Hindawi
Computational Intelligence and Neuroscience
Volume 2022, Article ID 5484119, 11 pages
https://doi.org/10.1155/2022/5484119

*Research Article*

# Dynamic and Static Features-Aware Recommendation with Graph Neural Networks

**Ninghua Sun**,[1,2] **Tao Chen**,[1] **Longya Ran**,[1] **and Wenshan Guo**[1]

[1]*School of Public Administration, Huazhong University of Science and Technology, Wuhan 430074, China*
[2]*Innovation Institute, Huazhong University of Science and Technology, Wuhan 430074, China*

Correspondence should be addressed to Tao Chen; chentao15@163.com

Received 16 March 2022; Revised 27 March 2022; Accepted 28 March 2022; Published 21 April 2022

Academic Editor: Zhongxu Hu



Recommender systems are designed to deal with structured and unstructured information and help the user effectively retrieve needed information from the vast number of web pages. Dynamic information of users has been proven useful for learning representations in the recommender system. In this paper, we construct a series of dynamic subgraphs that include the user and item interaction pairs and the temporal information. Then, the dynamic features and the long- and short-term information of users are integrated into the static recommendation model. The proposed model is called dynamic and static features-aware graph recommendation, which can model unstructured graph information and structured tabular data. Particularly, two elaborately designed modules are available: dynamic preference learning and dynamic sequence learning modules. The former uses all user-item interactions and the last dynamic subgraph to model the dynamic interaction preference of the user. The latter captures the dynamic features of users and items by tracking the preference changes of users over time. Extensive experiments on two publicly available datasets show that the proposed model outperforms several compelling state-of-the-art baselines.

## 1. Introduction

The amount of information on the Internet continues to grow rapidly, and determining useful information has become increasingly difficult. Fortunately, the advancement of recommender systems can substantially help people deal with the information overload problem. Collaborative filtering (CF) is one of the most famous methods in recommendation algorithms. Therefore, collaborative learning latent representations of users and items from user-item interactions is an important step in CF-based models. However, poor latent representations of users and items remain the factors limiting further performance.

Therefore, researchers have adopted different methods to capture latent representations. Till now, the most commonly used approach for CF is to learn latent features in the embeddings space generated from the user-item rating matrix, such as matrix factorization [1] and deep learning-based CF [2–4]. Some researchers [5, 6] use a bipartite graph to represent user-item interactions to further enhance the latent representations; hence, the topological features of the graph are introduced through graph neural networks (GNNs) [7]. The underlying assumption in leveraging the bipartite graph as input to obtain effective recommendations is as follows: nodes that are connected can spread information by aggregating their neighbors, thereby potentially contributing to capturing high-order features.

Latent representations obtained from dynamic user-item interactions serve as another method. Traditional CF usually defines a decay function of temporal information [8, 9], such as the exponential decay function $e^{-\omega t}$, to capture these dynamic features, while graph-based CF obtains a series of user-item bipartite graphs based on interaction time [10]. The underlying assumption in using temporal information is that the behaviors of users on items are a dynamic interactive process; consequently, the long- and short-term preferences of users are captured.

It is unclear, however, which of these approaches—static recommendation versus dynamic recommendation—is better for predicting user preference on items. The former ignores that user preference is dynamic, thus changing over time. The latter usually requires more parameters and training time than the static recommendation, which limits its application. Furthermore, the introduction of temporal information may bring additional noise, which can hinder the performance and scalability of the model. Two important problems must be solved to deal with these challenges:

(1) How to represent the behaviors of users with a dynamic graph? Temporal information is vital for capturing the dynamic preference of users. To avoid introducing additional noise data, utilizing temporal information data more efficiently should be priorities.

(2) How to obtain the dynamic features simply and swiftly? In addition to the interaction pairs of users and items, the dynamic graph also includes side information (e.g., temporal information). However, additional information will introduce an increase in the number of parameters and high computational complexity.

To this end, a simple and effective graph-based algorithm is proposed to introduce dynamic features into a static recommender system called dynamic and static features-aware graph recommendation (DSAGR). Firstly, rather than simply timestamps, the dynamic graph of users and items is constructed based on Takens' time embedding theorem [11] to use temporal information efficiently. This work employs the graph convolution network (GCN) [7] to learn the long- and short-term preferences of users because of the expressive graph-based models. Then, a novel module is proposed, that is, the dynamic sequence learning module, to transform the unstructured dynamic graph to structured sequence data to decrease the dynamic model complexity. In particular, convolutional neural networks (CNNs) [12, 13] are used to capture the dynamic features from the sequence data. Finally, the dynamic features and dynamic preference are integrated to obtain the predictor for each user. Our main contributions are as follows.

(1) This work can simply and swiftly capture the dynamic features from the constructed dynamic graph.

(2) A novel hybrid model is proposed in this work, which can easily capture the users' dynamic preferences.

(3) An offline experiment is performed on real-world datasets. The results show that the proposed model successfully performs the personalized recommendation task.

The rest of the paper is organized as follows. Section 2 elaborates on relevant research. Section 3 presents the proposed method, while Section 4 discusses the empirical study on the public datasets. Finally, Section 5 contains the conclusions.

## 2. Related Work

Collaborative filtering- (CF-) based recommendation aims to predict the preference of users and then return top-$N$ items of the user interests. Heuristic works, such as item- and user-based models, predict the preferences of users on items based on the $k$-nearest neighbor algorithm [14]. Model-based approaches usually learn the user and item with low-rank latent representations through matrix factorization [1]. The inner product between the two lower-rank vectors is then used to obtain the probability of the user clicking on the item.

Furthermore, deep learning has been shown to be particularly well suited to representation learning tasks [15, 16]. Therefore, many deep learning techniques have recently allowed CF to have expressive representation vectors from the historical behaviors of users, such as the multilayer perceptron (MLP), autoencoder (AE), recurrent neural network (RNN), and CNN. Many researchers often consider the combinations of matrix factorization and deep learning techniques for CF recommendation. MLP-based model Wide&Deep [17] captures linear and nonlinear latent features effectively. NeuMF [2] integrates MLP and MF to model high- and low-order interaction features. Furthermore, AE is used for recommendation tasks. Work [18] employs a denoising recurrent AE network and then generalizes it to the CF setting. RNN has been widely used for recommendation due to its excellent performance in modeling sequential data. The variants of RNN, such as long short-term memory (LSTM) [19]and gated recurrent unit (GRU) networks [20], are often employed in practice to overcome the vanishing gradient problem. For instance, work [21] uses LSTM to model the long- and short-term preferences of users. CNN is also a powerful tool [15]. Work [22] uses two parallel CNNs to learn deep representations of users and items. Work [23] integrates CNN and GRU networks to obtain distributed representations of users and items. These representations are then used to regularize the generation of latent features in matrix factorization.

