GAG: Global Attributed Graph Neural Network for Streaming Session-based Recommendation

Ruihong Qiu
The University of Queensland
Brisbane, Australia
r.qiu@uq.edu.au

Hongzhi Yin*
The University of Queensland
Brisbane, Australia
h.yin1@uq.edu.au

Zi Huang
The University of Queensland
Brisbane, Australia
huang@itee.uq.edu.au

Tong Chen
The University of Queensland
Brisbane, Australia
tong.chen@uq.edu.au

ABSTRACT

Streaming session-based recommendation (SSR) is a challenging task that requires the recommender system to do the session-based recommendation (SR) in the streaming scenario. In the real-world applications of e-commerce and social media, a sequence of user-item interactions generated within a certain period are grouped as a session, and these sessions consecutively arrive in the form of streams. Most of the recent SR research has focused on the static setting where the training data is first acquired and then used to train a session-based recommender model. They need several epochs of training over the whole dataset, which is infeasible in the streaming setting. Besides, they can hardly well capture long-term user interests because of the neglect or the simple usage of the user information. Although some streaming recommendation strategies have been proposed recently, they are designed for streams of individual interactions rather than streams of sessions. In this paper, we propose a **G**lobal **A**ttributed **G**raph (GAG) neural network model with a Wasserstein reservoir for the SSR problem. On one hand, when a new session arrives, a session graph with a global attribute is constructed based on the current session and its associate user. Thus, the GAG can take both the global attribute and the current session into consideration to learn more comprehensive representations of the session and the user, yielding a better performance in the recommendation. On the other hand, for the adaptation to the streaming session scenario, a Wasserstein reservoir is proposed to help preserve a representative sketch of the historical data. Extensive experiments on two real-world datasets have been conducted to verify the superiority of the GAG model compared with the state-of-the-art methods.

*Corresponding author and contributing equally with the first author.

SIGIR '20, July 25–30, 2020, Virtual Event, China

ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401109

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

streaming recommendation, session-based recommendation, graph neural networks

ACM Reference Format:
Ruihong Qiu, Hongzhi Yin, Zi Huang, and Tong Chen. 2020. GAG: Global Attributed Graph Neural Network for Streaming Session-based Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397271.3401109

1 INTRODUCTION

In many modern online systems, such as e-commerce and social media platforms, there usually exist a large number of interactions between users and items, such as clicking goods and playing songs. As illustrated in Fig. 1, a sequence of interactions occurring in a certain period can be considered as a session. Session-based recommendation (SR) has been widely studied recently by the academia and the industry [13, 19, 20, 22, 30, 38], which aims to recommend items to users based on sessions. However, most of them focus on the static setting, which is not suitable in real-life situation.

In practice, sessions are dynamically produced as a stream, which leads to urgent requirements of streaming session-based recommendation (SSR). As presented in Fig. 1, a general procedure for SSR is to train the recommendation model with the historical sessions to preserve the users' long-term interests and then conduct the online update with the streaming sessions to adapt to their recent preferences. Most of the current research for the SR focuses on the static scenario, where the recommendation models are trained in a batch way. As users' preferences are changing over time, it is infeasible to apply a static model for new coming sessions. To precisely capture the user preference, the model needs to be online updated with the latest sessions. Due to the memory space limit, it is unpractical to stack up the training data by absorbing every new session. In the meanwhile, the training time will be another concern in the streaming scenario, where the recommendation model is expected to be updated promptly. However, a typical SR model needs to train for a long time to converge, which cannot be guaranteed within a short period.

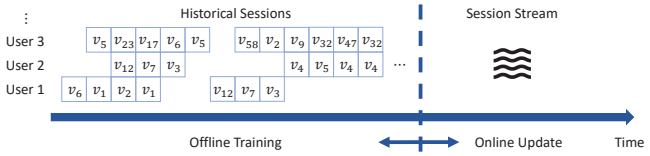


Figure 1: The general framework for SSR consists of two phases: the offline training and the online update. During the offline training, the recommender system is trained in a static style with the whole historical session data. When newly generated sessions arrive, the model is expected to conduct an efficient online update with streaming techniques to preserve the users' long-term interests and adapt to the newest preferences.

Recently, a few methods utilize the reservoir technique for streaming tasks [3, 4, 6, 34, 35]. In these cases, the interaction data is stored with the same probability in the reservoir and then sampled for the online training of the model. However, if streaming sessions are processed in such a manner, the SSR models will suffer from the loss of the session information because the data samples are stored and drawn as discrete interactions. Moreover, the reservoir for traditional streaming tasks is designed to capture the matrix factorization information rather than the session's sequence pattern. Besides, online learning can barely adapt to the session-based recommendation task for newly arrived data as well. For online learning, when a new session comes, the model will update accordingly to capture the recent transition pattern in the latest session. However, these models will easily overfit the new data and fail to maintain users' long-term preferences learned from historical data. Therefore, it is important for the SSR model to effectively exploit the user's information and thus obtain a comprehensive representation for both long- and short-term preferences.

More recently, Guo et al. [9] applied the reservoir technique to the SSR task with a weighted sampling scheme by evaluating how informative each session is. This method cannot be generalized to other models mainly because it needs to generate an informativeness score for every item in a session with pre-computed item feature vectors, which are commonly unavailable in other models. Moreover, this model directly combines the session-based method with a matrix factorization module for recommendation, which can hardly learn the complicated correlations between users and items in the SSR problem.

To address the issues discussed above, we propose a **G**lobal **A**ttributed **G**raph (GAG) neural network model with a Wasserstein reservoir as a solution to SSR. To make the most usage of the user embeddings and maintain long-term preference information for SSR, we firstly convert a user's session sequence into a session graph having the user embedding as a global attribute associated with the embeddings of interacted items. Based on the global attributed session graph, the GAG model performs graph convolution to learn an updated global attribute, which is passed to the ranking module to output a recommendation list. In the GAG model, the global attribute is applied to effectively assist the joint representation learning of both the entire session and the items within the session. To develop a general reservoir for the SSR problem, we propose the Wasserstein reservoir, which stores and samples session data according to the Wasserstein distance between the generated recommendation lists and the user's real interactions. During the sampling procedure, the Wasserstein reservoir samples the sessions whose recommendation results have a higher Wasserstein distance. Intuitively, the model makes worse predictions in these sessions with a higher Wasserstein distance, which is more informative to refine the model during the online update.

The main contributions of this paper are summarized as follows:

- We propose the GAG model to effectively memorize and incorporate users' long-term preferences into the embedding vectors for SSR by treating the user embedding as a global attribute for the session graphs to allow for more expressiveness when learning representations.
- A Wasserstein reservoir is designed to actively select the most informative training cases for updating the model in streaming settings. Moreover, our Wasserstein reservoir is an effective yet generic online learning approach that can be easily applied to other steaming session data.
- Extensive experimental results on two real-world datasets demonstrate that the proposed GAG model and the Wasserstein reservoir achieve the state-of-the-art performance.

2 RELATED WORK

2.1 Session-based Recommendation

Sequential recommendation is mainly based on the Markov chain model [29, 36, 44], which learns the dependency of items in a sequence data. Using probabilistic decision-tree models, Zimdars et al. [44] proposed to encode the state of the transition pattern of items. Shani et al. [29] made use of a Markov Decision Process (MDP) to compute item transition probabilities.

