Stage Evolving Graph Neural Network based Dynamic Recommendation with Life Cycles

Lin Meng
IFM Lab
Department of Computer Science
Florida State University
FL, USA
lin@ifmlab.org

Haoran Yang
Faculty of Engineering and IT
University of Technology Sydney
Sydney, Australia
haoran.yang-2@student.uts.edu.au

Jiawei Zhang
IFM Lab
Department of Computer Science
University of California, Davis
CA, USA
jiawei@ifmlab.org

***Abstract*—Existing methods of dynamic recommender systems usually build neural networks or collaborative filtering models according to the absolute timeline, which fails to consider the preferences of users or characteristics of products in their life cycles. With the rise of graph neural networks, researchers have recently proposed methods based on graph neural networks to solve the recommendation problem. However, such models do not consider evolving stages in their life cycles. In this paper, we study the dynamic recommendation problem and propose a novel model named Stage Evolving Graph Neural Network (SEGNN). SEGNN learns the life stage representation by the Gated Stage Attention Unit (GSAU) and then uses a fully connected layer as the rating score predictor. GSAU adopts two types of attention layers to learn the preferences of users and products within one coupled life stage and discover the preferences over life stages, respectively. GSAU also utilizes a memory mechanism to ensure that the evolutionary patterns are maintained in the final representations of users and products. Due to the lack of information caused by data sparsity, SEGNN designs a masked loss function to train the model. Extensive experiments are done on real-world datasets, and the experimental results show the effectiveness of our proposed model.**

I. Introduction

Nowadays, online shopping has become a popular way to shop for consumers due to the wide spread of the pandemic. Existing online stores like Amazon, eBay make a recommendation for their customers. As customers' preferences change over time, those online stores should also recommend different products with respect to time periods. Thus, it is worth investigating the recommendation problem over time. Formally, we name it the dynamic recommendation problem.

There are different approaches to handle the dynamic recommendations based on different data analysis methods [1], [7], and matrix factorization (MF) performs well in the recommendation field, which only requires the records (*i.e.*, ratings) of users. Massive matrix factorization-based works [9], [17] are proposed for dynamic recommendations to model the temporal effects. However, these methods only consider one unified timeline (absolute timeline) [11], failing to realize that the preference or popularity evolutions for users or items are related to the life cycles of users and products. Based on the observation, BiCycle [11] is proposed to use life cycles instead of the absolute timeline, which can capture the evolutionary trends for both users and products. However, it does not distinguish the importance of relations in a life stage. With the great development of deep learning, works like NeuMF [4] use the neural networks to learn the feature vectors of users and products, showing a more powerful representation ability.

In this paper, we formulate the interactions between users and products as a bipartite graph with a life stage index (Dynamic Bipartite Graph defined in definition 1). Similar to Figure 1, users in the same life stage and products in the same life stage connect together. With this bipartite graph, the connections can tell us valuable information about users' evolving preferences in their life cycles. Currently, there are lots of works that give possible solutions on static bipartite graphs as well [16], [22]. However, these models focus on the static graph, and the bipartite graph with a timestamp for recommendation still needs more research.

Problem studied: In this paper, we propose a graph neural network to solve the dynamic recommendation problem with life cycles. The dynamic recommendation problem with life cycles is challenging to address due to the following aspects:

- **Representation Learning based on Life Cycles:** There are extensive MF-based works on user/product embeddings like [7], [18], [20] are based on the absolute timeline, while neural network-based representation methods [16], [22] cannot be used in the dynamic scenario directly. Nevertheless, users' interests or the traits of products are different for different time periods in their life cycles. Moreover, the influence of interactions between users and products can have different contributions to the final representations. Thus, learning representations of users and products with life cycles is challenging.
- **Evolutionary Pattern Learning:** The evolutionary patterns of users or products can be crucial for the recommendation since the patterns reveal the trends of users or products over time. Besides the absolute timeline, how to capture the evolutionary patterns in life cycles still needs to be solved.
- **Recommendation with Sparse Data:** In most dynamic scenarios, sparsity is one of the issues that prevent accurate prediction of ratings [1]. Since a user could only rate a very small proportion of all products, the rating