In the last few years, GNN has been widely recognized as a state-of-the-art approach because of its successful applications in recommendation tasks [24]. GNN can effectively learn the structural representations of nodes by aggregating their neighborhood information. A pooling operation is typically used to output the node embeddings after an aggregation function. Many graph-based models are also proposed by using different aggregation and pooling functions such as GCN [25], GraphSAGE [26], and Graph Isomorphism Network (GIN) [27]. Among these models, the most popular recommendation method is LightGCN [6] coupled with NGCF [5]. LightGCN is an effective simplified version of the NGCF by omitting the transformation mechanism and applying the sum-based pooling layer. Some researchers also consider dynamic representation learning to model data. Work [28] employs matrix perturbation to model the changes in graphs, such as the adjacency matrix. Work [29] constructs the user-item interaction graph dynamically based on the users and items embeddings to improve the diversity of recommendations. Furthermore,

many graph-based algorithms [24] have been proposed to enrich the presentation of users and items with other auxiliary information [30, 31]. Therefore, this work tries to introduce dynamic features as side information to improve the recommendation performance.

These graph-based models have verified their superiority for the recommendation task. However, these models mainly focus on constructing static graph-based recommendation models without considering their combinations with dynamic graph features. As far as we know, there is no study to introduce dynamic graph features into a graph-based recommendation framework.

## 3. Proposed Method

In this part, the proposed DSAGR method is presented, and its framework is illustrated in Figure 1. Four components are included in the framework: (1) dynamic graph construction aims to convert the behaviors of users into a dynamic graph; (2) dynamic preference learning module is to learn the long- and short-term preferences of users; (3) dynamic sequence learning module aims to capture the user and item sequence features as side information; and (4) prediction layer is to obtain the predictors.

*3.1. Dynamic Graph Construction.* Given a user set $U$, an item set $I$, and a set of time stamps $T = \{t_1, t_2, t_3, \cdots\}$, the graph of the user-item interaction at the time stamp $t_1$ can be defined as $G_{t_1} = (U \cup I, \mathscr{E}_{t_1})$, where $U \cup I$ is the set of nodes, and edge $e \in \mathscr{E}_{t_1}$ represents the interaction between the user and the item at the time $t_1 \in T$. Therefore, the interactions of users and items can be seen as a time series, that is, $\{G_{t_1}, G_{t_2}, G_{t_3} \cdots\}$. Figure 2(b) shows different graphs at five different time stamps.

To understand the behaviors of users with the effects of temporal information, several time slices of user-item interactions are generated based on Takens' time embedding theorem [11] using a given delay factor. Considering the following example: given five timestamps [1–5], we assume that the delay factor is equal to 1 and the number of the time slices is 4. Takens' time embedding theorem indicates that the time series is embedded into $R^2$ vector space as follows: [[1, 2], [2, 3], [3, 4], [4, 5]]. Similarly, the user-item interaction time stamps can also be embedded into the vector space and further be divided into $l$ time slices $[T_1, T_2, T_3 \cdots, T_l]$. Therefore, an interaction graph for each time slice can be obtained as previously mentioned. More formally, the obtained interaction subgraphs are denoted as $\{G_{T_1}, G_{T_2}, G_{T_3}, \cdots, G_{T_l}\}$.

Figure 2 demonstrates the specific processes. Figure 2(a) presents the example dataset, which is ranked based on the interaction time in the order from small to large. For the sake of convenience, the interaction time is indicated by numbers 1–5. In (b), the user-item interaction graph at different timestamps can be observed. These user nodes are marked dark red, and item nodes are marked lilac color. In (c), Takens' embedding of temporal information generates three time slices marked by orange color (i.e., $T_1$: [1–3], $T_2$: [2–4], and $T_3$: [3–5]), in which the element is the time ID. The interacted pairs in each time slice constitute a user-item interaction subgraph.

*3.2. Dynamic Preference Learning Module.* The upper part in Figure 1 shows the dynamic preference learning module. In the recommendation task, the long-term preference of users reflects their inherent features and general preference, which can be learned from all interacted items of users. The short-term signals of the user reflect his/her latest preference. Furthermore, many studies [32] use the latest interaction item embedding and the latest timestamp as short-term information but ignore the dependence on historical interactions. The long- and short-term collaboration can be captured effectively by considering the same layer structure with Siamese and information sharing components [33] on all interaction graph $G$ and the last subgraph $G_{T_l}$. Siamese networks can naturally introduce inductive biases for invariance modeling because of identical weight-sharing subnetworks. Then, the two graphs can be parameterized using a GNN layer, such as LightGCN [6]. To offer a holistic view of the long-term and short-term collaborative nodes embeddings, we provide the matrix form.

Long-term:

$$\begin{aligned} \begin{pmatrix} L_u \\ L_i \end{pmatrix} &= \left(D^{-1/2} A D^{1/2}\right) \begin{pmatrix} E_u \\ E_i \end{pmatrix}, \\ \begin{pmatrix} E_{\text{long},u} \\ E_{\text{long},i} \end{pmatrix} &= \lambda_0 \begin{pmatrix} E_u \\ E_i \end{pmatrix} + \lambda_1 \begin{pmatrix} L_u \\ L_i \end{pmatrix}, \\ A &= \begin{pmatrix} M & 0 \\ 0 & M^T \end{pmatrix}, \end{aligned} \tag{1}$$

where $M \in R^{|U| \times |I|}$ is the user-item rating matrix, in which each element $M_{u,i}$ is 1 if the user $u$ interacted with the item $i$; otherwise, it is 0. Then, $A$ is the adjacency matrix of the graph $G$; $D$ is a $(|U| + |I|) \times (|U| + |I|)$ diagonal matrix, in which each entry $D_{jj}$ is the number of nonzero entries in the $j$th row vector of the adjacency matrix $A$; $E_u \in R^{|U| \times d}, E_i \in R^{|I| \times d}$ are the initial weight matrix of users and items, respectively. $\begin{pmatrix} E_u \\ E_i \end{pmatrix}$ denotes the concatenation of $E_u$ and $E_i$ $\lambda_0, \lambda_1 \in [0, 1]$, are the defined hyperparameters; $E_{\text{long},u}$ and $E_{\text{long},i}$ are the final representations of users and items for learning long-term preferences.