Deep learning models are popular with the boom of recurrent neural networks [13, 15, 19–21, 32]. Hidasi et al. [13] proposed the GRU4REC, which applies the GRU [5] to treat the data as time series. Some recent approaches use the attention mechanism to avoid the time order. NARM [19] stacks GRU as the encoder to extract information and then a self-attention layer to sum up as the session embedding. To further alleviate the bias by time series, STAMP [20] replaces the recurrent encoder with the attention layer. Recently, GNN has been widely used in recommendation [31, 37, 42]. Some methods utilize GNN to encode the session information to prevent the misguidance of the session order [22, 23, 38, 39]. SSRM [9] considers a specific user's historical sessions and applies the attention mechanism to combine them.

2.2 Streaming Recommendation

Online learning focuses on updating the old model with the new data [12, 14] to capture the most recent interest of the user [40, 41]. For instance, He et al. [12] proposed an element-wise alternative to the least squares technique to address the missing data. Jugovac et al. [14] applied a replay-based evaluation protocol to update the model with the new arrival events and articles in the news recommendation. Although the models above capture the user's recent interest by updating the model with new interactions, they fail to remember historical interactions.

Random sampling is a technique to address the history-ignoring problem by introducing a reservoir to store the user's long-term interactions [3, 4, 6, 9, 34, 35, 43]. For example, Diaz-Aviles et al. [6]

applied sampling strategies based on active learning principles on the matrix factorization method to update the model. More recently, Guo et al. [9] used the same reservoir technique to process the streaming sessions.

2.3 Graph Neural Networks

Originally, GNN is applied basically on directed graphs in a simple situation [8, 28]. In recent years, many GNN methods [10, 17] work very similar to the message passing network [7] to perform an aggregation over the neighborhood of nodes to compute the node embeddings. In [1, 11, 27], the global attribute is introduced into the GNN layer to maintain a graph level feature in physical systems.

3 METHOD

3.1 Task Definition

In the SSR problem, there is an item set $\mathcal{V} = \{v_1, v_2, v_3, \ldots, v_m\}$, where all items are unique and m denotes the number of items. Usually, an embedding layer is applied to represent all items, $\mathbf{x_i} = \text{Embed}_v(v_i), i \leq m$, where Embed_v is a mapping function that transforms an item into a continuous and dense representation $\mathbf{x_i} \in \mathbb{R}^d$. There is also a user set $\mathcal{U} = \{u_1, u_2, u_3, \ldots, u_n\}$, where all users are unique and n denotes the number of users. Similarly, another embedding layer performs mapping to all user ID, $\mathbf{u_j} = \text{Embed}_u(u_j), j \leq n$, where Embed_u is another mapping function from a user to $\mathbf{p_j} \in \mathbb{R}^d$. A session sequence at a time step t from a user u is defined as a list $S_{u,t} = [v_1, v_2, \ldots, v_l], v_* \in \mathcal{V}$. l is the length of the session S, which may contain duplicated items, $v_a = v_b, a, b < l$. In the setting of the SSR, at time step t, the recommender system needs to recommend an item v_{t+1} based on $\{S_{u,0}, S_{u,1}, \ldots, S_{u,t}\}$, which are all sessions of a user from the history to the current. The item v_{t+1} should match the user's preference the most. In the meantime, sessions arrive at a high speed, which means that the computation resource is limited to calculation. As a result, an algorithm should have an efficient way to process the history sessions as well as the current session. Usually, we only recommend the top-K ranked items to users.

3.2 Overview

In this paper, we propose a novel Global Attributed Graph (GAG) neural network model to address the SSR problem mainly by transforming a user's information into the global attribute and incorporating it in the session graph. The architecture of the GAG model is demonstrated in Fig. 2. There are two key components: GAG model for generating recommendation and Wasserstein reservoir for the streaming data learning.

3.3 Global Attributed Session Graph

As shown in Fig. 2, at the first stage, the session sequence is converted into a session graph with a global attribute for the purpose to process each session via GNN. Similar to [38] and [22], because of the natural order of the session sequence, we convert it into a weighted directed graph. In addition, we incorporate the user's general information as the global attribute u into the session graph, $G_s = (\mathbf{u}, V_s, E_s), G_s \in \mathcal{G}$, where $\mathcal{G}$ is the set of all session graphs. In the session graph G_s, the node set V_s represents the nodes in

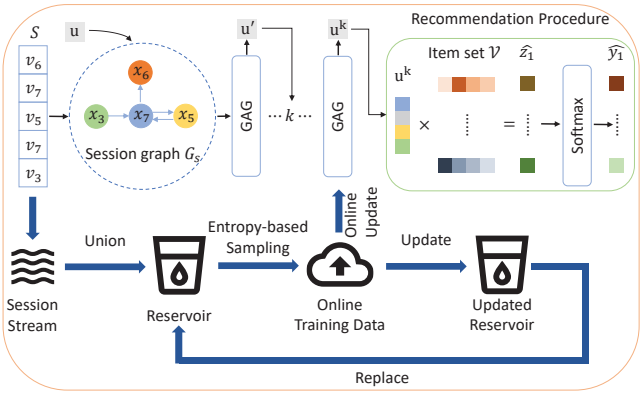


Figure 2: The pipeline of the GAG model for the SSR problem. In the upper half, for a specific session S from a user u, the GAG model first converts it into a global attributed session graph G_s. The GAG layer takes G_s as input and computes the graph convolution based on node features, edge weights and the global attribute. The output of the k-layer GAG model is the updated global attribute $\mathbf{u}^k$. To make a personalized recommendation, $\mathbf{u}^k$ is applied to compare with the whole item set to generate the recommendation list. In the bottom half, it shows the procedure of the Wasserstein reservoir dealing with streaming sessions. New streaming sessions and the current reservoir are united together and sampled the online training data according to their respective Wasserstein distance. The online training data is fed to help with the update of the current GAG model. After the online training of the model, the reservoir is updated with the session stream and itself. The updated reservoir is for the following new arrival sessions.

the session graph, which are items $v_{s,n}$ from S. For every node $\mathbf{v}$, the input feature is the initial embedding vector $\mathbf{x}$. The edge set E_s represents all directed edges $(w_{s,(n-1)n}, v_{s,n-1}, v_{s,n})$, where $w_{s,(n-1)n}$ is the weight of the edge and $v_{s,n}$ is the click of the item after $v_{s,n-1}$ in S. The weight of the edge is defined as the frequency of the occurrence of the edge within the session.

3.4 Global Attributed Graph Neural Network

With the construction of the global attributed session graph G_s, we propose the GAG model to perform graph convolution on G_s with the node features, edge features and the global attribute. When the GAG model is fed with the session graph as the input, the computation proceeds from the edge, the node to the global attribute.

