommendations, researchers have proposed *session-based social recommendations* [7, 22, 23]. The primary idea is that friends often share related interests, and users are likely to be influenced by their friends. For example, funny videos can spread along with the social networks, and introducing friends' viewing behaviors can effectively make timely recommendations when target users show relevant short-term interests. Graph neural networks (GNNs) have successfully modeled users' relationships for recommendations and have also been used in many different applications [5–7, 14, 15, 22, 23].

However, existing GNN-based methods often presume a predefined social network and use this static graph to learn the information flow patterns among users. Nevertheless, hypothesizing a given and static social network brings two limitations. First, constructing social networks in real-world applications is expensive, and it is implausible to apply existing methods without such a given graph. Second, social relationships are mainly static and not necessarily consistent with users' shared interests [22]. In many cases, two users with social connections may not share the same preferences over items. On the other hand, two users sharing similar preferences may not even know the existence of each other [29]. Under these scenarios, the social network-based graph may hinder the performance of recommendation algorithms, especially when users' interests are evolving fast. Song et al. [22] overcome the second issue by introducing virtual friends based on hand-crafted heuristic rules, which could be sub-optimal. To this end, we argue that learning the graph structure (i.e., links between users in the

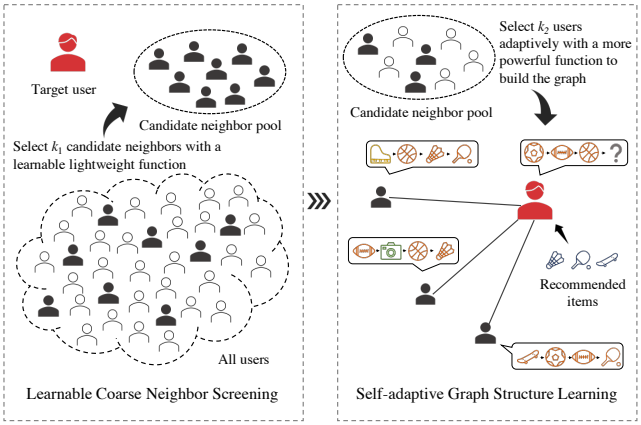


Figure 1: The overview of the GSL4Rec framework. We apply a two-stage strategy for learning graph structure. The first stage uses a learnable lightweight function to recall k_1 candidate neighbors from all users, and the second stage utilizes more information to adaptively select k_2 users from the candidate neighbor pool to build the user graph.

graph) and predicting the next interaction collectively may further benefit the recommendation performance.

To achieve this goal, we need to address two notable challenges, including,

- **Scalability**. Mining potential links between two arbitrary users require n^2-order calculations where n is the number of nodes (i.e., users) in the graph, which brings significant scalability challenges, especially for recommendation problems, where the number of users can easily reach tens of thousands. Thus, solving the scalability problem is one of the fundamental challenges for learning graph structures for recommendations.
- **Effectiveness**. With the enormous number of potential links among users, the model may easily get overfitted to the noise without sufficient inductive bias. Moreover, since learning graph structure and recommendation closely couple with each other, the changes in one part make the learning process of the other side more strenuous, especially when we initialize both parts with random states. So it would be more desirable to learn both the graph structure and recommendations simultaneously.

In this paper, we propose a novel Graph Structure Learning framework for session-based Recommendations (GSL4Rec) to address the aforementioned challenges synergistically. The overview of GSL4Rec is presented in Figure 1.

To address the scalability issue, we propose a two-stage strategy to learn graph structure. The first stage is the *coarse neighbor screening*, which uses a learnable lightweight function to recall k_1 potential neighbors from all users. In this way, we enable exploring possible links from the whole graph while maintaining an amount of tractable computation. The second stage is the *self-adaptive graph structure learning*, based on the coarse neighbor screening results and can adaptively select a small number, i.e., k_2, of neighbors of interest. This stage utilizes more information, including users' recent interactions, to predict which user's recent behavior has the most significant value for the target user's recommendation. We can successfully scale up to achieve the goal of learning graph structure in recommendation scenarios with this two-stage strategy and the setting of $k_2 < k_1 \ll n$.

To address the effectiveness issue, we propose a phased heuristic learning strategy to stabilize the learning process. The essential idea is to maximize the value of existing social networks to provide a warm start and steadily increase the weight of learned graph structure to encourage exploration. The learning strategy has three phases. In the first phase, we fix the graph structure as the given social network and train the recommendation until it reaches a good state. Then in the second phase, we gradually increase the weight of the learnable network structure part. Since the recommendation part is already pre-trained, we can avoid the phenomenon of two randomly initialized parts disrupting each other. In the final phase, we rely entirely on the learned network structure and fine-tune the whole network. This phased heuristic learning strategy can sequentially and synergistically train different parts of GSL4Rec and make the model easier to achieve good local optima. Moreover, even when the social network is absent, we can build an initial graph based on hand-crafted rules and apply the proposed strategy.

The contributions of this paper are summarized as follows.

- We propose GSL4Rec to learn graph structure from user behaviors for session-based recommendations. The proposed framework applies to large datasets and can infer weighted directed graphs adaptively with or without a given social network at prior. To the best of our knowledge, this is the first study on learning the graph structure of user networks instead of using hand-craft rules for session-based recommendations.
- We propose a phased heuristic learning strategy to stabilize the learning process, with which we overcome the challenges for parameter learning and achieve better performance.
- Experimental results show that our method outperforms the state-of-the-art methods on five benchmark datasets from different domains and demonstrate the effectiveness of learned graph structures.

The rest of this paper is organized as follows. Section 2 summarizes the related work. Section 3 gives introduction to the notations and formalize the problems. Section 4 introduces the methodology, and Section 5 talks about the learning strategy. Section 6 reports the experimental results. And finally, Section 7 draws the conclusion.

2 RELATED WORK

2.1 Session-based Recommendations

Session-based recommendations have drawn increasing interest in recent years. To capture the most timely and dynamic user interests, session-based recommendations only utilize the active session's previous actions to predict users' interests and make recommendations. A session refers to multiple user-item interactions that happen together in a continuous time, which usually lasts for several minutes to several hours [25]. Existing work towards session-based recommendations can be divided into three categories, i.e., conventional

approaches, latent representation-based approaches, and neural network (NN)-based approaches.