Short-term:

$$\begin{aligned} \begin{pmatrix} S_u \\ S_i \end{pmatrix} &= \left(D_{\text{last}}^{-1/2} A_{\text{last}} D_{\text{last}}^{1/2}\right) \begin{pmatrix} E_u \\ E_i \end{pmatrix}, \\ \begin{pmatrix} E_{\text{short},u} \\ E_{\text{short},i} \end{pmatrix} &= \lambda_0 \begin{pmatrix} E_u \\ E_i \end{pmatrix} + \lambda_1 \begin{pmatrix} S_u \\ S_i \end{pmatrix}, \end{aligned} \tag{2}$$

where $A_{\text{last}}$ is the adjacency matrix of the latest subgraph $G_{T_l}$; $D_{\text{last}}$ is also the diagonal matrix calculated based on $A_{\text{last}}$. $E_{\text{short},u}$ and $E_{\text{short},i}$, respectively, denote the final representation of users and items in the short-term preference learning.

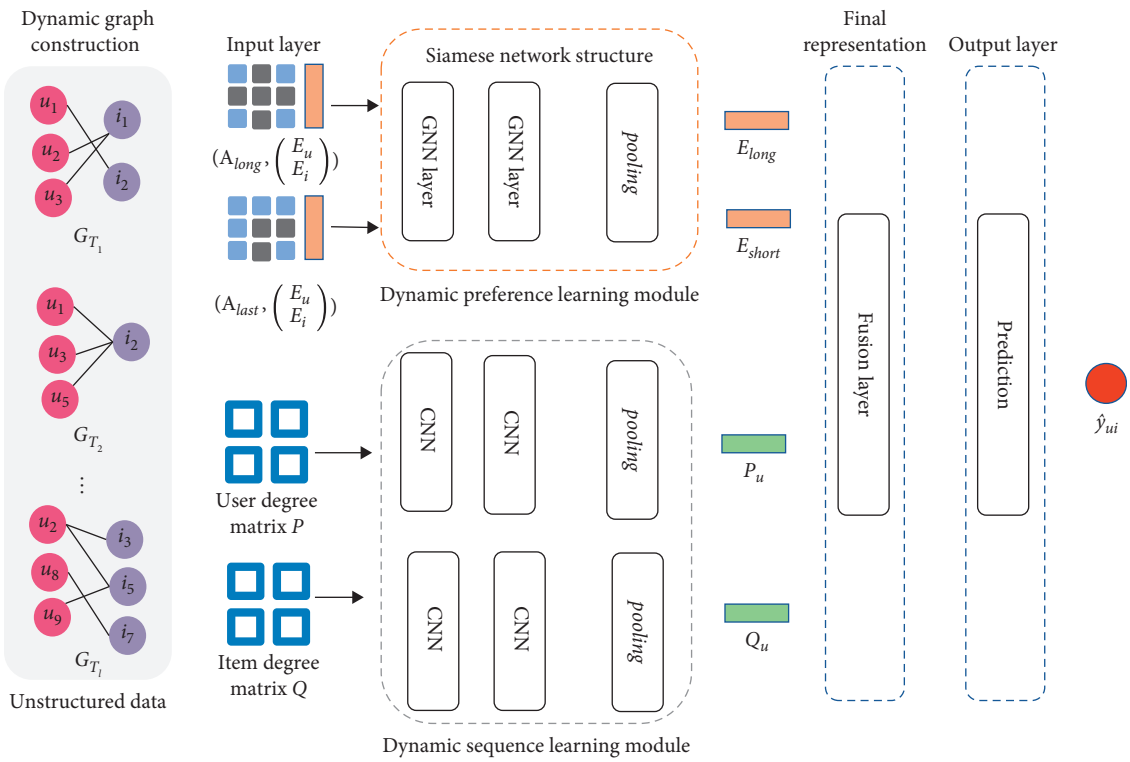


Figure 1: Framework of our model.

| Interaction time ID | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| User ID | 1 | 1 | 2 | 1 | 3 |
| Item ID | 1 | 2 | 1 | 3 | 1 |

(a)

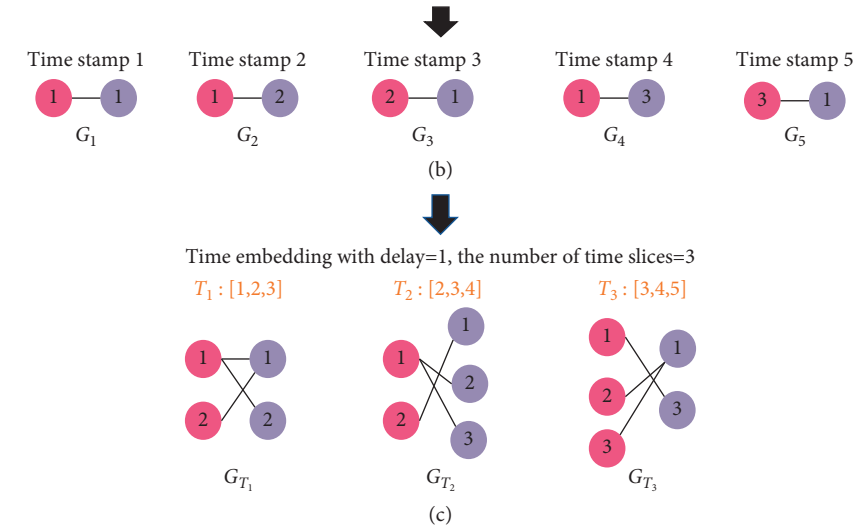


Figure 2: The construction of the dynamic graph. (a) Dataset. (b) The graph of user-item interaction at different time stamps. (c) Dynamic subgraphs.

*3.3. Dynamic Sequence Learning Module.* The degree of the graph is shown to be effective for evaluating the popularity of nodes [34, 35]. Therefore, the degree matrixes of users and items are proposed to track the dynamic changes in the user and item nodes in constructed dynamic graph $\{G_{T_1}, G_{T_2}, G_{T_3}, \ldots, G_{T_l}\}$, respectively. For instance, the degree matrix of items is denoted as $Q = \{q_1, q_2, \cdots, q_l\}$, in which element $q_k$ is a $|I|$ dimensional vector, its $i$th element $q_{i,k}$ is the number of edges incident to the item $i$ in the $k$th subgraph $G_{T_k}$. And the element $q_{i,k}$ means the popularity of item $i$ in the time slice $T_k$. Similarly, the user degree matrix is denoted as $P = \{p_1, p_2, \ldots, p_l\}$, where $p_l \in R^{|U|}$. Therefore, this study offers a novel means of processing the unstructured graph data and hence may shed light on the task of graph-based recommendation.