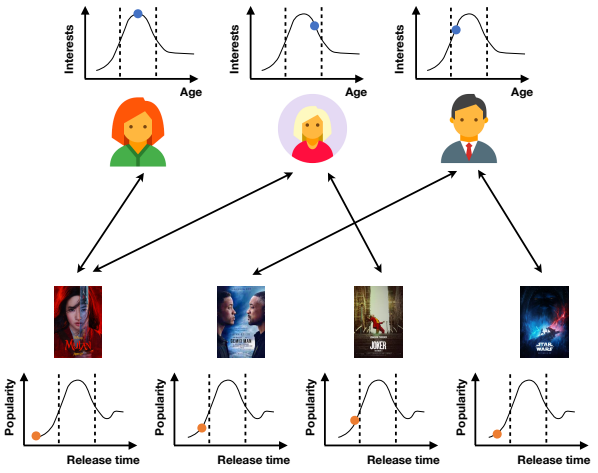


Fig. 1. An Example of Dynamic Bipartite Graph for Coupled Life Stages. Life stages are separated by a dotted line. The blue dot in the life cycles of users and the orange dot in the life cycles of products show the current stage users and products are at, respectively. For the dynamic recommendation problem, since there exist multiple life stages for both users and products, we have multiple copies of the graph according to the number of life stages.

matrix is quite sparse, and the amount of information for estimating a candidate's rating is far from enough. Traditional and deep learning methods suffer from such a data sparsity problem a lot. Thus, how to learn a recommendation model with such sparse data is a challenging problem.

To solve the aforementioned challenges, we represent the historical user-product transaction data as a dynamic bipartite graph. Based on the graph, we propose the Stage Evolving Graph Neural Network (SEGNN) in this paper. SEGNN model first learns the life stage representations for users and products with the Gated Stage Attention Unit (GSAU) and then predicts ratings (*i.e.*, weights on edges) between users and products. GSAU has two learning phrases for the life stage representations for users and products: intra-life-stage representation learning and inter-life-stage learning. Intra-life-stage representation learning learns the preferences among users or products, whereas inter-life-stage learning discovers the evolving features in life stages. After learning the representations of life stages, we can predict the ratings between the users and products. To alleviate the data sparsity problem, we train the model by a masked loss function.

II. Related Work

Our work relates to dynamic recommender system as well as deep learning and graph-based recommender system.

A. Dynamic Recommender System

The goal of a dynamic recommender system is to make recommendations by taking temporal dynamics into consideration. Prior to dynamic models, there was a sort of traditional models like ItemKNN and Matrix Factorization (MF). Sarwar *et al.* proposed an item-based method to do CF by utilizing the similarity among items to make recommendations [13]. Koren *et al.* proposed to obtain latent features of users and items via MF, then making recommendations with the product of latent feature matrices of users and products. There are many varieties of MF. For example, a kernelized MF proposed by Liu *et al.* [12] is one of the successful MF models. MF is one of the most popular methods to make recommendations in the literature. However, these early traditional methods are not expressive enough; they fail to capture the temporal dynamics of users and products. TimeSVD proposed by Koren *et al.* is the very first method to introduce temporal information to fulfill the drawbacks of traditional collaborative filtering methods [17]. The bayesian factorization model is a better model with introducing temporal dynamics proposed by Xiong *et al.* [18]. Temporal dynamics are essential factors for recommender systems. For utilizing temporal factors better, Liu *et al.* proposed BiCycle [11]. BiCycle contains evolution inference called *life cycles* to utilize temporal dynamics. As to evolution inference, this concept was first introduced by Zhang *et al.* [21]. We adopt the concept of *life cycles* in our models to capture temporal dynamics. To our best knowledge, this paper is the first endeavor to incorporate the concept of *life cycles* graph-based recommendation.

B. Deep Learning and Graph-based Recommender System

In order to learn latent factors of users and products from graphs, several researchers have proposed deep learning-based recommender systems. AutoRec proposed by Sedhain *et al.* [14] tries to learn a hidden structure to reconstruct the features of users and items according to the original features, but an identity function learned by AutoRec can not be generalized to unobserved interactions. Neural Collaborative Filtering (NeuMF) [4] is a combination of MF and Multi-Layer Perceptron (MLP). Due to simple structures and lack of extra information, the models mentioned above are less expressive than our proposed models.

Nowadays, graph neural networks give better interpretability to deep recommender systems. SpectralCF is not only a deep model but also a graph-based model which models collaborative filtering signals in spectral domain. In graph neural networks, Graph Attention Networks (GATs) attract a lot of researchers' interests. There are a series of GATs like [10], [15]. GATs seek ways to fuse the neighboring nodes to learn a new representation. The major difference is that GATs adopt the attention mechanism to assign larger weights to those nodes that are more important. The attention weight is learned together with neural network parameters. To collect as much attention information as possible, NGCF [16] considers both the first-order and the higher-order propagations based on the message-passing model.