Among them, conventional approaches leverage conventional data mining or machine learning techniques, including rule mining-based approaches [19, 30], k nearest neighbor (kNN)-based approaches [16], and Markov Chain-based approaches [26]. On the other hand, latent representation-based approaches make recommendations with low-dimensional latent representations for each interaction within sessions with shallow models [11]. Recently, NN-based approaches have become popular thanks to the ability to model complex intra- and inter-session dependencies. For example, GRU4Rec [9] applied gated recurrent units (GRU) to model sequential behaviors and make recommendations. Hidasi and Karatzoglou [8] further improved the performance by new ranking loss functions. Li et al. [13] proposed a hybrid encoder with an attention mechanism to better capture the user's primary purpose.

More recently, graph neural networks (GNNs) have shown great potential in modeling complex relationships embedded in graph-structured data, such as user relationships. To model relevant interests and mutual influence of users in social networks, Song et al. [23] proposed DGRec by introducing graph attention networks (GATs) [24] together with recurrent neural networks (RNNs). To improve the performance when the social network is too sparse, Song et al. [22] proposed DREAM by introducing virtual friends based on heuristic rules. This paper argues that fixed social networks and heuristic rules may be sub-optimal and propose the GSL4Rec framework to jointly learn the graph structure and predict the next interaction in a coupled framework.

2.2 Graph Neural Networks

In many real-world applications, the data can be represented in graphs, e.g., social networks, citation networks, chemistry molecules, and road maps. A graph may have a variable size of unordered nodes, and the nodes may have different numbers of neighbors, making it difficult for traditional operations designed for Euclidean data (e.g., convolutions) to apply [27]. As a result, graph neural networks (GNNs) emerge to tackle this challenge. Here we mainly review important graph convolutional network (GCN) models. Bruna et al. [3] first proposes a spectral-based graph convolutional network by introducing the spectral graph theory into neural networks. On the other side, spatial-based graph convolutional networks [2, 18] directly propagate node information along edges in the spatial domain. Graph attention networks (GATs) improve spatial-based graph convolutional networks by considering different weights to the central node for different neighbors [24]. However, existing GNN methods usually assume the graphs are given and fixed, while in this paper, we explore how to learn graph structures beyond the social networks.

3 PROBLEM DEFINITION

Definition 1 (Session-based Recommendations). *Let $\mathcal{U}$ denote the set of users, and let $\mathcal{I}$ denote the set of items. The behavior history of a user $u \in \mathcal{U}$ by the time step of T can be segmented into a series of sessions, i.e., $H_T^u = \{S_1^u, S_2^u, \ldots, S_T^u\}$, where S_t^u is the session of user u at time step t. For the tth session of u, a series of items are interacted i.e., $S_t^u = \{i_{t,1}^u, i_{t,2}^u, \ldots, i_{t,n_t^u}^u\}$ and n_t^u is the amount of items in u's tth session. Given a part of a new session $S_{T+1}^u = \{i_{T+1,1}^u, i_{T+1,2}^u, \ldots, i_{T+1,m}^u\}$, the task of session-based recommendations is to recommend a set of items from $\mathcal{I}$ that are likely to be interacted by user u in the next step, i.e., $i_{T+1,m+1}^u$.*

Recent research brings social relationships and the neighbors' behavior on the social network into session-based recommendations. Similar to Song et al. [23], we formalize the problem of session-based social recommendations as follows:

Definition 2 (Session-based Social Recommendations). *Let $\mathcal{G} = \langle \mathcal{U}, \mathcal{E} \rangle$ denote the social network, where $\mathcal{U}$ is the set of users, and $\mathcal{E}$ is the set of social links between users. Given a part of a new session $S_{T+1}^u = \{i_{T+1,1}^u, i_{T+1,2}^u, \ldots, i_{T+1,m}^u\}$, the task of session-based social recommendations is to recommend a set of items from $\mathcal{I}$ that are likely to be interacted by user u in the next step, i.e., $i_{T+1,m+1}^u$, with the help of the information from the neighbors, i.e., $\cup_{k \in \mathbf{N}_u} S_T^k$, where $\mathbf{N}_u$ is the neighbors of user u in the graph.*

Song et al. [23] address the problem of session-based social recommendations. However, we argue that social relationships are expensive to collect and may become a limitation of graph neural networks. In this paper, based on the definition of session-based social recommendations, we propose graph structure learning for (session-based social) recommendations as follows:

Definition 3 (Graph Structure Learning for Recommendations). *Let $\mathcal{G}_0 = \langle \mathcal{U}, \mathcal{E}_0 \rangle$ denote the initial graph structure. Besides recommending the potential items of interest as Definition 2, graph structure learning also requires predicting the dynamic links among users to maximum the recommendation performance, i.e., $\{\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_{T+1}\}$.*

4 METHODOLOGY

This section outlines our GSL4Rec framework's overall architecture and gives detailed descriptions of its components. We put the time complexity analysis in Appendix A.1.

4.1 Model Architecture

The framework of GSL4Rec is illustrated in Figure 2. We introduce its architecture from left to right. GSL4Rec consists of four core modules. (1) ***Coarse Neighbor Screening Layer*** that computes a graph similarity matrix to discover hidden associations among massive users. (2) ***Dynamic Interest Modeling***, which models the short-term and long-term interest of each user. (3) ***Self-adaptive Graph Structure Learning***, which utilizes users' dynamic interest to adaptively discover a group of more influential neighbors in the graph and omit those lower-impact edges. (4) ***Attentive Interest Propagation Layer***, which leverages a graph-convolution network to re-scale weights of the user's neighbors and combine their representations with the target user's representation.

Modules (1) and (3) form the two-stage strategy for graph structure learning, while modules (2) and (4) form the user interest modeling and diffusing process. These two processes are alternatively performed and support each other to simultaneously improve the graph structure learning and recommendation.

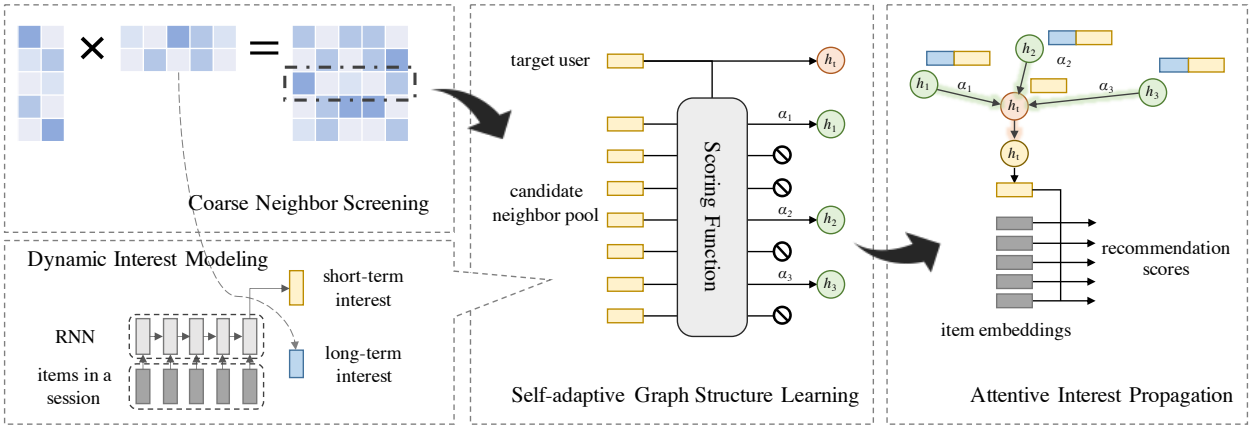


Figure 2: The overview of the model structure of GSL4Rec. It consists of four parts: a coarse neighbor screening layer, a dynamic interest modeling module, a self-adaptive graph structure learning module, and an attentive interest propagation layer.