The lower part in Figure 1 shows the dynamic sequence features learning modeled by two parallel CNN layers. The input of the module is the obtained user degree matrix and item degree matrix $P \in R^{|U|\times l}, Q \in R^{|I|\times l}$. The CNNs generally comprise a set of convolutional and pooling layers in their architectures. In this work, two 1D-convolutional layers and one pooling layer are designed to learn dynamic features. The first and second convolutional layers with a set of $\{f_1, f_2\}$ filters with the kernel size of $\tau$, shared weights $w_1 \in \mathbb{R}^{f_1\times 1\times\tau}, w_2 \in \mathbb{R}^{f_2\times f_1\times\tau}$, as shown in the following equations.

$$g_t = p_t * w_1, \tag{3}$$

$$h_t = g_t * w_2, g_t \in R^{|U|\times f_1}, \tag{4}$$

where $p_t \in R^{|u|}$ is the column vector of user degree matrix $P \in R^{|U|\times l}$, $*$ is the convolution operator, and $h_t \in R^{|U|\times f_2}$ denotes a feature matrix for all users. After the 1D-convolutional operation, $l$ feature matrixes can be obtained. Inspired by graph-based models [5, 6], the weighted sum operator is designed as the pooling layer and then normalized by the sigmoid function $\sigma$. The output is shown in the following equation.

$$P_u = \sigma(h_1 + h_2 + \cdots + h_l). \tag{5}$$

Analogously, the items' degree representations are defined as follows.

$$\begin{aligned} g'_t &= q_t * w'_1,\ t = 1, 2, \ldots, l, \\ h'_t &= g'_t * w'_2,\ g'_t \in R^{|I|\times f_1}, t = 1, 2, \ldots, l, \\ Q_i &= \sigma\left(\sum_{t=0}^{l} h'_t\right), h'_t \in R^{|I|\times f_2}, \end{aligned} \tag{6}$$

where $q_t \in R^{|I|}$ is the column vector in item degree matrix $Q$, $*$ is the convolution operator, and $w'_1 \in \mathbb{R}^{f_1\times 1\times\tau}$, $w'_2 \in \mathbb{R}^{f_2\times f_1\times\tau}$ are shared factors.

*3.4. Prediction Layer.* The embeddings and degree representations of nodes of the user and item are obtained after the dynamic preference learning module and dynamic sequence learning module. Then, a fusion layer is defined to learn the final representations:

$$E^*_u = \left(E_{\text{long},u} + E_{\text{short},u}, P_u\right), E^*_i = \left(E_{\text{long},i} + E_{\text{short},i}, Q_i\right), \tag{7}$$

where (,) is the concatenation operator.

Thereafter, we use an inner product on the final embedding of the users and items to predict the recommendation results. The formula is as follows:

$$\left(\widehat{y}_{u,i}\right)_{|U|\times|I|} = E^*_u \left(E^*_i\right)^T. \tag{8}$$

*3.5. Training.* In this work, the Bayesian Personalized Ranking (BPR) loss [36] is used, which is a pairwise loss that encourages the prediction of an interacted entry to be higher than its uninteracted counterparts:

$$L_{BPR} = \sum_{(u,i,j)\varepsilon O} -\ln\ \sigma\left(\widehat{y}_{u,i} - \widehat{y}_{u,j}\right), \tag{9}$$

where $O = \{(u, i, j) | (u, i) \in O^+, (u, i) \in O^-\}$, is the dataset in the training process, which consists of interacted pairs set $O^+$ and uninteracted pairs set $O^-$. What is more, $L_2$ regularization is used to optimize the model parameter to prohibit overfitting risk. Therefore, the final objective function in our model is combined by BPR loss and regularization:

$$L_{our} = L_{BPR} + \gamma_1 \left\|\Theta_1\right\|_2^2 + \gamma_2 \left\|\Theta_2\right\|_2^2, \tag{10}$$

where set $\Theta_1 = \{E_u, E_i\}$ is the set of embedding parameters, $\Theta_2 = \{w_1, w_2, w'_1, w'_2\}$ is the set of weights in CNN layers, and $\gamma_1, \gamma_2$ are the hyperparameters to control the regularization. Furthermore, the Adam [37] is used in a minibatch manner to optimize the proposed model.

## 4. Experiments

Empirical results are proposed to evaluate the proposed model. The experiments aim to answer the following research questions:

RQ1: How does DSAGR perform as compared with state-of-the-art models?

RQ2: How do dynamic features affect DSAGR?

RQ3: What are the effects of hyperparameters on the DSAGR model?

*4.1. Dataset.* The dynamic preference learning module in the proposed method requires implicit feedback and temporal information; thus, the proposed model is evaluated on ML_100k and ML_1M movie datasets (https://grouplens.org/datasets/movielens/). Table 1 presents the statics of the datasets. These datasets have 5-level rating scores, and each user has rated at least 20 movies. The ratings of the datasets are binarized because the proposed model only requires implicit feedback. Specifically, every element in the original rating matrix (scores 1 to 5) is binarized to 1 and 0, where 1 indicates that the rating score is not less than 4, 0 indicates the rating score is less than 4, and no interaction. This work

Table 1: Statistics of the datasets.

| Statistics | ML_100k | ML_1M |
|---|---|---|
| Number of users | 944 | 6040 |
| Number of items | 1683 | 3952 |
| Number of ratings | 100000 | 1000209 |

also follows the same settings described in NGCF [5] to select 20% of interaction recodes randomly from each user to represent the test and valid sets and then treat the remaining as the training set.

### 4.2. Experimental Settings

*4.2.1. Baselines.* The GSAGR model is compared with the following methods:

(i) Item-based CF (ICF) [38]: ICF is usually a two-step process: (1) determining the similarity set for target items and (2) predicting rating scores based on the most similar items. The rating scores of unseen items for the user are predicted in the second phase according to the weighted average rating of his $k$-nearest neighbor.

(ii) PMF [1]: With the probabilistic matrix factorization (MF) algorithm, this model maps the user-item rating matrix into two low-dimensional matrixes. Then, this algorithm predicts the preference of users by the inner product between the two low-dimensional matrixes.