First, the per-edge update is calculated among all edges to compute the output features from sender nodes $\mathbf{v_{s_k}}$ to receiver nodes $\mathbf{v_{r_k}}$ with additional features of the edge itself $\mathbf{e_k}$ and the global attribute $\mathbf{u}$. In our design of GAG, the edge feature, i.e., the weight of the edge, will not be updated because the edge feature is not in the dense vector form. This setting means that the output of the edge update function $\mathbf{e_k}'$, will only be used to update other node features and global features. The output new session graph G_s' has the same edge set E_s as the input session graph G_s. Because the session graph is built in the directed situation, we compute the propagation in both directions to represent the different meanings for a node as a sender and a receiver in an edge. Therefore, the ϕ^e

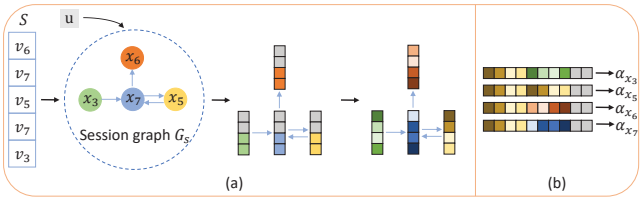


Figure 3: The usage of the global attribute in GAG. (a) In the input stage, the global attribute is concatenated with the node feature for every node, which gives out a node feature concatenated with the global attribute. (b) In the global attribute update procedure, the attention weight α_i is calculated based on the concatenation of the features of the last node $\mathbf{v_l}$, the individual node $\mathbf{v_i}$ and the global attribute u itself.

function is designed as:

$$\begin{aligned}\mathbf{e_{k,in}}' &= \phi^e_{\text{in}}\left(\mathbf{e_k}, \mathbf{v_{r_k}}, \mathbf{v_{s_k}}, \mathbf{u}\right) \\ &= w_k \cdot \text{MLP}(\mathbf{v_{s_k}}||\mathbf{u}),\end{aligned} \tag{1}$$

$$\begin{aligned}\mathbf{e_{k,out}}' &= \phi^e_{\text{out}}\left(\mathbf{e_k}, \mathbf{v_{r_k}}, \mathbf{v_{s_k}}, \mathbf{u}\right) \\ &= w_k \cdot \text{MLP}(\mathbf{v_{s_k}}||\mathbf{u}),\end{aligned} \tag{2}$$

where w_k is the scalar form of $\mathbf{e_k}$, MLP stands for the multi-layer perceptron to encode the features provided by a concatenation of the sender and receiver node and $||$ means the concatenation between two vectors along the unit dimension. MLPs in both equations are not shared because they perform different operations to the node features. In Fig. 3 (a), the procedure of exploiting the global attribute in the node update function is demonstrated in detail.

After updating the new per-edge features, per-node features are updated based on the per-edge features when the node is the sender and the receiver. The new per-node feature consists of the normalized summation of the in-coming and out-going neighborhoods. The aggregation procedures are as:

$$\mathbf{v_{i,in}}' = \sum_{j \in \{\mathbf{v_{s_j}} = \mathbf{v_{r_i}}\}} \frac{\mathbf{e_{j,out}}'}{\sqrt{N_{in}(i)N_{out}(j)}}, \tag{3}$$

$$\mathbf{v_{i,out}}' = \sum_{j \in \{\mathbf{v_{r_j}} = \mathbf{v_{s_i}}\}} \frac{\mathbf{e_{j,in}}'}{\sqrt{N_{out}(i)N_{in}(j)}}, \tag{4}$$

where $N_{in}(*)$ and $N_{out}(*)$ represent the in-coming and the out-going degree of a node.

The final result of the neighborhood aggregation is a linear transformation of the in-coming and the out-going feature:

$$\mathbf{v_i}' = \text{MLP}(\mathbf{v_{i,in}}'||\mathbf{v_{i,out}}'). \tag{5}$$

The updated node feature $\mathbf{v_i}'$ actually includes the information from the node feature of itself and the neighborhood, the edge weight and the global attribute.

At the last step of the GAG layer forward computations, the global attribute is updated based on all the features of nodes, edges and the global attribute itself in the graph. It is worth noting that the purpose of the session-based recommendation is to generate a representation of a session to recommend items. Therefore, the final global attribute is exactly the representation we desire to represent the whole graph. Similar to the previous methods to separate the representation of the preference in long-term and short-term parts inside a session [19, 20, 38], a self-attention on the last input item v_l, of the session is applied to aggregate all item features of the session to be the session-level feature. The computation of updating $\mathbf{u}$ to $\mathbf{u_{sg}}$ is defined as:

$$\begin{aligned}\mathbf{u}' &= \phi^u\left(V', \mathbf{u}\right) \\ &= \text{Self-Att}(v_l', v_i', \mathbf{u}) + \mathbf{u},\end{aligned} \tag{6}$$

where $v_i \in V', i = 1, 2, 3, \ldots, l$ represent all items in the session after being updated to the new features. In the setting of the session graph, items are converted into nodes and the Self-Att can be divided into two steps:

$$\alpha_i = \text{MLP}(\mathbf{v_l}'||\mathbf{v_i}'||\mathbf{u}), \tag{7}$$

$$\mathbf{u_{sg}} = \sum_{i=1}^{n} \alpha_i \mathbf{v_i}', \tag{8}$$

where an MLP is utilized to learn the weights that aggregate the node features, the last node feature and the global attribute. In Fig. 3 (b), the detail of the computation of attention weights is presented.

Besides, because the user's profile is applicable in the SSR setting, the incorporation of the user embedding can provide the extra user information. Therefore, the final formula for how to compute the output of the global attribute with user information is defined as:

$$\mathbf{u}' = \mathbf{u_{sg}} + \mathbf{u}. \tag{9}$$

The residual addition can help to alleviate the burden of directly learning the updated global attribute.

3.5 Recommendation

The last stage of the GAG to perform the recommendation is the generation of candidate items based on the representation of the input session and the user's profile. We compute a score for every item and form a score vector $\hat{\mathbf{z}} \in \mathbb{R}^n$, where n is the size of the item set. Specifically, the score vector $\hat{\mathbf{z}}$ is calculated as:

$$\hat{\mathbf{z}} = \mathbf{u}'^{\top}\mathbf{X}, \tag{10}$$

where $\mathbf{X}$ is embeddings of all items in the item set.

The probabilistic form of the prediction $\hat{\mathbf{y}}$ is defined as:

$$\hat{\mathbf{y}} = \text{Softmax}(\hat{\mathbf{z}}). \tag{11}$$

3.6 Wasserstein Reservoir for Streaming Model

In this section, we extend our offline model to the streaming setting. Our purpose is to update our model with the new arrival session data while keeping the knowledge learned from the historical sessions. Traditionally, online learning methods update the model only with the new data, which will always lead to forgetting the past [25]. To prevent the model from losing the awareness of historical data, we leverage the reservoir to maintain a long-term memory of the historical data [4, 9, 34, 35]. The reservoir technique is widely used in the streaming database management systems.

The purpose of applying a reservoir is to maintain a representative sketch of all the historical data. Therefore, we conduct a random sampling [33] to select the data stored in the reservoir. Let C denote the reservoir, which contains $|C|$ sessions. Let t be

the time order of the arrival session instance. When $t > |C|$, the reservoir will store this t-th session with the probability:

$$p_{\text{store}} = \frac{|C|}{t}, \tag{12}$$

and replaces a uniformly random session that is already in C. This method of generating the reservoir is actually sampling randomly from the current dataset, and it can successfully maintain the model's long-term memory [6].