4.2 Coarse Neighbor Screening Layer

Our GSL4Rec framework offers a Coarse Neighbor Screening Layer to discover and quantify interest dependencies among a tremendous amount of user vertices. To achieve this with limited computing resources, we randomly initialize two learnable parameter matrices $\mathbf{E}_{tar}, \mathbf{E}_{src} \in \mathbb{R}^{N \times d}$ to represent the target and source node embeddings, respectively. We propose to obtain the asymmetric first-stage similarity matrix as follows:

$$\mathbf{S} = \text{ReLU}(\tanh(\mathbf{E}_{tar}\mathbf{E}_{src}^T)). \tag{1}$$

By multiplying $\mathbf{E}_{tar}$ and $\mathbf{E}_{src}^T$ followed by a ReLU(tanh$(\cdot)$) operation, each cell S_{uk} in the matrix $\mathbf{S}$ can be interpreted as the probability that source node k has influence on target node u. For each user u, we select the top-k_1 vertices with the highest probabilities in $S_{u\cdot}$ as her coarse screening neighbors, which can be denoted as $\mathbf{N}_u = \text{argtop}_{k_1}(S_{u\cdot})$.

At the serving stage, to further reduce the computational cost, we apply the approximate nearest neighbor searching algorithms [1], as the ReLU(tanh$(\cdot)$) function is monotonical.

4.3 Dynamic Interest Modeling

After obtaining the coarse-screened neighbors, we consider both short-term and long-term interests to get a user's fast-changing and comprehensive preference.

Short-term interest. We refer to the short-term interest as a user's latent preference hidden in the corresponding sessions (i.e., the current interest after interacting with a series of items). Following recent advances in session-based recommendations [23], we use RNNs to model the interactions of a user in a session. Given user u's session $S_T^u = \{i_{T,1}^u, i_{T,2}^u, \ldots, i_{T,m}^u\}$, the RNN model infers the user's current state by combining her previous state and the latest interacting item. In this paper, we learn the function by using a long-short term memory (LSTM) [10] structure, which is well-known for alleviating vanishing and exploding gradients, i.e.,

$$s_T^u = s_{T,m}^u = \text{LSTM}(i_{T,m}^u, s_{T,m-1}^u), \tag{2}$$

where s_T^u represents the short-term interests.

Long-term interest. The long-term interest usually remains roughly constant for a long time. Thus we use a learnable embedding to depict a user's long-term interest. In practice, we reuse the embedding matrix $\mathbf{E}_{src}$ in Section 4.2, since we mainly focus on the influence of preferences on others in the long-term interest modeling. Formally, user u's long-term interest is defined as follows:

$$l^u = \mathbf{E}_{src}[u, :], \tag{3}$$

where $\mathbf{E}_{src}[u, :]$ means the u-th row of $\mathbf{E}_{src}$.

After we obtain the short-term and long-term representations of user u's neighbor $k \in \mathbf{N}_u$, we concatenate both of them followed by a non-linear transformation to get k's final representation:

$$p_{src}^k = \text{ReLU}(\mathbf{W}_1 concat([s_T^k, l^k])), \ k \in \mathbf{N}_u, \tag{4}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d \times 2d}$ is the transformation matrix. For the target user u, since we aim to predict her next step and don't consider her influence on others, we only use the short-term interest to characterize [23]:

$$p_{tar}^u = s_{T+1}^u. \tag{5}$$

4.4 Self-adaptive Graph Structure Learning

After we receive the short-term interest of the user nodes from Section 4.3, the self-adaptive graph structure learning layer performs the second-stage neighbor selection in a more fine-grained way based on short-term information. Existing methods in other research fields, like time series forecasting, usually measure the similarity between pairs of nodes by multiplying two matrices [28, 29, 31] to construct a graph between the user and her potential neighbors, which may limit the model's capability of extracting the nonlinear patterns among nodes. Nevertheless, most users get influence from a few key opinion leaders in recommendation scenarios, making the adjacency matrix extremely asymmetric. To remedy the above issues, we apply a two-layer neural network to extract such nonlinear and asymmetric relationships between users, which we illustrate as follows:

$$\tilde{\mathbf{S}}[u, k] = \text{ReLU}(\mathbf{h}^T \text{ReLU}(\mathbf{W}_2 concat([s_{T+1}^u, s_T^k]) + \mathbf{b}_2))), \tag{6}$$

where $\tilde{\mathbf{S}}[u,k] \in \mathbb{R}$ refers to the inferred asymmetric similarity from user u to her neighbor k and $\tilde{\mathbf{S}} \in \mathbb{R}^{N\times N}$ denotes the second-stage similarity matrix. Note that the entry $\tilde{\mathbf{S}}[u,k]$ will only be calculated when k is in user u's coarse screening neighbors or otherwise will be set as zero. The coarse screening step in Section 4.2 makes the similarity matrix sparse and greatly reduces the similarity computation cost. $\mathbf{W}_2 \in \mathbb{R}^{d\times 2d}$, $\mathbf{b}_2 \in \mathbb{R}^d$ and $\mathbf{h} \in \mathbb{R}^d$ are model parameters. We then select the top-k_2 neighbors with the highest relevance for the target user as the second-stage selection and set the other neighbors' weights as zeros to form the adjacency matrix $\tilde{\mathbf{A}}$, which is formulated as follows:

$$\tilde{\mathbf{N}}_u = \text{argtop}_{k_2}(\tilde{\mathbf{S}}[u,:]), \tag{7}$$

$$\tilde{\mathbf{A}}[u,k'] = \begin{cases} Softmax(\tilde{\mathbf{S}}[u,k']), & k' \in \tilde{\mathbf{N}}_u \\ 0, & \text{Otherwise} \end{cases}, \tag{8}$$

where argtop_{k_2} returns the index of the top-k_2 most relevant neighbors of the user and $\tilde{\mathbf{N}}_u$ denotes the neighbors selected from $\mathbf{N}_u$ after the second-stage learning. The neighbor-wise $Softmax$ is applied to normalize the self-adaptive adjacency matrix $\tilde{\mathbf{A}}$.