(iii) DMF [39]: DMF is an MF-based CF method, which obtains the latent features of users and items through deep representation learning, that is, MLP. This method then uses the inner product between the two latent features to predict the preferences of users on items.

(iv) Wide&Deep [17]: Wide&Deep is a famous deep learning recommender system that combines wide linear models and MLP neural network layers to obtain latent representations of users and items.

(v) NGCF [5]: This work learns the representations of users and items by aggregating the information of high-order neighbors. Specifically, each node obtains the transformed representation of neighbors by propagating embeddings on the bipartite graph structure. NGCF introduces collaborative signals in the pooling layer to enhance high-order latent features learning.

(vi) DGCF [40]: DGCF is an advanced graph-based CF model. This work focuses on the intentions of users for interacting with different items. The implementation of DGCF is based on the NGCF and graph attention network to model different intents of users.

*4.2.2. Evaluation Metrics.* Unlike the previous studies [17, 39] that perform metrics from sampled uninteracted items, this experiment conducts metrics for all the items that the user has not interacted with. Two widely used evaluation protocols Recall@$N$ and NDCG@$N$ (normalized discounted cumulative gain) ($N = 20$ by default) are adopted to evaluate the effectiveness of top-$N$ recommendation and preference ranking. The specific formula is as follows:

$$\text{Recall@}N = \frac{1}{|U|} \sum_{u} \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{11}$$

where TP (i.e., Ture Positive) indicates the number of items in the top-$N$ recommendation list that hit the target items and FN (i.e., False Negative) is the number of the positive items in the test set that are falsely identified as the negative items.

$$N\ DC\ G@N = \frac{1}{|U|} \sum_{u} \frac{DC\ G@N}{I\ DC\ G@N}, \tag{12}$$

where $DC\ G@N = \sum_{i=1}^{N} 2^{r_i} - 1/\log_2(i+1)$; here, $r_i = 1$ if the test item is in position $i$, else 0; $I\ DC\ G@N$ indicates the ideal $DC\ G@N$ such that the target items are present at the top of the recommendation list.

*4.2.3. Parameter Settings.* The DSAGR model is implemented in Python under the TensorFlow (https://www.tensorflow.org) framework. For comparison algorithms, the parameter settings are given in the original works of literature. The proposed model uses the following parameter settings: (1) a random normal distribution (the mean and standard deviation are set to 0 and 0.01, resp.) is used to initiate the embedding matrix of users and items. Furthermore, the dimensionality of the embedding matrix is set to 64; (2) the delay factor for dynamic graph construction is set to one three-hundredth of the length of the dataset. (3) GCN and pooling layers with the hyperparameter $\lambda_0 = \lambda_1 = 1$ are used to represent the interaction features of users and items; (3) two GCN layers with 2 and 32 filter factors are used, and the kernel size in each layer is 3; (4) following NGCF [5] and DGCF [40], Adam optimization is used to train the model. The learning rate of the Adam algorithm is 0.0003, which is set by experiments.

### 4.3. Results and Discussion

*4.3.1. Performance Comparison (RQ1).* To answer the first research question, the proposed model is compared with six other methods in terms of Recall@$N$ and NDCG@$N$. Two of the methods, ICF, and PMF are traditional and are frequently used CF algorithms. DMF and Wide&Deep are deep learning-based CF models. The remaining two, referred to as DGCF and NGCF, are versions of GCN with graph structure. Table 2 reports the performance for each algorithm. The following observations from this table are presented.

(1) This table reveals that DSAGR has achieved the best result on ML_100k and ML_1M datasets. On metrics Recall@$N$ and NDCG@$N$, DSAGR has improved the performance by at least 2.72% and 4.81%, respectively. For instance, DSAGR improves the NDCG@$N$ over the strongest baselines by 5.798% and 4.81% on

Table 2: Performance comparison of different methods.

| Datasets / Metrics | ML_100k | | ML_1M | |
|---|---|---|---|---|
| | Recall@$N$ | NDCG@$N$ | Recall@$N$ | NDCG@$N$ |
| ICF | 0.1642 | 0.2913 | 0.1504 | 0.2030 |
| PMF | 0.2303 | 0.3061 | 0.1943 | 0.2301 |
| DMF | 0.3396 | 0.3562 | 0.2215 | 0.2472 |
| Wide&Deep | 0.3402 | 0.3797 | 0.2502 | 0.2613 |
| NGCF | 0.3487 | 0.4225 | 0.2637 | 0.2947 |
| DGCF | 0.3339 | 0.4057 | 0.2973 | 0.3264 |
| Ours | 0.3672 | 0.4470 | 0.3054 | 0.3421 |

ML_100k and ML_1M, respectively. DSAGR can employ the dynamic information simply to provide additional side information for prediction by constructing the degree matrix. Meanwhile, DGCF and NGCF only aggregate the presentations of adjacent nodes in the graph. Significantly, DGCF uses multi-intent-aware graphs but performs worse than the proposed DSAGR model. The reason is that DGCF ignores the dynamic features and the short-term collaborative signals.

(2) DGCF, NGCF, DMF, and Wide&Deep achieve better performance than traditional methods PMF and ICF. Therefore, compared with the CF only, the employment of deep learning and GCN is advantageous across the board. Wide&Deep and DMF fail to go beyond NDCG@$N$ despite outperforming DGCF in Recall@$N$ on the ML_100k dataset, implying that the graph-based methods are more effective than the deep learning methods in modeling the preference of users. In particular, the NGCF model improves Recall@$N$ compared with the Wide&Deep model, with enhancements of 2.44%, 5.40%, on ML_100k and ML_1M datasets, respectively. The NGCF model also outperforms the DMF, with at least improvements of 2.68% and 18.61% on Recall@$N$ and NDCG@$N$, respectively. Such improvement might be attributed to the GCN module, which captures more complex behavior patterns than MLP.

Furthermore, the experiment is repeated for all methods 10 times. Therefore, the freedom degree of $t$-distribution is 9. Specifically, this experiment accepts the hypothesis that DSAGR achieves better performance than baseline models on the two datasets for significance levels of 0.005. The statistical tests and results for this analysis are shown in Table 3. This table reveals that the method DSAGR successfully enhances the representation of users and items by considering the dynamic features and preferences.