Although the reservoir can be updated in the way introduced above, the probability for a new arrival session to be included tends to be smaller over time, and the reservoir will have a chance of overlooking the recent data. However, the recent data is crucial for predicting the user's varying preference. Besides, new users and new items are exposed to the system continually. In consequence, when the new session data C^{new} arrives, we update the pre-trained model with C^{new} and the reservoir C.

The reservoir sampling procedure above enables the model to continually update according to the new and old data. However, it can narrowly achieve a good performance in reality. The reason is that most training sessions in C are already learned well by M, which results in C^{rand} mainly containing helpless training samples. Actually, if the current model makes a worse prediction on a session, it is more worthwhile to update the model with this session because it either contains the latest preference of a user or there is some item transition patterns that the current model cannot learn well. Such a session is called an *informative* session to the model and this session is more significant to the model update. In our work, the *informativeness* of a session is defined as the distribution distance d between its predicted recommendation result $\hat{\mathbf{y}}$ and the real interaction $\mathbf{y}$. $\hat{\mathbf{y}}$ is a distribution given by the model M and $\mathbf{y}$ is a one-hot vector. Intuitively, the greater of this distance, M predicts the worse over the session. There are different distance metrics of two distributions:

- The *Wasserstein* distance (EMD distance) [26]:

$$d_W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|] \tag{13}$$

- The *Kullback-Leibler* (KL) divergence [18]:

$$d_{KL}(\mathbb{P}_r \| \mathbb{P}_g) = \sum_{i=1}^{n} P_r(x) \log \frac{P_r(x)}{P_g(x)} \tag{14}$$

- The *Total Variation* (TV) distance:

$$d_{TV}(\mathbb{P}_r, \mathbb{P}_g) = \sup_{A \in \Sigma} \left| \mathbb{P}_r(A) - \mathbb{P}_g(A) \right| \tag{15}$$

Within the recommendation task, the real distribution P_r is always a one-hot vector $\mathbf{y}$. Under this situation, the KL divergence between $\mathbf{y}$ and $\hat{\mathbf{y}}$, $d_{KL}(\mathbf{y}\|\hat{\mathbf{y}})$ is:

$$d_{KL}(\mathbf{y}\|\hat{\mathbf{y}}) = -\log P_g(v_i), \tag{16}$$

where $P_g(v_i)$ is the predicted probability over the ground truth item v_i given by the model M. Therefore, the KL divergence fails to take the whole distribution into consideration.

As for the TV distance, $d_{TV}(\mathbf{y}, \hat{\mathbf{y}})$ is:

$$d_{TV}(\mathbf{y}, \hat{\mathbf{y}}) = \max_{j \neq i}(1 - P_g(v_i), P_g(v_j)), \tag{17}$$

Algorithm 1 Online Update with Wasserstein Reservoir

Input: the current time step t, the current model M, the current reservoir C and new sessions C^{new};
Output: the updated model M' and the updated reservoir C';

```
initialize a blank update dataset S;
if the last epoch finished then
    for each session s_i in C^new do
        if a new item or a new user appears then
            append s_i to S;
        end if
    end for
    for each session s_i in C ∪ C^new − S do
        compute the Wasserstein distance d_i of s_i;
    end for
    compute the sample probability p(s_i);
    sample the the rest of S according to Eq. (18);
end if
update the current model M with S to M';
for each session s_i in C^new do
    update the reservoir with s_i according to Eq (12) to C';
    update t;
end for
```

where $P_g(v_j)$ is the predicted probability over the item v_j given by the model M other than the ground truth item v_i. The TV distance either captures the difference over the real interacted item or other unrelated items.

Obviously, both KL divergence and TV distance have drawbacks of focusing on a certain elementary event while neglecting the whole distribution. The only metric that can preserve the difference at each prediction score is the Wasserstein distance.

Therefore, we propose a Wasserstein reservoir construction strategy that samples the session whose output probability distribution over the recommendation item has a higher Wasserstein distance to the user's real interaction with a higher probability. Intuitively, when the output probability of a session has higher Wasserstein distance, the output will contain more information compared with those with lower Wasserstein distance. Therefore, we sample sessions according to the Wasserstein distance of their output probabilities. For session s_i with corresponding Wasserstein distance d_i for its output, the sampling probability is calculated as follows:

$$p_{\text{sample}}(s_i) = \frac{d_i}{\sum_{s_j \in C \cup C^{new} - S} d_j}, \tag{18}$$

where S is the ongoing updated dataset. The detailed construction of the Wasserstein reservoir is shown in Algorithm 1. During the sampling procedure, because there are always new items and new users in the streaming data, their corresponding embedding vectors are not trained before they show up. To prevent the model from neglecting the new sessions, these sessions are directly included.

3.7 Training

Since the recommendation task is considered as a classification problem over the whole item set, we can apply a multi-class cross-entropy loss between the predicted recommendation distribution $\hat{\mathbf{y}}$

and the real interaction $\mathbf{y}$:

$$L = -\sum_{i=1}^{l} \mathbf{y}_i \log(\hat{\mathbf{y}}_i), \tag{19}$$

where l is the number of training sessions in a mini-batch.

4 EXPERIMENT SETUP

4.1 Dataset

LastFM [1] is a real-world music recommendation dataset, which is released by Celma Herrada [2]. In this work, we mainly focus on music artist recommendation. As shown by Guo et al. [9], we also consider doing recommendations on artists and choose the 10,000 most popular ones. Based on the time order, we group transactions in 8 hours from the same user as a session. Following [19], sessions that contain more than 20 transactions or less than 2 will be filtered out. In total, there are 298,919 sessions after the pre-processing.

Gowalla [2] is a point-of-interest real-world dataset collected from a social network for users' check-in. The same as Guo et al. [9], the 30,000 most popular places are used for experiments and check-ins within 1 day are defined as a session. Again, sessions that contain more than 20 transactions or less than 2 will be filtered out. Finally, we have 198,680 sessions during experiments.

4.2 Metrics

Following Chang et al. [3] and Guo et al. [9], to simulate the streaming situation of the data arriving situation, the dataset is split into two proportions (60% and 40%) by the chronological order of all data. The first part is defined as the training set ($\mathcal{D}^{train}$) while the second part is the candidate set ($\mathcal{D}^{candidate}$). Specifically, $\mathcal{D}^{train}$ is used for training the GAG model as offline data. As for $\mathcal{D}^{candidate}$, it is designed to simulate the online streaming session data. Especially for $\mathcal{D}^{candidate}$, it is further divided into five same-sized test set by time order, $\mathcal{D}^{test,1}, \ldots, \mathcal{D}^{test,5}$. Test sets are provided in the time order to the model during test time. And after testing on the current test set, the model will be updated according to it and the updated model accounts for the test for the next test set. Such an online update is designed for the streaming occasion.

To evaluate the performance of our model, according to the nature of the user's picking of the first few recommended items, the top-20 recommendation is applied here and we mainly compare different models based on the **Recall@**K and **MRR@**K.