4.5 Attentive Interest Propagation Layer

After receiving the inferred fine-grained neighbors and their influence weight from the graph structure learning process, we use a GCN-based model to obtain the neighbor context information of the target user by integrating her related neighbors' preferences.

4.5.1 Graph building. According to Section 4.4, we build the weighted directed graph with the nodes corresponding to user u and her selected neighbors in $\tilde{\mathbf{N}}_u$. And each entry $\tilde{\mathbf{S}}[u,k'])$ in the self-adaptive adjacency matrix $\tilde{\mathbf{A}}$ is the level of influence or weight of friend k' on user u. Based on Section 4.3, we assign p^u_{tar} as the target user's initial representation, which only includes her short-term interest and will be updated when a new item interacts. We use $p^k_{src},\ k \in \tilde{\mathbf{N}}_u$ containing both long-term and short-term interest as the neighbors' initial representation. We use $\mathbf{P}_{tar}$ and $\mathbf{P}_{src}$ to denote the target and source feature matrices, respectively. Specifically, each of their rows $\mathbf{P}_{tar}[j,:] = p^j_{tar}$ and $\mathbf{P}_{src}[j,:] = p^j_{src}$.

4.5.2 Graph convolution. Our GSL4Rec integrates a GCN-based module as the backend to simulate the interest propagation between pairs of users. By defining different Laplacian matrices, e.g., normalized Laplacian and random walk Laplacian, GCN can update a node's signal by aggregating and transforming its neighborhood information through different graph convolution operators, e.g., Chebyshev convolution and diffusion convolution [14]. In this paper, we find that diffusion convolution performs better when modeling users' interest propagation, which is defined as follows:

$$\mathbf{P}^{(l)}_{tar} = \text{ReLU}((\tilde{\mathbf{A}}\mathbf{P}_{src} + \mathbf{P}^{(l-1)}_{tar})\mathbf{W}^{(l)}), \tag{9}$$

where $\mathbf{W}^{(l)}$ is the shared and learnable weight matrix at layer l and $\mathbf{P}^{(l)}_{tar}$ is the updated target feature matrix at layer l. Note that we keep target nodes' origin information by adding the term $\mathbf{P}^{(l-1)}_{tar}$. Formally, we let $\mathbf{P}^{(0)}_{tar} = \mathbf{P}_{tar}$. We obtain the target user's final representation by stacking the interest propagation layer L times.

4.6 Recommending

To avoid the problem of gradient vanishing and combine both the user's recent behavior and her neighbors' context influence, we use skip connections to add p^u and $p^{u(l)}$ together:

$$p^u_{final} = p^u_{tar} + p^{u(l)}_{tar}, \tag{10}$$

where p^u_{final} is the final representation of user u's current preference. We then calculate the probability of being the next interacted item on the whole item set by a softmax function:

$$\begin{aligned} &\text{Prob}(y|i^u_{T+1,1}, i^u_{T+1,2}, \ldots, i^u_{T+1,m}; \{S^k_T, k \in \tilde{\mathbf{N}}_u\}) \\ =&\frac{exp(z_y^T p^u_{final})}{\sum_{j=1}^{|\mathcal{I}|} exp(z_j^T p^u_{final})}, \end{aligned} \tag{11}$$

where z_y is item y's embedding and $|\mathcal{I}|$ is the number of all items.

5 PARAMETER LEARNING

This section discusses parameter learning and optimization for our GSL4Rec framework, which enhances the scalability to deal with the massive users in recommendation scenarios and improves the convergence to a better local optimum.

5.1 Overall Learning Objective

The GSL4Rec framework proposed a two-stage strategy to discover and quantify the dynamic interest dependencies among all the users, i.e., *coarse neighbor screening* and *self-adaptive graph structure learning*. We train and update these two stages alternatively. Specifically, the first-stage coarse neighbor screening only works and learns between epochs, while the second-stage self-adaptive graph structure learning is trained along with the recommendation part in an end-to-end manner during every training step. This intuition is that a user's long-term interest mostly depends on her character and reflects the average interest, which stays stable for a relatively long time. So we do not perform the coarse neighbor screening very often to save computation. The second-stage self-adaptive graph structure learning always receives the potential candidates from the first-stage coarse screening and selects more related neighbors from those candidates. So we expect the second-stage network to be the first-stage network's teacher, of which the feedback can guide the training process of the coarse screening layer. This two-stage strategy provides each user the full possibilities of being compared with all the other users while maintaining a tractable amount of computation for scalability. Next, we introduce the loss function for these two stages.

Loss function for recommendations. We choose the negative log-likelihood as loss function to supervise the training process under the recommendation part, which is formulated by the Equation (12):

$$\begin{aligned} \mathscr{L}_{rec} = \sum_{u\in\mathcal{U}} \sum_{t=1}^{T} \log \text{Prob}(&i^u_{t+1,m+1} | i^u_{t+1,1}, i^u_{t+1,2}, \ldots, \\ &i^u_{t+1,m}; \{S^k_t, k \in \tilde{\mathbf{N}}_u\}), \end{aligned} \tag{12}$$

where m refers to the number of items in the session.

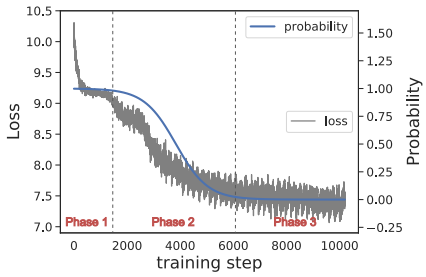


Figure 3: The loss curve on the *Yelp* dataset *v.s.* the current training step. There are three stages to drive the GSL4Rec framework, corresponding to different values of ϵ.

Loss function for the coarse screening process. The learning process of the coarse screening is somehow equivalent to the factorization of the user-user similarity matrix and the generation of a similarity ranking among all the users. Also, the feedback signal from the second stage includes the chosen (positive) neighbors and the unchosen (negative) ones from the entire user set. Thus to better perform ranking with implicit feedback, we optimize the coarse screening task using the BPR loss [21] with a score regularization:

$$\mathscr{L}_{scr} = \sum_{u\in\mathcal{U}} \sum_{i\in\tilde{\mathrm{N}}_u} \sum_{j\in(\mathcal{U}-\tilde{\mathrm{N}}_u)} -\log\left(\sigma(\mathrm{S}[u,i]-\mathrm{S}[u,j])\right) + \lambda \mathrm{S}[u,j]^2, \tag{13}$$

where the second term in Equation (13) is the regularization term. It makes sure that $\mathrm{S}[u,j]$ is moving towards zero for negative user pairs, which benefits the overall learning process.