III. Terminology and Problem Definition

A. Notations and Definitions

Throughout this paper, the lowercase letters denote (*e.g.*, x) the scalars. The lowercase boldface letters (*e.g.*, $\mathbf{x}$) denote the vectors and the uppercase boldface letters *e.g.*, $\mathbf{X}$ represent the matrices. Let $\mathbf{x}(i)$ denote the i^{th} element in vector $\mathbf{x}$, the

$\mathbf{X}(i,:)$ and the $\mathbf{X}(:,j)$ denote the i_{th} row vector and the j_{th} column vector of matrix $\mathbf{X}$, respectively. $\mathbf{X}(i,j)$ denotes the element in the i^{th} row and j^{th} column in $\mathbf{X}$. We use $\sqcup$ and $\cdot$ to denote concatenation operation and scalar multiplication of matrices, respectively. The $\odot$ denotes the element-wise matrix product. The F-norm of a matrix is denoted as $||\cdot||_F$. To better formulate our model , we define the following terms as:

DEFINITION *1:* (Dynamic Bipartite Graph) The dynamic bipartite graph studied in recommender system can be represented as $G = (\mathcal{U}, \mathcal{P}, \mathcal{E})$ for a certain time period $[\tau_s, \tau_e)$, where $\mathcal{U}$ and $\mathcal{P}$ are the sets of users and products, and $\mathcal{E}$ is the edge set describing the relationships only between the user set $\mathcal{U}$ and product set $\mathcal{P}$. An edge $e_{ij} = (u_i, p_j, r_{ij}, \tau) \in \mathcal{E}$ is consisted of four elements those are user $u_i \in \mathcal{U}$, product $p_j \in \mathcal{P}$, r_{ij} denotes the user u_i's rating score of product p_j and the timestamp $\tau \in [\tau_s, \tau_e)$ for the rating.

To capture the evolutionary patterns of users and products, we give the definitions of the life cycle and life stage to interpret the evolution of users and products.

DEFINITION *2:* (Life Cycle) In the system with users and products, we assume life cycles exist in both users and products. For the life cycles of users or products, we suppose each user has at most M life stages, and each product has at most N stages.

DEFINITION *3:* (Life Stage) There are multiple life stages for a user or a product. The l-th life stage of user u_i or product p_i is denoted by time window $[\tau_{sl}^{(i)}, \tau_{el}^{(i)})$. Specially, we consider the M-window $[\tau_{sM}^{(i)}, \infty)$ or $[\tau_{sN}^{(i)}, \infty)$ with no explict end time. This is sensible since in real-world evolutionary patterns, users or products usually enter a stable stage eventually at some time point.

DEFINITION *4:* (Dynamic Rating Matrix) Given a dynamic bipartite graph $G = (\mathcal{U}, \mathcal{P}, \mathcal{E})$ as well as the m^{th} user life stage $[\tau_{sm}^{(i)}, \tau_{em}^{(i)})$ and n^{th} product life stage $[\tau_{sn}^{(j)}, \tau_{en}^{(j)})$. If $(u_i, p_j, r_{ij}, \tau) \in \mathcal{E}$, we can represent the corresponding dynamic rating matrix $\mathbf{R}^{(mn)} \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{P}|}$ for the m^{th} user life stage and n^{th} product life stage as:

$$\mathbf{R}^{(mn)}(i,j) = r_{ij}, \tau \in [\tau_{sm}^{(i)}, \tau_{em}^{(i)}) \wedge [\tau_{sn}^{(j)}, \tau_{en}^{(j)}) \quad (1)$$

Suppose both users and products have three life stages, Figure 1 shows an example illustrating the temporal sub-bipartite graph that corresponds $\mathbf{R}^{(21)}$ when users are in the second life stage and products are in the first stage.