*4.3.2. Effect of the Proposed Technologies (RQ2).* The proposed DSAGR is compared with different variants on the ML_100k and ML_1M datasets to investigate the superiority of the key technologies proposed in this work. Table 4 reports the variant models and their performances. The following findings are presented: DSAGR-L performs better than DSAGR-S, which removes the long-term information. This finding is probably because the preference of users cannot be captured by the short-term information alone. DSAGR-DL, DSAGR-DG, and DSAGR also outperform DSAGR-D and DSAGR-G. This phenomenon proves that the captured dynamic features and short-term information can effectively improve the model's performance. Moreover, DSAGR performs better than GRU-based [20] variant DSAGR-DG and LSTM-based [19] variant DSAGR-DL. This result is probably due to the small length of the row vectors of the dynamic matrix, allowing the CNN to model their dynamic features effectively.

*4.3.3. The Sensitivity of Hyperparameters (RQ3).* This work investigates how four hyperparameters, namely, the number of time slices, the filter factors in the first and second CNN layers, and the embedding size, affect DSAGR to examine the effect of the constructed dynamic graph among dynamic preference of users. The experiments on two datasets are conducted, providing similar rules, and only the results on the ML_100k are presented herein.

Inspired by the work [41], the experiment also adopts the orthogonal experimental design (OED) method to get a reasonable combination of these hyper-parameters. Specifically, the number of levels for the four parameters is set as follows: four levels for the time slices {11, 21, 31, 41}; four levels for the filter factors in the first CNN layer {2, 4, 6, 8}; four levels for the filter factors in the second CNN layer {8, 16, 32, 64}; and four levels for the embedding factors {16, 32, 64, 72}. A full-factorial analysis needs $4^4 = 256$ experiments. Taguchi's method employs the orthogonal arrays to obtain the possible combinations of the hyperparameters from the whole combinations, thus bringing a minimum experimental run and the best estimation of parameters during the execution. In our experiments, the orthogonal array $L_{16}(4^4)$ has only 16 experiments, as shown in Table 5. This table shows that DSAGR can achieve better performance by setting time slices as 31, filter factors in the first and second CNN layers as 2 and 32, and embedding factors as 64.

The average values of Recall@$N$ are used to investigate the effect of each factor. For example, the mean value of the first 4 rows in Table 5 is calculated to investigate the effect of time slice with level 11. The average values of Recall@$N$ with different factors are shown in Figures 3–6.

Table 3: The *t*-test for paired comparisons in terms of Recall@*N* on the datasets.

| ICF | PMF | DMF | Wide&Deep | NGCF | DGCF |
|---|---|---|---|---|---|
| *ML_100k* | | | | | |
| 337.98 | 181.01 | 17.19 | 14.98 | 12.88 | 57.77 |
| <0.005 | <0.005 | <0.005 | <0.005 | <0.005 | <0.005 |
| *ML_1M* | | | | | |
| 305.99 | 141.38 | 225.11 | 73.34 | 67.96 | 14.95 |
| <0.005 | <0.005 | <0.005 | <0.005 | <0.005 | <0.005 |

Table 4: Performance of compared with different variants of DSAGR ("—" indicates DSAGR removes the key technology).

| Variants | Dynamic preference | | Dynamic feature | ML-100k | | ML_1M | |
|---|---|---|---|---|---|---|---|
| | Long-term | Short-term | | Recall@*N* | NDCG@*N* | Recall@*N* | NDCG@*N* |
| DSAGR-S | — | GCN | CNN | 0.36013 | 0.44148 | 0.2891 | 0.3214 |
| DSAGR-L | GCN | — | CNN | 0.36370 | 0.32779 | 0.3000 | 0.3325 |
| DSAGR-G | GCN | — | — | 0.33217 | 0.4041 | 0.2345 | 0.2911 |
| DSAGR-D | GCN | GCN | — | 0.36006 | 0.44097 | 0.2798 | 0.3154 |
| DSAGR-DL | GCN | GCN | LSTM | 0.36184 | 0.44274 | 0.2921 | 0.3255 |
| DSAGR-DG | GCN | GCN | GRU | 0.36456 | 0.44921 | 0.2966 | 0.3310 |
| DSAGR | GCN | GCN | CNN | 0.3672 | 0.4470 | 0.3054 | 0.3421 |

Table 5: Performance of 16 experiments obtained from Taguchi's method.

| ID | Time slices | Filter-first CNN layer | Filter-second CNN layer | Embedding factors | Recall@*N* | NDCG@*N* |
|---|---|---|---|---|---|---|
| 1 | 11 | 2 | 8 | 8 | 0.3422 | 0.4200 |
| 2 | 11 | 4 | 16 | 16 | 0.3552 | 0.4369 |
| 3 | 11 | 6 | 32 | 32 | 0.3617 | 0.4420 |
| 4 | 11 | 8 | 64 | 64 | 0.3540 | 0.4296 |
| 5 | 21 | 2 | 16 | 32 | 0.3560 | 0.4373 |
| 6 | 21 | 4 | 8 | 64 | 0.3626 | 0.4412 |
| 7 | 21 | 6 | 64 | 8 | 0.3451 | 0.4224 |
| 8 | 21 | 8 | 32 | 16 | 0.3438 | 0.4186 |
| 9 | 31 | 2 | 32 | 64 | 0.3672 | 0.4470 |
| 10 | 31 | 4 | 64 | 32 | 0.3660 | 0.4481 |
| 11 | 31 | 6 | 8 | 16 | 0.3463 | 0.4237 |
| 12 | 31 | 8 | 16 | 8 | 0.3469 | 0.4259 |
| 13 | 41 | 2 | 64 | 16 | 0.3547 | 0.4232 |
| 14 | 41 | 4 | 32 | 8 | 0.3483 | 0.41786 |
| 15 | 41 | 6 | 16 | 64 | 0.3533 | 0.4217 |
| 16 | 41 | 8 | 8 | 32 | 0.3491 | 0.4286 |

(1) *Effect of Time Slice Numbers*. The number of time slices determines the number of dynamic subgraphs and the last subgraph. As shown in Figure 3, DSAGR can achieve the best performance by setting the time slice as 31. Figure 3 shows that when the number of time slices reaches 31, adding more time slices cannot improve the recommendation performance. Also, more time slices will increase the dimension of the row vectors of the degree matrix, and consequently, there will be an increase in the training time taken.

(2) *Effect of Filter Factors in CNN*. Figures 4 and 5 show the recommendation performance of different filters in the first and second CNN layers. Figure 5 reveals that the performance gradually becomes better with the increase of filter factors in the second CNN layer. However, blindly increasing the filter factors does not necessarily improve the performance of DSAGR. This is maybe because that more information is encoded when the filter factors become larger, but it may also bring a little overfitting.