5.2 Phased Heuristic Learning Strategy

As the training process goes, the information loop is gradually formed between the coupled graph structure learning process and the recommendation process, which helps improve each other. However, at the beginning stage, since the parameters of both sides are initialized with random states, these two coupled processes may hamper each other, making it more difficult to converge to a good local optimum. We proposed a phased heuristic strategy to stabilize the learning process to provide a warm start for the graph learning process. Specifically, we set a probability ϵ to control whether using neighbors from the social network or otherwise the learned result. With ϵ gradually decays, the recommendation process relies more and more on the learned graph structure when the graph learning part has been trained well to a certain level. We choose the inverse sigmoid function to simulate the decay of ϵ:

$$\epsilon = \frac{r}{r + \exp\left(\frac{step}{r}\right)}, \tag{14}$$

where *step* refers to the current training step, and r is the hyperparameter to control how many steps ϵ will take to reduce to about 0. Generally, we adjust r to ensure ϵ becomes close to 0 within 5 epochs. As illustrated in Figure 3, the heuristics can be divided into three phases.

At phase one, we mainly rely on the existing social networks (or the initial graph based on the handcrafted rules). The recommendation process learns to extract the user's dynamic interest and how it propagates restricted on the given graph, while the graph learning process does not work. At phase two, ϵ begins to drop rapidly, which means the graph structure comes from either prior knowledge or inferred values from the graph learning process. At this phase, the graph learning process begins learning how to measure the similarities between users based on long-term and short-term interest with an already-trained recommendation part and choose the most related neighbors for the target users. At phase three, the ϵ has converged to 0, then the two processes are entirely dependent on each other and optimized together. Through the phased heuristics, different parts of the GSL4Rec framework can be sequentially and synergistically trained, making the GSL4Rec framework easier to achieve good local optima.

Table 1: Descriptive statistics of the datasets.

	Delicious	Yelp	Lthing	Dianping	Mind
Users	1,795	6,005	21,821	13,499	5,536
Items	4,656	10,470	11,805	8,432	3,801
Event	353,802	122,526	574,875	671,793	99,292
Social Links	14,472	111,931	52,492	111,404	– –
Start Date	2006/05/04	2015/06/30	2005/01/01	2012/01/01	2019/11/09
End Date	2010/11/09	2016/06/30	2013/08/25	2013/09/30	2019/11/15
Avg. friends per user	8.06	18.64	2.41	8.25	– –
Avg. events per user	196.99	20.41	26.34	49.76	17.94
Avg. session length	6.16	5.79	9.26	9.46	3.34

6 EXPERIMENTS

6.1 Dataset Description

To comprehensively study our proposed GSL4Rec, we conduct experiments on five real-world datasets, of which descriptive statistics for all datasets are in Table 1, and more detailed information about the datasets is in Appendix B. Note that we include the available social links to evaluate some social-based methods, even though GSL4R does not rely on these social features.

We chronologically split all datasets into the training sets, the validation sets, and the test sets during the experiments. We reserved 14, 60, 603, 120, 1 days for *Delicious*, *Yelp*, *Lthing*, *Dianping* and *Mind* respectively for validation and test, and equally split the reserved sessions into validation and test sets.

6.2 Experimental Setup

Evaluation metrics. We adopt two well-known metrics to evaluate all models' performance: Normalized Discounted Cumulative Gain@K (NDCG@K) and Hit Ratio@K (HR@K). The formulation of the two metrics is in Appendix C. For both metrics, the higher scores they can obtain, the better. Compared with HR@K, NDCG@K focuses more on the top of recommendation lists. Following standard practice, we set $K = 20$ in our experiments. We repeated experiments five times with different initialization and reported the average values. Specially, we conduct the metric calculation on the entire item set. Krichene and Rendle [12] proved that this would give more reliable evaluation results and avoid the bias brought by sampling procedures.

Baselines. We compare GSL4Rec with the following seven baseline methods to evaluate the performance. We can categorize them

Table 2: Comparison between different models. Boldface denotes the highest score, and underline indicates the best result of the baselines. *Improv.* present the relative improvement of GSL4Rec over the best result of the baselines, and $*$ denotes that the improvement is significant at $p < 0.01$ level with a two-tailed pairwise t-test. Note that we calculate the metrics on the entire item sets instead of sampling negative items randomly [12].

Model	Delicious		Yelp		Lthing		Dianping		Mind†	
	NDCG@20	HR@20	NDCG@20	HR@20	NDCG@20	HR@20	NDCG@20	HR@20	NDCG@20	HR@20
BPR-MF [21]	3.75%	9.29%	0.77%	2.33%	0.41%	1.14%	0.78%	2.16%	0.74%	2.59%
SBPR [32]	5.93%	12.93%	1.14%	3.25%	0.57%	1.45%	0.90%	2.43%	– –	– –
SoReg [17]	6.55%	14.21%	1.52%	4.29%	0.48%	1.35%	0.87%	2.31%	– –	– –
GRU4Rec [9]	12.20%	26.24%	2.10%	5.80%	1.07%	2.86%	1.04%	2.84%	1.10%	3.29%
NARM [13]	11.12%	23.35%	2.09%	5.70%	1.21%	3.04%	0.91%	2.46%	2.08%	5.58%
DGRec [23]	13.14%	28.01%	2.44%	6.23%	1.08%	2.73%	1.09%	2.89%	– –	– –
DREAM [22]	13.03%	27.65%	2.40%	6.25%	1.26%	3.27%	1.19%	3.13%	2.46%	6.89%
GSL4Rec	**14.11%**	**29.93%**	**2.85%**	**7.57%**	**1.42%**	**3.62%**	**1.37%**	**3.56%**	**3.08%**	**8.21%**
Improv.	+7.38%*	+6.85%*	+16.80%*	+21.12%*	+12.70%*	+10.70%*	+15.13%*	+13.74%*	+25.20%*	+19.16%*

† Mind does not provide a social network, so SBPR, SoReg, and DGRec are not applicable, and for DREAM, we only use virtual friends to build the graph.

into four classes:(1) traditional methods only considering users' implicit feedback, i.e., BPR-MF; (2) social-based methods incorporating the social influence, i.e., SBPR and SoReg; (3) neural network-based session-based methods considering users' actions in a session, i.e., GRU4Rec and NARM; and (4) graph-based session-based models combining both social relations and temporal information of users, i.e., DGRec and DREAM. We put a detailed introduction to the baselines in Appendix D.