4.3 Baselines

In our experiments, we will mainly compare our GAG model to the following representative baseline methods:

- **POP** always chooses the most popular items of all users to recommend to other users. It is a simple yet strong baseline.
- **S-POP** recommends the most popular items that appear in the current session instead of the whole item set.
- **BPR-MF** is a method that mainly makes use of a pairwise ranking loss [24]. Also, the Matrix Factorization is modified to suit the session-based recommendation as in [19].
- **GRU4REC** utilizes GRU layers to learn the session embedding in the anonymous setting [13].
- **NARM** adds an attention layer to item level across the session to encode the session information within an anonymous setting [19].
- **FGNN** makes use of GNN to learn an embedding of an anonymous session to make recommendation [22]. This method does not consider the user information.
- **SSRM** is a state-of-the-art method for the SSR problem, which applies a reservoir to sample the history sessions to help the current session embedding learning [9].

4.4 Training Detail

In the implementation of the model, we set all MLPs with 1 layer and the embedding size is 200 for the fairness of comparison. We use Adam [16] with a learning rate of 0.003 and set the batch size as 100 to train the GAG model. The size of the reservoir is set to $|D|/100$ and the window size is set to $|C^{new}|/2$ on each $C \cup C^{new}$.

5 EXPERIMENT RESULTS

In this section, we will describe our experiments on two real-world datasets and demonstrate the efficacy of our proposed model GAG. Specifically, four research questions will be addressed:

- **RQ1** How does our proposed GAG model perform compared with current state-of-the-art methods? (Section 5.1)
- **RQ2** How does the global attribute help to solve the SSR problem? (Section 5.2)
- **RQ3** How is the performance of the Wasserstein reservoir? (Section 5.3)
- **RQ4** How is the parameter sensitivity of the GAG model? (Section 5.4)

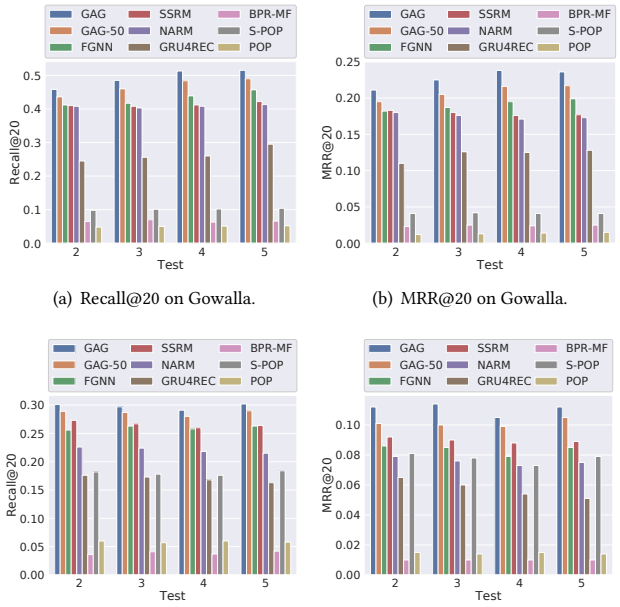
5.1 Comparisons with Baseline Methods

To evaluate the overall performance of the GAG, we compare the GAG model with the baseline methods mentioned in Section 4.3 by the Recall@20 and MRR@20 scores on Gowalla and LastFM datasets. The overall results are demonstrated in Fig. 4. We also use the top-5 and top-10 recommendation results for a more in-depth comparison. For fairness, we have the GAG-50 model with the embedding size as 50 in accordance with the baseline methods. For the state-of-the-art performance, we have the GAG model with the embedding size as 200.

5.1.1 ***General Comparison.*** The overall performance is shown in Fig. 4. Clearly, the proposed GAG-50 model outperforms all the baseline methods in all situations. With the embedding size increasing to 200, the GAG model achieves the state-of-the-art results. Both of them show the superiority of the GAG model.

The performance is much worse for conventional methods, such as POP and S-POP, both of which recommend the most popular items to users. POP recommends the most popular ones from the whole item set while S-POP chooses the most popular item in the current session. POP fixes the recommendation list, which fails to detect the different patterns of users' behaviors in different sessions. However, S-POP is still a strong session-based baseline method because it can capture the item's re-occurrence patterns of the

[1] http://mtg.upf.edu/static/datasets/last.fm/lastfm-dataset-1K.tar.gz
[2] https://snap.stanford.edu/data/loc-gowalla.html



(a) Recall@20 on Gowalla. (b) MRR@20 on Gowalla.

(c) Recall@20 on LastFM. (d) MRR@20 on LastFM.

Figure 4: Results of streaming session-based recommendation performance.

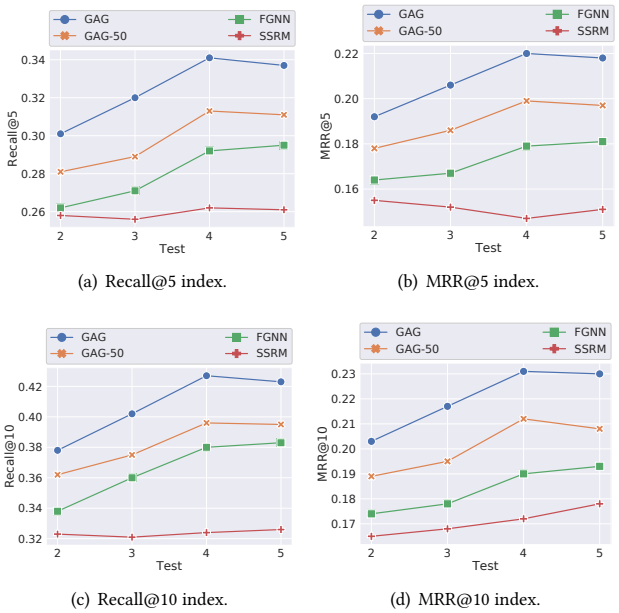
session. Besides, for the shallow method BPR-MF, which performs a matrix factorization of the whole user-item interaction matrix, it has higher performance compared with POP because it can perform the personalized recommendation. However, BPR-MF still fails to outperform S-POP in the SSR problem because S-POP can further extract the session-specific information.

For deep learning models, GRU4REC is a method that utilizes GRU to process the session as a sequence and output a session embedding to make a recommendation. It outperforms traditional methods in most situations, which is proof of the superiority of the deep learning-based approaches. Besides, methods utilizing the attention mechanism, e.g., NARM and SSRM, obtain a great improvement compared with GRU4REC, which shows the capability of the attention mechanism to learn the inter-dependency of items. Especially, SSRM is specifically for the SSR problem and it is the strongest baseline in the experiment.

Recently, graph neural networks have been demonstrated to have a strong ability to model structured data. For example, FGNN is a state-of-the-art method for SR. In this experiment, we can see that FGNN achieves a comparable performance with SSRM. Compared with GAG-50, FGNN has a worse performance because it is only designed for SR and cannot model the user information.

5.1.2 ***In-depth Comparison.*** We further evaluate GAG model by the top-5 and top-10 recommendation results on the Gowalla dataset. Specifically, we use the Recall@K and the MRR@K (K = 5, 10) scores to demonstrate the result in Fig. 5.

According to the results, GAG and GAG-50 still have superiority in the higher standard recommendation. Compared with GNN-based methods, SSRM has a greater drop in both top-5 and top-10 performance, implying that the graph structure and GNN are more suitable for the session representation and the generalization ability.



(a) Recall@5 index. (b) MRR@5 index.