B. Life Stage Data Partition

In this paper, we group interactions into life stages as [11]. Suppose we have M user life stages and N product life stages, we fix the length for the preceding $M-1$ or $N-1$ stages for users and products. The length of each life stage for users and products are l_u and l_p, respectively. Here, we cluster interactions into M bins for users and N bins for products and define the i^{th} bin as:

$$\begin{aligned} B_u(i,\tau) &= min(M, \lceil \tfrac{\tau - \tau_0^i}{l_u} \rceil) \\ B_p(j,\tau) &= min(N, \lceil \tfrac{\tau - \tau_0^j}{l_p} \rceil) \end{aligned} \quad (2)$$

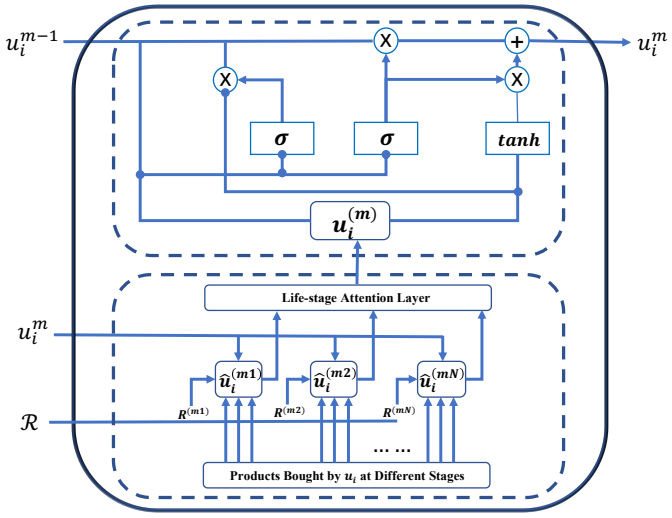


Fig. 2. Gated Stage Attention Unit (GSAU) for User Representation. Product Representation can be formed in a similar way.

where τ_0^i and τ_0^j denote the first observed time of user i and product j, respectively. Regarding the first observed time as its start of life cycle is reliable since it marks the beginning of their experiences. To be honest, it is the experience that truly affects the tastes of users or the characteristics of products. Moreover, the actual age or the date of manufacture has less probability to be available all the time. Therefore, we have $M \times N$ dynamic rating matrices $\{\mathbf{R}^{(mn)} \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{P}|} \mid m \in \{1, \cdots, M\} \wedge n \in \{1, \cdots, N\}\}$. If $(i, j, r_{ij}, \tau) \in \mathcal{E}$ and $m = B_u(i,\tau), n = B_p(j,\tau)$, then $\mathbf{R}^{(mn)}(i,j) = r_{ij}$.

C. Problem Formulation

In this paper, different from the conventional recommendation, we aim to solve the life stage-aware recommendation based on a dynamic bipartite graph, named dynamic recommendation problem. Formally, we give the definition of it.

DEFINITION *5:* (Dynamic Recommendation Problem) Given the user-product temporal bipartite graph $G = (\mathcal{U}, \mathcal{P}, \mathcal{E})$ for the time period $[\tau_s, \tau_e)$ and the life stage partition rules for users and products, a set of dynamic rating matrices $\mathcal{R} = \{\mathbf{R}^{(11)}, \cdots, \mathbf{R}^{(1N)}, \cdots, \mathbf{R}^{(M1)} \cdots, \mathbf{R}^{(MN)}\}$ can be formed. With these matrices, we want to estimate the ratings of a user to the same set of products in the next time period $[\tau_e, \tau_f)$. In other words, we want to predict the rating of a user at a certain life stage to all products at their certain life stage in $[\tau_e, \tau_f)$. If the the predicted u_i's user rating $\hat{r}_{ij}$ for product p_j greater than $\hat{r}_{ij'}$ for product $p_{j'}$, then u_i is more likely to purchase the product p_j than product $p_{j'}$ in the next time period $[\tau_e, \tau_f)$.

IV. PROPOSED METHOD

In order to solve the dynamic recommendation problem, we propose Stage Evolving Graph Neural Network named SEGNN. SEGNN includes three parts: (1) user life stage representation learning, (2) product life stage representation

learning, (3) rating score prediction. To learn the life stage representations of users and products, we apply the Gated Stage Attention Unit (GSAU) to every stage of users and products. Illustrated by Figure 2, the GSAU includes intra-life-stage representation learning and inter-life-stage representation learning. Intra-life-stage representation learning distinguishes the users' taste and the products' characteristics in the life stage and identifies the importance of the life stages for users and products. In contrast, inter-stage representation learning adopts the memory gates to capture the evolutionary patterns. Also, to train the model, we utilize the masked loss function to alleviate the sparsity problem. We discuss the mentioned modules in detail in the following subsections.