Hyper-parameter settings. We initialize the latent vectors with small random values for all the models. For fair comparisons, the dimensions of latent factors are all fixed to 100. We also set the number of hidden units of LSTMs or RNNs in all models as 100. We apply a grid search for hyper-parameters. The learning rate for all models are tuned amongst [0.005, 0.01, 0.02, 0.05]. To prevent overfitting, we add L_2 norm with coefficient tuned from [0.001, 0.005, 0.01, 0.02, 0.1]. We select the best models by early stopping when the HR@20 on the validation set does not increase for three consecutive epochs. For the coarse neighbor screening layer, we use SGD as the optimizer to fasten the convergence since it only works and learns once an epoch. As for the recommendation part including the second-stage self-adaptive graph structure learning, we use Adam with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1e^{-8}$, which is the same for all baseline models.

We set the recall number k_1 of the first-stage coarse screening as 100. Since different datasets have various training steps, to ensure the model have a full warm start, we set r as 800, 600, 700, 2000 and 700 for *Delicious*, *Yelp*, *Lthing*, *Dianping* and *Mind* respectively. We attempt to recall more potential neighbors but observed no significant improvement. Based on the observation in Section 6.5, we set k_2 in our GSL4Rec as $10, 40, 60, 50, 40$ for *Delicious*, *Yelp*, *Lthing*, *Dianping* and *Mind*, respectively. For a fair comparison, the virtual friend number in DREAM is also set equal to k_2.

Implementation details. We implement our model using PyTorch [20]. For the phased heuristic learning strategy, we utilize the existing social network to be the initial graph (i.e., $\mathcal{G}_0 = \mathcal{G}$). For the dataset *Mind* where the social information is missing, we identify the user's "friends" based on their historical consumption and build an initial graph based on these "friends" similar to DREAM [22].

6.3 Performance Comparisons

We summarize the performance of different algorithms in terms of HR@20 and NDCG@20 over all five datasets in Table 2. It demonstrates that GSL4Rec outperforms other methods in all evaluation metrics. Besides, We have the following observations:

- BPR-MF only considers the unordered interaction histories, which explains its limited performance. SBPR and SoReg perform better than BPR-MF in most cases, demonstrating that social neighbors' information could help improve the recommendation accuracy. By incorporating the temporal information, GRU4Rec and NARM significantly outperform BPR-MF and obtain better performance than SBPR and SoReg.
- The graph-based methods, including our GSL4Rec, consistently outperform the other methods. These results verify that incorporating social relations and users' dynamic interest is essential to enrich users' representations.
- GSL4Rec outperforms both DGRec and DREAM. DGRec is performed directly on the user's existing social networks, while DREAM proposed a rule-based heuristic technique to expand the user's social network by identifying similar users. The results demonstrate that a learned graph structure has more vital expressiveness to capture the dynamical dependencies among users, which is a critical issue in session-based recommendations.
- When the social information is absent, our GSL4Rec shows consistent performance in *Mind*. The graph-based session-based models, i.e., DREAM and GSL4Rec, perform better than other baselines. This makes sense since the timeliness of news tends to reduce the importance of the past interactions when recommending current news, while the graph-based methods can fetch useful information about the current trend from the neighborhood.

Table 3: Effect of two-stage graph structure learning. The results of the Delicious and Dianping dataset are similar hence omitted.

Variation	Yelp		Lthing	
	NDCG@20	HR@20	NDCG@20	HR@20
w/o CN	2.69%	7.18%	1.34%	3.36%
w/o SL	2.43%	6.65%	1.22%	3.30%
GSL4Rec	2.85%	7.57%	1.42%	3.62%

6.4 Ablation Studies

We conduct ablation studies on *Yelp* and *Lthing* to validate the effectiveness of the two-stage graph structure learning strategy and the phased heuristic learning strategy.

Effect of two-stage graph structure learning. To investigate the two-stage of the graph structure learning, we name GSL4Rec without different components as follows:

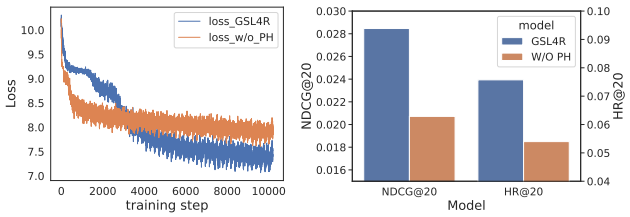
- **w/o CN**: GSL4Rec without the first-stage coarse neighbor screening layer. Since it's infeasible to apply the second-stage self-adaptive graph structure learning on complete $\mathcal{U}$ with limited memory, we replace the coarse screening neighbors N_i as the neighbors in the initial graph.
- **w/o SL**: GSL4Rec without the second-stage self-adaptive graph structure learning. We directly select k_2 neighbors through the coarse screening.

Table 3 summarizes the experimental results. Removing either coarse neighbor screening or self-adaptive graph structure learning compromises the expressive ability of the model adversely. Besides, the result show that **w/o CN** performs better than **w/o SL**. It may be because, in the task of session-based recommendation, a user's short-term interest is usually more valuable than the long-term interest, and **w/o SL** relying solely on coarse screening can only capture the long-term dependencies among users. Note that **w/o SL** still outperforms DREAM, indicating that learned long-term dependencies are more effective than handcrafted virtual friends.

Effect of phased heuristic learning. To evaluate the phased heuristics' effect, we disable the phased heuristic learning strategy and term it as **w/o PH**. Figure 4 (a) shows the training loss *w.r.t.* the number of training steps and the evaluation results on Yelp, from which we observe that the GSL4Rec framework driven by the phased heuristics performs significantly better than GSL4Rec without it. From the loss curve, we find that the loss of the GSL4Rec framework without the phased heuristics drops more quickly at the beginning and turns to a steadily decreasing state afterward. However, with the phased heuristics, GSL4Rec is more likely to converge to a better local optimum instead of getting an early-stop, which might be the reason why phased heuristic learning helps our coupled graph structure learning process, and the recommendation process performs better, as illustrated by Figure 4 (b).