(c) Recall@10 index. (d) MRR@10 index.

Figure 5: In-depth results of streaming session-based recommendation performance on Gowalla.

In contrast, the attention mechanism fails to distinguish the item transition pattern in sessions.

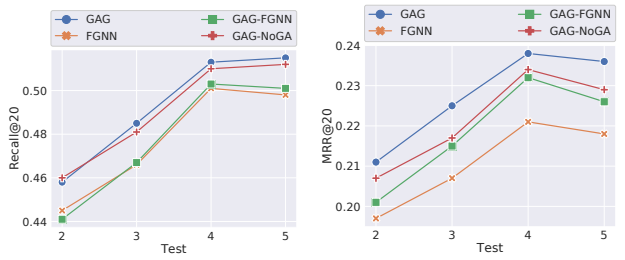
5.2 Effect of Global Attribute

In our GAG model, we utilize the global attribute in both the node embedding update and the global attribute update itself. In this experiment, we conduct the ablation study and make different substitutions of the global attribute to evaluate its efficacy. We use the Recall@20 and MRR@20 on Gowalla and LastFM datasets to evaluate the performance.

5.2.1 ***Ablation Study.*** In this experiment, we compare the GAG model with the following variants:

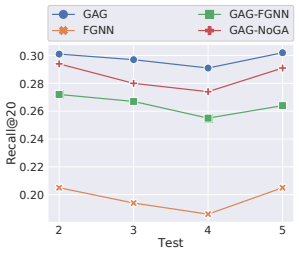
- **FGNN**: FGNN uses the GNN layer that does not take the user information in both the node update layer and the readout function (readout function in a normal GNN model represents the graph level output function). It serves as the basic baseline method.
- **GAG-FGNN**: We substitute the node update function with FGNN's node update layer and maintain the global attribute update function in GAG to evaluate the integration of the user's information in the global attribute update.
- **GAG-NoGA**: In this variant, we keep the global attribute in the node update procedure while removing it in the global attribute update function.

The results are presented in Fig. 6. Each module using the global attribute has a contribution to the recommendation performance. In general, FGNN is the worst because it neglects the global attribute. GAG-FGNN and GAG-NoGA both make improvements by introducing the global attribute. Specifically, GAG-FGNN uses the global attribute in the global attribute update function while GAG-NoGA incorporates the global attribute in the node update function. Comparing these two variants, the results prove that the

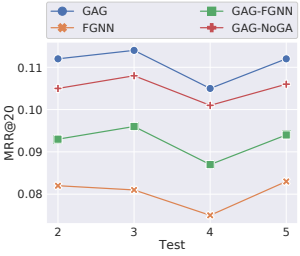


(a) Recall@20 on Gowalla.

(b) MRR@20 on Gowalla.

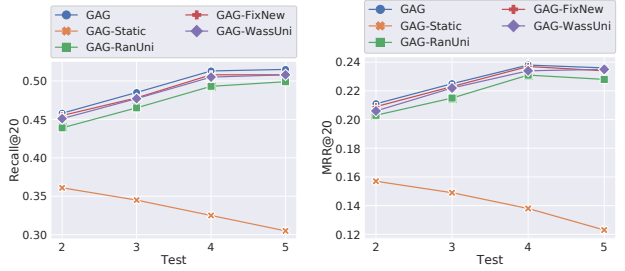


(c) Recall@20 on LastFM.



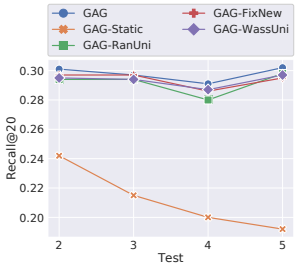
(d) MRR@20 on LastFM.

Figure 6: Results of the ablation study of the global attribute.

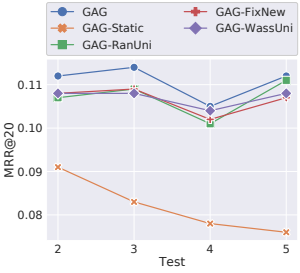


(a) Recall@20 on Gowalla.

(b) MRR@20 on Gowalla.



(c) Recall@20 on LastFM.



(d) MRR@20 on LastFM.

Figure 7: Results of the different reservoir sampling strategies.

global attribute applied to the node update procedure has a greater impact on the recommendation performance than in its self-update.

5.3 Effect of Wasserstein Reservoir

In this section, we conduct experiments to prove the efficacy of the Wasserstein reservoir. The reservoir consists of two major designs: (1) The sampling procedure is based on the Wasserstein distance between the session's predictive distribution and the real interaction; (2) Sessions containing new items or new users will be added to the training sample directly. We substitute the Wasserstein reservoir with other reservoirs to evaluate how the design of the reservoir affects the performance of the GAG model in the SSR problem.

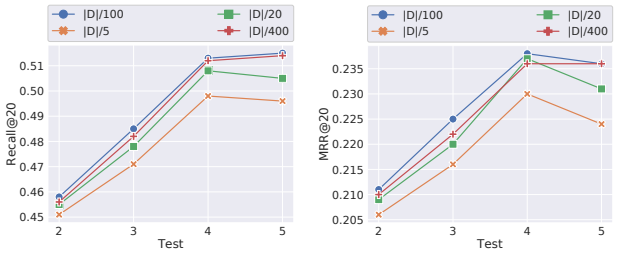
5.3.1 ***Ablation Study.*** In this experiment, an ablation study is conducted to prove the efficacy of both designs in our Wasserstein reservoir. The variants are listed out as follows:

- **GAG-Static**: For this method, we simply eliminate the online training of the model.
- **GAG-RanUni**: This variant only performs random sampling on the union set of the current reservoir and new arrival sessions. It is the most common design of a reservoir.
- **GAG-FixNew**: This variant directly adds a new session containing new items or new users to the training data. For the rest, it still performs a random sampling.
- **GAG-WassUni**: This method samples the training data from the union set of the current reservoir and the new sessions according to their Wasserstein distance.

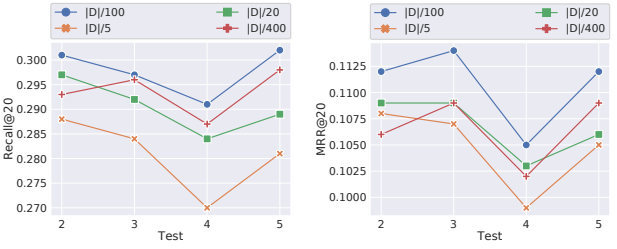
According to the result in Fig. 7, our proposed Wasserstein reservoir achieves the best performance in all situations. For the static recommendation version, GAG-Static, its performance decreases along with the time because there is a shift of the users' preference and the streaming sessions contain new items and new users. In most cases, the random sampling version variant, GAG-RanUni, performs worse than other methods that use a specialized reservoir sampling strategy. The conclusion can be drawn from these two figures that incorporating the sessions containing new items and new users helps with the online update of the model. Comparing GAG-FixNew with GAG-RanUni, the performance of GAG-FixNew is better on Gowalla while it has a decrease in the long-term prediction of $\mathcal{D}^{test,4}$ and $\mathcal{D}^{test,5}$ on LastFM. The model is distracted because the new items and new users in these two parts of the dataset are not representative. Comparing the complete GAG model with GAG-WassUni, it can be seen that the incorporation of new users and items can help with online training. Furthermore, the efficacy of Wasserstein distance is demonstrated. With the Wasserstein distance-based sampling, GAG-WassUni outperforms GAG-RanUni in most cases. Similarly, GAG also has higher scores than GAG-FixNew, which randomly samples the training data.