A. Intra-life-stage Representation Learning

The intra-life-stage representation learning aims to find users' preferences or the traits of products for coupled life stages. Traditional methods like MF treat all interactions equally important, which may lose some information about identifying user preferences or product characteristics. To learn such preferences for users at m^{th} life stage and products at n^{th} life stage, we adopt the interaction attention layer. Here, assume we have life stage representations of users and products those are $\mathbf{U}^{(m)} \in \mathbb{R}^{|\mathcal{V}|\times H}$ and $\mathbf{P}^{(n)} \in \mathbb{R}^{|\mathcal{P}|\times H}$ (where H is the feature dimension of each user/product vector), respectively. With the corresponding dynamic rating matrix $\mathbf{R}^{(mn)}$, if $\mathbf{R}^{(mn)}(i,j) > 0$, it can be learned by the following attention equation:

$$\mathbf{K}_{u1}^{(mn)}(i,j) = \frac{\exp(\sigma(\mathbf{a}_{u1}^{(mn)^\top}[\mathbf{U}^{(m)}(i,:) \sqcup \mathbf{P}^{(n)}(j,:)]))}{\sum_{k=1}^{|\mathcal{P}|}\exp(\sigma(\mathbf{a}_{u1}^{\top}[\mathbf{U}^{(m)}(i,:) \sqcup \mathbf{P}^{(n)}(k,:)]))} \tag{3}$$

where σ denotes the LeakyRelu activation function. $\mathbf{a}_{u1}^{(mn)} \in \mathbb{R}^{2H}$ denotes the attention variable in the attention layer for users at the m^{th} user stage. $\mathbf{K}_{u1}^{(mn)}(i,j)$ denotes weight scalars of the relation between user i and product j at user stage m. Once we obtain all $\mathbf{K}_{u}^{(mn)}(i,:)$ for all purchased products and $\mathbf{K}_{p}^{(mn)}(j,:)$ for all interacted users, we can form the user representations at m^{th} user life stage, if $\mathbf{R}^{(mn)}(i,j) > 0$:

$$\bar{\mathbf{U}}^{(mn)}(i,:) = \gamma(\sum_{j}\mathbf{K}_{u1}^{(mn)}(i,j)\mathbf{P}^{(n)}(j,:)), \tag{4}$$

where the γ denotes the activation function ELU. Therefore, we obtain the learned $\bar{\mathbf{U}}^{(mn)}$ for users at m^{th} user stage. Similarly, we can have all paired life stages for users $\bar{\mathbf{U}}^{(11)}, \cdots, \bar{\mathbf{U}}^{(MN)}$. Thus, after the interaction attention layer with one dynamic adjacency matrix, we have N user representations for m^{th} user stage. In fact, multiple user/product representations also have practical significances. For example, at the user stage m, the N representations show the consumption habits – at what life stage the product is consumed by consumers. Based on the observations, we adopt the life-stage attention layer to learn the preferences across coupled life stages. Formally, given $\{\bar{\mathbf{U}}^{(m1)}, \cdots, \bar{\mathbf{U}}^{(mN)}\}$ for user stage m, we formulate the life-stage attention layer for m^{th} life stage of users as:

$$\bar{\mathbf{U}}^{(m)}(i,:) = \bigsqcup_{j=1}^{N}\Big(\mathbf{a}_{u2}(1)\cdot\bar{\mathbf{U}}^{(m1)}(i,:)\Big) \tag{5}$$

where the $\mathbf{a}_{u2} \in \mathbb{R}^N$ denote the weight vectors for corresponding representations. To eliminate the minor information and maintain dimensional consistency, we adopt one fully-connected layer (*i.e.*, fc-layer), which can be formulated by:

$$\hat{\mathbf{U}}^{(m)}(i,:) = \mathbf{W}_{fu}\bar{\mathbf{U}}^{(m)}(i,:) + \mathbf{b}_{fu} \tag{6}$$

where the $\mathbf{W}_{fu}$ denotes the weights of corresponding fc-layers. By applying the life-stage attention layer to all life stages, we have all life stage representations for users $\hat{\mathbf{U}}^{(1)}, \cdots, \hat{\mathbf{U}}^{(M)}$, and all life stage representations for products $\hat{\mathbf{P}}^{(1)}, \cdots, \hat{\mathbf{P}}^{(N)}$ is learned similarly. So far, two attention layers learn the unique life stage representations for users and products but still, fail to capture the evolutionary features during the learning process. To maintain the continuous evolutionary features over time, we propose the inter-life-stage representation learning.