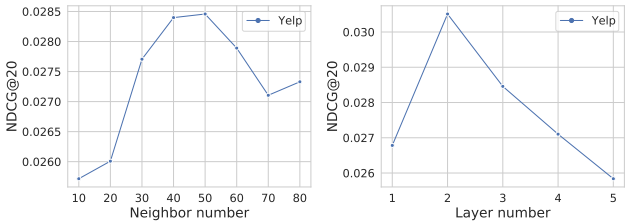
6.5 Parameter Sensitive Studies

This section investigates the influence of parameter k_2 controlling the number of selected neighbors in stage two and propagation layer number l. We vary k_2 from 10 to 80 and l from 1 to 5, respectively, while keeping other parameters fixed. The results of NDCG@20 on *Yelp* are presented in Figure 5, and the similar results on other datasets are omitted due to the space limitation. We observe from Figure 5(a) that, with the increase of k_2, the performance is boosted at first since a larger group of neighbors can bring more useful information. However, it drops after $k_2 = 50$ because it nullifies the second-stage graph structure learning and brings too much noise. From Figure 5(b), we can see that GSL4Rec achieves the best performance when $l = 2$, which implies that the interests of 2nd-hop neighbors help for recommendations, and stacking too many layers may make the target user's neighborhood features tend to be similar and reduce the performance.



(a) Loss Curve (b) Performance Comparison

Figure 4: Effect of phased heuristic learning on *Yelp*.



(a) Effect of Neighbor Number k_2 (b) Effect of Layer Number L

Figure 5: Parameter Studies on *Yelp*.

7 CONCLUSIONS

GNNs show great potential to capture the information flow on the graph and have been introduced to benefit session-based social recommendations. However, the social network is not always available and may not be consistent with shared interests among users. To expand the usage scenarios and further optimize the recommendation effect, in this paper, we raise the bar by proposing a novel graph structure learning framework for session-based recommendation (GSL4Rec) to learn the graph structure and predicting the next interaction collectively and overcome the scalability issue and the effectiveness issue. Experiments demonstrate state-of-the-art performance on 5 real-world datasets.

REFERENCES

[1] Sunil Arya, David M Mount, Nathan S Netanyahu, Ruth Silverman, and Angela Y Wu. 1998. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM (JACM)* 45, 6 (1998), 891–923.

[2] James Atwood and Don Towsley. 2016. Diffusion-convolutional neural networks. In *Advances in neural information processing systems*. 1993–2001.

[3] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral Networks and Locally Connected Networks on Graphs. In *Proceedings of the 2nd International Conference on Learning Representations, ICLR 2014*.

[4] Tianwen Chen and Raymond Chi-Wing Wong. 2020. Handling information loss of graph neural networks for session-based recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1172–1180.

[5] Edward Choi, Zhen Xu, Yujia Li, Michael Dusenberry, Gerardo Flores, Emily Xue, and Andrew Dai. 2020. Learning the graphical structure of electronic health records with graph convolutional transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 606–613.

[6] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The World Wide Web Conference*. 417–426.

[7] Pan Gu, Yuqiang Han, Wei Gao, Guandong Xu, and Jian Wu. 2021. Enhancing session-based social recommendation through item graph embedding and contextual friendship modeling. *Neurocomputing* 419 (2021), 190–202.

[8] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM international conference on information and knowledge management*. 843–852.

[9] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *The 4th International Conference on Learning Representations, ICLR 2016*.

[10] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[11] Liang Hu, Longbing Cao, Shoujin Wang, Guandong Xu, Jian Cao, and Zhiping Gu. 2017. Diversifying Personalized Recommendation with User-session Context. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*. 1858–1864.

[12] Walid Krichene and Steffen Rendle. 2020. On sampled metrics for item recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1748–1757.

[13] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1419–1428.

[14] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *Proceddings of the 6th International Conference on Learning Representations*.

[15] Haozhe Lin, Yushun Fan, Jia Zhang, and Bing Bai. 2021. REST: Reciprocal Framework for Spatiotemporal-coupled Predictions. In *Proceedings of the Web Conference 2021*. 3136–3145.

[16] Malte Ludewig and Dietmar Jannach. 2018. Evaluation of session-based recommendation algorithms. *User Modeling and User-Adapted Interaction* 28, 4-5 (2018), 331–390.

[17] Hao Ma, Dengyong Zhou, Chao Liu, Michael R Lyu, and Irwin King. 2011. Recommender systems with social regularization. In *Proceedings of the fourth ACM international conference on Web search and data mining*. 287–296.

[18] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning Convolutional Neural Networks for Graphs. In *Proceedings of the 33nd International Conference on Machine Learning*. PMLR, 2014–2023.

[19] Utpala Niranjan, RBV Subramanyam, and V Khanaa. 2010. Developing a web recommendation system based on closed sequential patterns. In *International Conference on Advances in Information and Communication Technologies*. Springer, 171–179.

[20] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32, 8026–8037.

[21] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. 452–461.

[22] Liqiang Song, Ye Bi, Mengqiu Yao, Zhenyu Wu, Jianming Wang, and Jing Xiao. 2020. DREAM: A Dynamic Relation-Aware Model for Social Recommendation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2225–2228.

[23] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. 2019. Session-based social recommendation via dynamic graph attention networks. In *Proceedings of the Twelfth ACM international conference on web search and data mining*. 555–563.

[24] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph Attention Networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

[25] Shoujin Wang, Longbing Cao, Yan Wang, Quan Z Sheng, Mehmet A Orgun, and Defu Lian. 2021. A survey on session-based recommender systems. *ACM Computing Surveys (CSUR)* 54, 7 (2021), 1–38.

[26] Xiang Wu, Qi Liu, Enhong Chen, Liang He, Jingsong Lv, Can Cao, and Guoping Hu. 2013. Personalized next-song recommendation in online karaokes. In *Proceedings of the 7th ACM Conference on Recommender Systems*. 137–140.

[27] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.

[28] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. 2020. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 753–763.

[29] Z Wu, S Pan, G Long, J Jiang, and C Zhang. 2019. Graph WaveNet for Deep Spatial-Temporal Graph Modeling. In *The 28th International Joint Conference on Artificial Intelligence*. 1907–1913.

[30] Ghim-Eng Yap, Xiao-Li Li, and S Yu Philip. 2012. Effective next-items recommendation via personalized sequential pattern mining. In *International conference on database systems for advanced applications*. 48–64.

[31] Qi Zhang, Jianlong Chang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. 2020. Spatio-temporal graph structure learning for traffic forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 1177–1185.

[32] Tong Zhao, Julian McAuley, and Irwin King. 2014. Leveraging social connections to improve personalized ranking for collaborative filtering. In *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*. 261–270.