5.3.2 ***Reservoir Efficiency.*** There are two important parameters in the design of the Wasserstein reservoir: the reservoir size and the window size. On one hand, the reservoir size indicates the volume of the reservoir, which determines the storage requirement of the online update for the recommender system. On the other hand, the window size restricts how many data instances will be sampled for the online training, which represents the work load of the online update for the recommender system.

The default reservoir size is set to $|D|/100$. For comparison, we change the reservoir size to $\{|D|/5, |D|/20, |D|/400\}$ to evaluate the effect of the reservoir size. Results of different reservoir sizes are presented in Fig. 8. When the reservoir size is set to $|D|/100$, our GAG model achieves the best performance. When the reservoir size increases, the probability of the new sessions stored in the reservoir decreases, which makes the model concentrate more on the historical data. However, for the streaming data, the recent ones are more

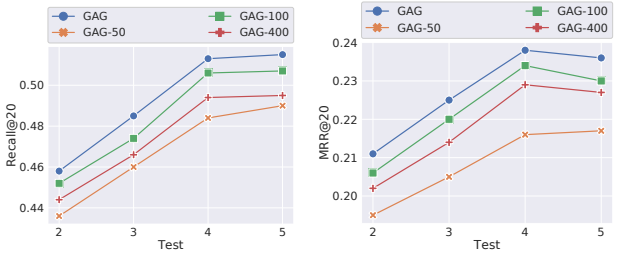


(a) Recall@20 on Gowalla. (b) MRR@20 on Gowalla.

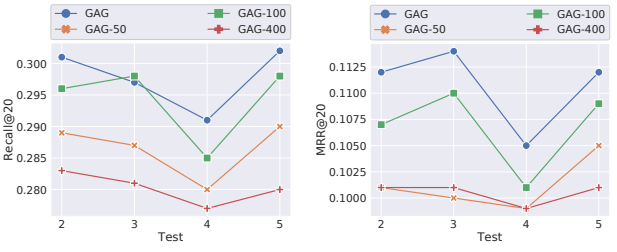


(c) Recall@20 on LastFM. (d) MRR@20 on LastFM.

Figure 8: Results of different reservoir size.



(a) Recall@20 on Gowalla. (b) MRR@20 on Gowalla.

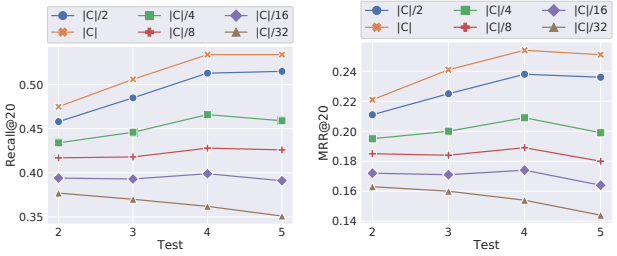


(c) Recall@20 on LastFM. (d) MRR@20 on LastFM.

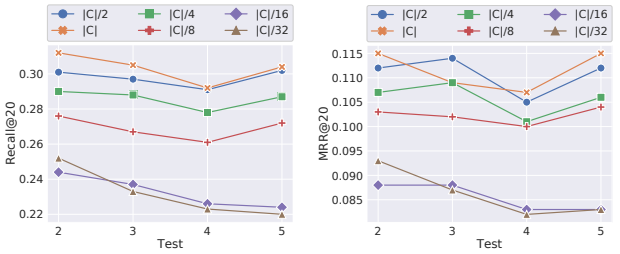
Figure 9: Results of different embedding sizes.

representative of the users' recent preference. When the reservoir size decreases, the streaming performance drops on a smaller scale, which indicates that new sessions are more important for the recommendation performance. For the state-of-the-art method SSRM, it achieves its highest performance with the reservoir size set to $|D|/20$, which is 5 times larger than our GAG model. Apparently, our design has a higher efficiency in reservoir storage.

To evaluate the effect of the window size, we substitute the default window size, $|C|/2$, with $\{|C|, |C|/4, |C|/8, |C|/16, |C|/32\}$ to evaluate the effect of the window size. In Fig. 10, we demonstrate the results. It is clear that when the window size is larger, the model



(a) Recall@20 on Gowalla. (b) MRR@20 on Gowalla.



(c) Recall@20 on LastFM. (d) MRR@20 on LastFM.

Figure 10: Results of different window size.

can achieve a better recommendation performance because it can utilize more data to update itself.

5.4 Parameter Sensitivity

In this section, we conduct experiments to evaluate the parameter sensitivity of our GAG model.

5.4.1 ***Embedding Size.*** The previous methods achieve the best results when the embedding size is set to 50 or 100. Therefore, we test the following variants of our GAG model with the embedding size of [50, 100, 200, 400]: **GAG-50**, **GAG-100**, **GAG** and **GAG-400**.

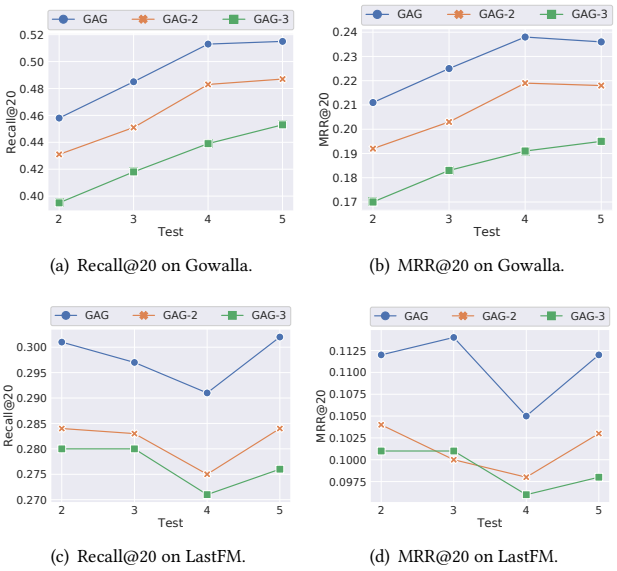
In Fig. 9, results of the sensitivity of the embedding size are presented. It is clear that when the embedding size is set to 200, the GAG model has the highest performance in all situations. Size 100 is a relatively strong variant when compared with 50 and 400. When the embedding size is set as 50 and 400, they are unrepresentative and over-parameterized in their respective methods, which causes difficulty in training a strong model.

5.4.2 Number of Layers. The number of GAG layers controls the depth of the model. We test our model with different numbers of layers of [1, 2, 3]: **GAG**, **GAG-2** and **GAG-3**.

In Fig. 11, the result of different layers is presented. Generally, GNN models always suffer from an increase in the depth of the model because of the gradient explosion. In our experiment, the performance of the GAG model decreases as the model goes deeper, which is consistent with the common observation. Furthermore, the connectivity of sessions is smaller than the traditional graph data, which also limits the power of deeper GNN models.