(3) *Effect of Embedding Factors*. Figure 6 illustrates the performance of DSAGR under the different embedding factors. The figure reveals that the performance of the model gradually improves as the dimensionality increases. And the performance tends to be stable when the embedding factors are set as 64.

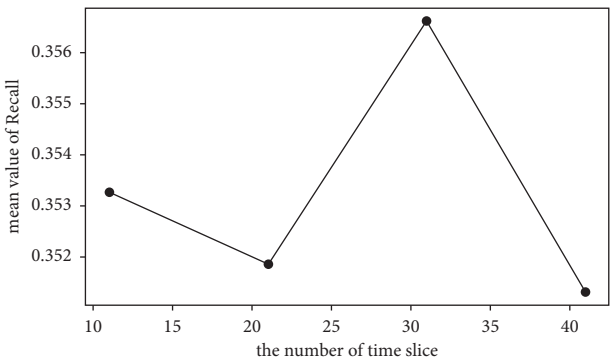


Figure 3: Effect of time slice numbers.

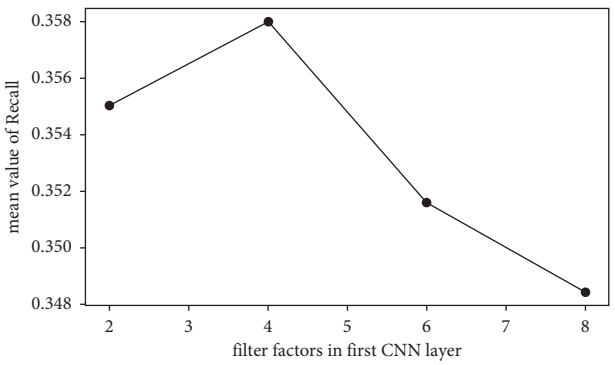


Figure 4: Effect of filter factors in the first CNN layer.

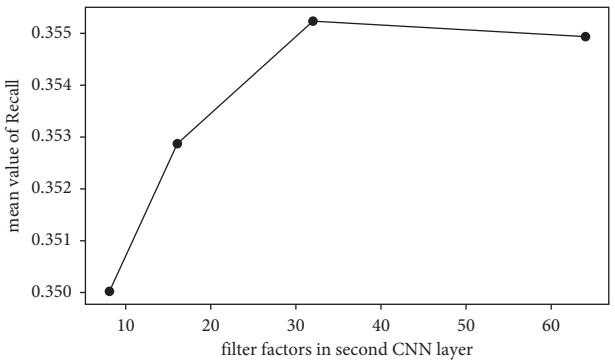


Figure 5: Effect of filter factors in the second CNN layer.

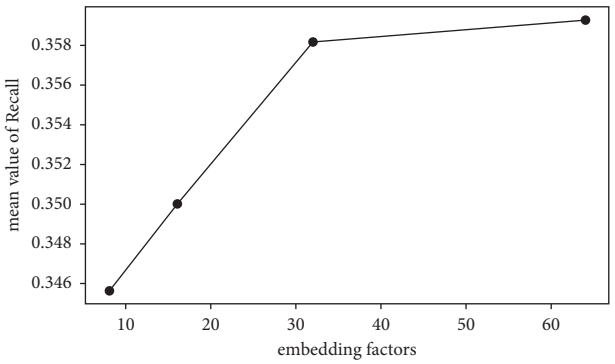


Figure 6: Effect of embedding factors.

## 5. Conclusion

In this work, a hybrid recommender system is proposed to capture the dynamic preferences of users and dynamic sequence features. The proposed model, namely, DSAGR, combines GCN and CNN to obtain the latent representations of users and items and then makes a prediction. The dynamic preference is modeled by the long- and short-term interaction graphs of users. The dynamic sequence includes the degree matrixes of users and items captured from the dynamic graph. To our knowledge, this type of modeling using the short-term interaction graph and degree matrixes has not been previously applied to predict users' preferences. The experimental results show that the DSAGR model significantly improves the performance compared with baselines. Considering future work, two feasible avenues are available: (1) The work concentrates on learning the latent representation of users and items via dynamic information. Thus, one direction of further study is to design an effective way to aggregate the long- and short-term representations to a single vector, which is successful in maximizing deep learning. (2) The method of capturing dynamic features provides a new idea to many other unstructured data, such as social networks. It is worth trying to improve the recommendation performance.

## Data Availability

The data used include ML_100k and ML_1M movie datasets. The movie dataset address is as follows: https://grouplens.org/datasets/movielens/.

## Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this paper.

## Authors' Contributions

Tao Chen and Ninghua Sun conceived and designed the experiments; Ninghua Sun proposed the method and performed the experiments; Longya Ran analyzed the data; Ninghua Sun prepared the original draft; and Longya Ran and Wenshan Guo reviewed and edited the manuscript. All authors have read and agreed on the published version of the manuscript.

## Acknowledgments

This work was supported by the Fundamental Research Funds for the Central Universities, China (HUST: 2020JYCXJJ036), Humanities and Social Science Fund of the Ministry of Education of China (19YJA630010), National Natural Science Foundation of China (71734002 and 72042016), and Key R&D Projects, Hubei Province (2021BAA033).

## References


[1] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *Proceedings of the 20th International Conference on Neural Information Processing Systems*, pp. 1257–1264, Red Hook, NY, USA, 2007.

[2] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th International World Wide Web Conference, WWW 2017, International World Wide Web Conferences Steering Committee*, pp. 173–182, Republic and Canton of Geneva, Geneva, Switzerland, 2017.

[3] Y. Guo and Z. Yan, "Recommended system: attentive neural collaborative filtering," *IEEE Access*, vol. 8, 2020.

[4] H. Liu, Y. Wang, Q. Peng et al., "Hybrid neural recommendation with joint deep representation learning of ratings and reviews," *Neurocomputing*, vol. 374, pp. 77–85, 2020.

[5] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 165–174, ACM, New York, NY, USA, 2019.

[6] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "LightGCN: simplifying and powering graph convolution network for recommendation," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 639–648, ACM, New York, NY, USA, 2020.

[7] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2021.

[8] A. Aji, Y. Wang, E. Agichtein, and E. Gabrilovich, "Using the past to score the present," in *Proceedings of the 19th ACM International Conference on Information and Knowledge Management - CIKM '10*, p. 629, 2010.