A SCALABILITY ANALYSIS

A.1 Time Complexity Analysis

The computation of GSL4Rec during model training consists of four parts. (1) In the coarse screening layer, we need to build a $N \times N$ similarity matrix by multiplying two embedding matrices. The time complexity of coarse screening is $O(N^2 d)$, where N is the number of total users, and d denotes the embedding size. The time complexity of $\text{argtop}_{k_1}(\cdot)$ operator for all N users is $O(N^2)$, which is omitted. (2) When inferring the dynamic interest of the user and her k_1 neighbors, we use an LSTM-based module to update the node's current status. Thus for all users, the time complexity would be $O(Nk_1md)$, where m denotes the average length of users' sessions. (3) The second-stage self-adaptive graph structure learning computes the scores of the edges between the target user and each of her neighbors based on a shallow network. The complexity for this part is $O(Nk_1 d)$. The time complexity of $\text{argtop}_{k_2}(\cdot)$ operator for all N users is $O(Nk_2)$, which can be omitted due to $k_2 < k_1$. (4) According to the relevant scores from the 2nd-stage, k_2 neighbors' interest will be aggregated to update the target user. The time to update all users will be $O(Nk_2 dL)$, where L denotes the number of the propagation layers. Hence, the overall theoretical time complexity for all user in one full step of model training is $O(N^2 d + Nk_1 md + Nk_1 d + Nk_2 dL)$. In practice, we set $k_2 < k_1 \ll N$ and usually $L \leq 3$. From the analyses above, we can find that the heaviest computation, in theory, is from the coarse screening layer, i.e., $O(N^2 d)$. However, it is not necessary to update the coarse neighbor screening very often in practice. Thus we propose to update it every epoch. This method will reduce the computational cost dramatically without much loss of performance. Also, we propose the phased heuristic learning strategy to accelerate further the model convergence, which is to be introduced in Section 5.2.

For serving, two additional strategies can be applied to reduce the computational cost further. (1) Approximate nearest neighbor searching algorithms [1] can be applied for coarse neighbor screening, which can reduce the time complexity to $O(d \log n)$ for a user query. (2) A user's short-term interest can be efficiently cached and reused and only updated when interacting with a new item.

Table 4: Computation Cost on Different User Scale. Some results are infeasible with limited resources.

User Scale	Training Time (s/epoch)			Memory Occupation (MiB)		
	GSL4Rec	**F-CN**	**F-SL**	**GSL4Rec**	**F-CN**	**F-SL**
2K	5	5	8	2327	2337	18855
10K	84	82	–	3047	3057	–
20K	327	319	–	5357	5365	–
50K	2053	2037	–	21429	20383	–

A.2 Computational Cost

Beyond the theoretical analysis, we compare the empirical computation cost of our GSL4Rec with its variants on the synthetic datasets with varied numbers of users. We introduce two variants: (1) F-CN, which selects the neighbors by fully relying on the first-stage coarse screening; (2) F-SL, which selects the neighbors by fully relying on the second-stage self-adaptive graph structure learning. We construct the synthetic datasets by randomly sampling 1,000 users alongside their interaction history from the *Yelp* dataset and replicate them different times to create datasets of different user scales. Table 4 shows the training time and the memory occupation with a GTX3090. The training time is greatly affected by the number of training instances (i.e., the number of interactions), while the number of users more influences the memory occupation.

From table 4, we can observe that the memory cost of GSL4Rec and F-CN are very close, which is consistent with our analysis in A.1 that the heaviest computation is from the coarse screening layer. Removing the coarse screening, F-SL directly performs the second-stage self-adaptive graph structure learning on all the users, with huge memory occupations even with a small user set. These observations prove that the proposed two-stage strategy can significantly address the scalability issue in real-world recommendation scenarios.

B DETAILED INFORMATION ABOUT THE DATASETS

Delicious. [1] This dataset contains social connections, bookmarking, and tag information from Delicious Social Bookmarking System. For session-based recommendations, we aim to provide tag recommendations for bookmarks. We define a session as a sequence of tags a user has assigned to a bookmark, which is different from the ordinary definition as a sequence of consumption in a given time window. We also converted the "trust" endorsements into directed edges in the social network.

Yelp. [2] This is an online location-based review system. Users make friends with others and express their experience (i.e., local businesses) through reviews and ratings. We construct our dataset by using each review as evidence of a user's consumption. Based on the empirical frequency of the offline consumption, we segment the data into month-long sessions following [23].

LibraryThing. [3] Lthing (for brevity) is a popular book-reviewing website that allows users to create an online catalog of their own or have read. We treat each review as an observation and segment users' behaviors into week-long sessions for their high activeness. Similar to Delicious, we also collect its explicit social network information.

Dianping. [4] This is a popular social network-based local restaurant search and review platform on which users can review restaurants and shops. We crawled the data using the users' identities, obtaining the business they reviewed along with associated timestamps and users' social relations.

Mind. [5] This dataset was collected from anonymized behavior logs of the Microsoft News website. We collected a set of impression logs in a week-long interval. For session-based recommendation tasks, we reserved the clicked news in each impression as a session and filtered out users who clicked on less than 10 articles.

C FORMULATION OF METRICS

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:

$$\text{NDCG@}K = \frac{1}{R_N} \sum_{i=1}^{N} \frac{2^{rel_i - 1}}{\log_2(1+i)}, \tag{15}$$

$$\text{HR@}K = \frac{\sum_{i=1}^{K} rel_i}{|\mathcal{I}|}, \tag{16}$$

where K is the size of the recommendation list, $rel_i = 0$ or 1 denotes whether the item 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. $|\mathcal{I}|$ is the number of all items.

D DETAILED INFORMATION ABOUT THE BASELINES

BPR-MF [21]. A competitive method improving matrix factorization (MF) with the BPR objective function for a implicit feedback-based recommendation.

SBPR [32]. A ranking model considering social relationships in the learning process, assuming that users tend to assign higher ranks to items that their friends prefer.

SoReg [17]. A method utilizing the social network to regularize the latent user factors of matrix factorization.

GRU4Rec [9]. A recent approach using recurrent neural networks for session-based recommendations.

NARM [13]. A method employing RNNs with an attention mechanism to capture the user's main purpose and sequential behavior.

[1] Data set available from https://grouplens.org/datasets/hetrec-2011/
[2] Data set available from https://www.yelp.com/dataset
[3] Data set available from https://cseweb.ucsd.edu/ jmcauley/datasets.html
[4] http://www.dianping.com
[5] Data set available from https://msnews.github.io/index.html

DGRec [23]. A method modeling dynamic user behaviors with RNNs, and social influence with graph attention neural networks based on given social networks.

DREAM [22]. A method utilizing a glove-based method to obtain users' virtual friends and employing a relational-GAT to update users' representations.