6 CONCLUSION

In this paper, we proposed a GAG model with a Wasserstein reservoir to perform SSR. We addressed the problem of how to preserve users' long-term interests by introducing the global attribute and the GAG layer. We designed an effective and generic Wasserstein



(a) Recall@20 on Gowalla. (b) MRR@20 on Gowalla.

(c) Recall@20 on LastFM. (d) MRR@20 on LastFM.

Figure 11: Results of different numbers of layers.

reservoir, which samples sessions according to the Wasserstein distance between their recommendation results and the real interactions. In the future, it is significant to investigate how to incorporate the cross-session information for the SSR problem.

7 ACKNOWLEDGMENTS

The work has been supported by Australian Research Council (Grant No. DP190101985, DP170103954 and FT200100825).

REFERENCES

[1] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinícius Flores Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Çaglar Gülçehre, Francis Song, Andrew J. Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. 2018. Relational inductive biases, deep learning, and graph networks. *CoRR* abs/1806.01261 (2018).

[2] Òscar Celma. 2010. *Music Recommendation and Discovery - The Long Tail, Long Fail, and Long Play in the Digital Music Space.* Springer.

[3] Shiyu Chang, Yang Zhang, Jiliang Tang, Dawei Yin, Yi Chang, Mark A. Hasegawa-Johnson, and Thomas S. Huang. 2017. Streaming Recommender Systems. In *WWW*.

[4] Chen Chen, Hongzhi Yin, Junjie Yao, and Bin Cui. 2013. TeRec: A Temporal Recommender System Over Tweet Stream. *PVLDB* 6, 12 (2013).

[5] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS*.

[6] Ernesto Diaz-Aviles, Lucas Drumond, Lars Schmidt-Thieme, and Wolfgang Nejdl. 2012. Real-time top-n recommendation in social streams. In *RecSys*.

[7] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural Message Passing for Quantum Chemistry. In *ICML*.

[8] Marco Gori, Gabriele Monfardini, and Franco Scarselli. 2005. A new model for learning in graph domains. In *IJCNN*, Vol. 2.

[9] Lei Guo, Hongzhi Yin, Qinyong Wang, Tong Chen, Alexander Zhou, and Nguyen Quoc Viet Hung. 2019. Streaming Session-based Recommendation. In *SIGKDD*.

[10] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NIPS*.

[11] Jessica B. Hamrick, Kelsey Allen, Victor Bapst, Tina Zhu, Kevin R. McKee, Josh Tenenbaum, and Peter Battaglia. 2018. Relational inductive bias for physical construction in humans and machines. In *CogSci*.

[12] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast Matrix Factorization for Online Recommendation with Implicit Feedback. In *SIGIR*.

[13] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *ICLR*.

[14] Michael Jugovac, Dietmar Jannach, and Mozhgan Karimi. 2018. Streamingrec: a framework for benchmarking stream-based news recommenders. In *RecSys*.

[15] Wang-Cheng Kang and Julian J. McAuley. 2018. Self-Attentive Sequential Recommendation. In *ICDM*.

[16] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

[17] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[18] S. Kullback and R. A. Leibler. 1951. On Information and Sufficiency. *Ann. Math. Statist.* 22, 1 (1951), 79–86.

[19] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural Attentive Session-based Recommendation. In *CIKM*.

[20] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: Short-Term Attention/Memory Priority Model for Session-based Recommendation. In *SIGKDD*.

[21] Chen Ma, Peng Kang, and Xue Liu. 2019. Hierarchical Gating Networks for Sequential Recommendation. In *SIGKDD*.

[22] Ruihong Qiu, Jingjing Li, Zi Huang, and Hongzhi Yin. 2019. Rethinking the Item Order in Session-based Recommendation with Graph Neural Networks. In *CIKM*.

[23] Ruihong Qiu, Jingjing Li, Zi Huang, and Hongzhi Yin. 2020. Exploiting Cross-Session Information for Session-based Recommendation with Graph Neural Networks. *ACM Trans. Inf. Syst.* (2020).

[24] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*.

[25] Steffen Rendle and Lars Schmidt-Thieme. 2008. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *RecSys*.

[26] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. 2000. The Earth Mover's Distance as a Metric for Image Retrieval. *International Journal of Computer Vision* 40, 2 (2000), 99–121.

[27] Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin A. Riedmiller, Raia Hadsell, and Peter Battaglia. 2018. Graph Networks as Learnable Physics Engines for Inference and Control. In *ICML*.

[28] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The Graph Neural Network Model. *IEEE Trans. Neural Networks* 20, 1 (2009).

[29] Guy Shani, Ronen I. Brafman, and David Heckerman. 2002. An MDP-based Recommender System. In *UAI*.

[30] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *CIKM*.

[31] Jianing Sun, Yingxue Zhang, Chen Ma, Mark Coates, Huifeng Guo, Ruiming Tang, and Xiuqiang He. 2019. Multi-graph Convolution Collaborative Filtering. In *ICDM*.

[32] Jiaxi Tang and Ke Wang. [n.d.]. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding.

[33] Jeffrey Scott Vitter. 1985. Random Sampling with a Reservoir. *ACM Trans. Math. Softw.* 11, 1 (1985), 37–57.

[34] Qinyong Wang, Hongzhi Yin, Zhiting Hu, Defu Lian, Hao Wang, and Zi Huang. 2018. Neural Memory Streaming Recommender Networks with Adversarial Training. In *SIGKDD*.

[35] Weiqing Wang, Hongzhi Yin, Zi Huang, Qinyong Wang, Xingzhong Du, and Quoc Viet Hung Nguyen. 2018. Streaming Ranking Based Recommender Systems. In *SIGIR*.

[36] Weiqing Wang, Hongzhi Yin, Shazia Wasim Sadiq, Ling Chen, Min Xie, and Xiaofang Zhou. 2016. SPORE: A Sequential Personalized Spatial Item Recommender System. In *ICDE*.

[37] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *SIGIR*.

[38] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based Recommendation with Graph Neural Networks. In *AAAI*.

[39] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Victor S. Sheng, Jiajie Xu, Fuzhen Zhuang, Junhua Fang, and Xiaofang Zhou. 2019. Graph Contextualized Self-Attention Network for Session-based Recommendation. In *IJCAI*.

[40] Hongzhi Yin, Bin Cui, Ling Chen, Zhiting Hu, and Xiaofang Zhou. 2015. Dynamic User Modeling in Social Media Systems. *ACM Trans. Inf. Syst.* 33, 3 (2015).

[41] Hongzhi Yin, Bin Cui, Xiaofang Zhou, Weiqing Wang, Zi Huang, and Shazia W. Sadiq. 2016. Joint Modeling of User Check-in Behaviors for Real-time Point-of-Interest Recommendation. *ACM Trans. Inf. Syst.* 35, 2 (2016).

[42] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *SIGKDD*.

[43] Lingling Zhang, Hong Jiang, Fang Wang, Dan Feng, and Yanwen Xie. 2019. T-Sample: A Dual Reservoir-Based Sampling Method for Characterizing Large Graph Streams. In *ICDE*.

[44] Andrew Zimdars, David Maxwell Chickering, and Christopher Meek. 2001. Using Temporal Data for Making Recommendations. In *UAI*.