[9] H. Takemura and K. Tajima, "Tweet classification based on their lifetime duration," in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management - CIKM '12*, p. 2367, 2012.

[10] M. Zhang, S. Wu, X. Yu, Q. Liu, and L. Wang, "Dynamic graph neural networks for sequential recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, p. 1, 2022.

[11] J. C. Robinson, "The takens time-delay embedding theorem," in *Proccedings of the Dimensions, Embeddings, and Attractors*, pp. 145–159, Cambridge University Press, Cambridge, 2010.

[12] H. Yu, L. T. Yang, Q. Zhang, D. Armstrong, and M. J. Deen, "Convolutional neural networks for medical image analysis: state-of-the-art, comparisons, improvement and perspectives," *Neurocomputing*, vol. 444, 2021.

[13] S. Lawrence, C. L. Giles, and A. D. Ah Chung Tsoi, "Back, Face recognition: a convolutional neural-network approach," *IEEE Transactions on Neural Networks*, vol. 8, 1997.

[14] D. Wang, Y. Yih, and M. Ventresca, "Improving neighbor-based collaborative filtering by using a hybrid similarity measurement," *Expert Systems with Applications*, vol. 160, p. 160, 2020.

[15] Z. Hu, Y. Zhang, Y. Xing, Y. Zhao, D. Cao, and C. Lv, "Toward human-centered automated driving: a novel spatiotemporal vision transformer-enabled head tracker," *IEEE Vehicular Technology Magazine*, 2022.

[16] Z. Hu, C. Lv, P. Hang, C. Huang, and Y. Xing, "Data-driven estimation of driver attention using calibration-free eye gaze and scene features," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 2, pp. 1800–1808, 2022.

[17] H.-T. Cheng, L. Koc, J. Harmsen et al., "Wide & deep learning for recommender systems," in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, 2016.

[18] H. Wang, X. Shi, and D.-Y. Yeung, "Collaborative recurrent autoencoder: recommend while learning to fill in the blanks," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2016.

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[20] K. Cho, B. van Merrienboer, C. Gulcehre et al., "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Association for Computational Linguistics, Stroudsburg, PA, USA, 2014.

[21] K. Sun, T. Qian, T. Chen, Y. Liang, Q. V. H. Nguyen, and H. Yin, "Where to go next: modeling long- and short-term user preferences for point-of-interest recommendation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, pp. 214–221, 2020.

[22] L. Zheng, V. Noroozi, and P. S. Yu, "Joint deep modeling of users and items using reviews for recommendation," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pp. 425–434, ACM, New York, NY, USA, 2017.

[23] H. Wu, Z. Zhang, K. Yue, B. Zhang, J. He, and L. Sun, "Dual-regularized matrix factorization with deep neural networks for recommender systems," *Knowledge-Based Systems*, vol. 145, pp. 46–58, 2018.

[24] C. Gao, X. Wang, X. He, and Y. Li, "Graph neural networks for recommender system," in *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pp. 1623–1625, ACM, New York, NY, USA, 2022.

[25] S. Zhang, H. Tong, J. Xu, and R. Maciejewski, "Graph convolutional networks: a comprehensive review," *Computational Social Networks*, vol. 6, no. 1, p. 11, 2019.

[26] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, I. Guyon, U. v Luxburg, S. Bengio et al., Eds., pp. 1025–1035, Curran Associates Inc., Red Hook, NY, USA, 2017.

[27] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *Proceedings of the International Conference on Learning Representations*, New Orleans, LI, USA, 2019.

[28] J. Li, H. Dani, X. Hu, J. Tang, Y. Chang, and H. Liu, "Attributed network embedding for learning in a dynamic environment," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 387–396, ACM, New York, NY, USA, 2017.

[29] R. Ye, Y. Hou, T. Lei et al., "Dynamic graph construction for improving diversity of recommendation," in *Proceedings of the Fifteenth ACM Conference on Recommender Systems*, pp. 651–655, ACM, New York, NY, USA, 2021.

[30] H. Xu, C. Huang, Y. Xu, L. Xia, H. Xing, and D. Yin, "Global context enhanced social recommendation with hierarchical graph neural networks," in *Proceedings of the 2020 IEEE International Conference on Data Mining (ICDM)*, pp. 701–710, IEEE, Sorrento, Italy, 2020.

[31] W. Song, Z. Xiao, Y. Wang, L. Charlin, M. Zhang, and J. Tang, "Session-based social recommendation via dynamic graph attention networks," in *Proceedings of the Twelfth ACM*

*International Conference on Web Search and Data Mining*, pp. 555–563, ACM, New York, NY, USA, 2019.

[32] Q. Liu, Y. Zeng, R. Mokhosi, and H. Zhang, "STAMP : short-term attention/memory priority model for session-based recommendation," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1831–1839, ACM, New York, NY, USA, 2018.

[33] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a "siamese" time delay neural network," in *Proceedings of the 6th International Conference on Neural Information Processing Systems*, pp. 737–744, San Francisco, CA, USA, 1993.

[34] J. Son and S. B. Kim, "Academic paper recommender system using multilevel simultaneous citation networks," *Decision Support Systems*, vol. 105, pp. 24–33, 2018.

[35] T. Pradhan and S. Pal, "A hybrid personalized scholarly venue recommender system integrating social network analysis and contextual similarity," *Future Generation Computer Systems*, vol. 110, pp. 1139–1166, 2020.

[36] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: bayesian personalized ranking from implicit feedback," in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pp. 452–461, AUAI Press, Arlington, Virginia, USA, 2009.

[37] D. P. Kingma and J. Ba, "Adam: a method for stochastic 7 optimization," in *Proceedings of the International Conference on Learning Representations (ICLR)*, Ithaca, NY, USA, May 2015.

[38] G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th International Conference on World Wide Web*, pp. 285–295, New York, NY, USA, 2001.

[39] H.-J. Xue, X. Dai, J. Zhang, S. Huang, and J. Chen, "Deep matrix factorization models for recommender systems," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pp. 3203–3209, International Joint Conferences on Artificial Intelligence Organization, California, 2017.

[40] X. Wang, H. Jin, A. Zhang, X. He, T. Xu, and T.-S. Chua, "Disentangled graph collaborative filtering," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020.

[41] S. Gao, M. Zhou, Y. Wang, J. Cheng, H. Yachi, and J. Wang, "Dendritic neuron model with effective learning algorithms for classification, approximation, and prediction," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 2, pp. 601–614, 2019.