B. Inter-life-stage Representation Learning

The evolutionary patterns contain important information about the growth trends of users and products. To take advantage of the evolutionary trend, we use one storage gate and one forget gate to learn the inter-life-stage representations. Combining two gates can retain useful information from the previous life stage and discard the useless information (*e.g.*, noise) in the current life stage. Similar to LSTM [5], the storage gate learns a storage coefficient to decide how much information from the previous life stage is maintained and the forget gate learns a discard coefficient to determine what is discarded from the current input. Formally, we compute the storage coefficient z and the discard coefficient g as:

$$\begin{aligned} z &= \delta(\mathbf{W}_z(\mathbf{U}^{(m-1)}(i,:) \sqcup \mathbf{U}^{(m)}(i,:))) \\ g &= \delta(\mathbf{W}_g(\mathbf{U}^{(m-1)}(i,:) \sqcup \mathbf{U}^{(m)}(i,:))) \end{aligned} \tag{7}$$

where the $\mathbf{W}_z$ and $\mathbf{W}_g$ are the learnable weights for two coefficients, respectively. The δ denotes the Sigmoid function. With z and g, we can get the updated user representation at life stage m as follows:

$$\begin{aligned} \mathbf{U}^{(m)}(i,:) = \quad & (1-z)\cdot\hat{\mathbf{U}}^{(m-1)}(i,:) + \\ & z\cdot\tanh(\mathbf{W}_{zt}(g\cdot(\hat{\mathbf{U}}^{(m-1)}(i,:) \sqcup \mathbf{U}^{(m)}(i,:)))). \end{aligned} \tag{8}$$

where the $\mathbf{W}_{zt}$ is the weight to make sure the same dimensions. We can also apply the memory mechanism to the product life stage representation learning for $\mathbf{P}^{(m)}(i,:)$ in a similar way.

C. Framework Training with Masked Loss Function

To predict the rating, we need to predict the relationships between nodes (*i.e.*, edge weights between users and products). Here, to form the edge representations, we concatenate the

representations of two linked nodes. Then, one fully-connected layer is used as the predictor. The prediction can be represented as:

$$\hat{\mathbf{R}}^{(mn)}(i,j) = \mathbf{W}_{pred}(\mathbf{U}^{(m)}(i:) \sqcup \mathbf{P}^{n}(j:)) + b_{pred}. \quad (9)$$

where the $\mathbf{W}_{pred} \in \mathbb{R}^{2H}$ and $b_{pred} \in \mathbb{R}$ are the weight and bias in the predictor. $\hat{\mathbf{R}}^{(mn)}$ contains all predicted ratings of user u_i to product p_j.

However, the recommender system usually suffers from severe data sparsity. However, in real-world applications, the phenomenon that a user does not consume the product does not imply the user dislikes or never consumes the product. The reason can be various, such as ineffective publicity, slow promotion, or inappropriate retrieval error, *etc.*. To avoid such a problem, we make use of the non-existing edges. Thus, simply setting weights of non-existing edges to a fixed number (i.e., 0.5) is not realistic since how much information is provided by non-existing edges is unknown. Here, we utilize a hyper-parameter α to control the extent to which we regard the weights of non-existing edges. Now, the loss function for the SEGNN is

$$\mathcal{L} = \sum_{m=1}^{M}\sum_{n=1}^{N} ||\mathbf{C}^{(mn)} \odot (\hat{\mathbf{R}}^{(mn)} - \mathbf{R}^{(mn)})||_F \quad (10)$$

where $\mathbf{C}$ is the mask matrix for each edge and it can be defined as:

$$\mathbf{C}^{(mn)}(i,j) = \begin{cases} 1 & \text{if } \mathbf{R}^{(mn)}(i,j) > 0 \\ \alpha & \text{otherwise} \end{cases}. \quad (11)$$

The general procedure of SEGNN is shown in Algorithm 1. Overall, in each epoch, we train the M user life stage to obtain user embeddings and then train on N product life stage to obtain product embeddings. Lastly, we compute the masked loss and update all parameters by backpropagation.

Algorithm 1: SEGNN Model

Result: The life stages representations for users $\mathcal{X}_u = \{\mathbf{X}_u^{(1)}, \cdots, \mathbf{X}_u^{(M)}\}$ and products $\mathcal{X}_p = \{\mathbf{X}_p^{(1)}, \cdots, \mathbf{X}_p^{(N)}\}$

Input: Random Initialization for $\mathbf{X}_u^{(0)}$ and $\mathbf{X}_p^{(0)}$ and set $\mathcal{X}_u$ and $\mathcal{X}_p$ zero; the dynamic rating matrices $\mathbf{R}^{(11)}, \cdots, \mathbf{R}^{(MN)}$

while $i <$ *Epochs* **do**
  for *user stage* m **do**
    $\hat{\mathbf{X}}_u^{(m)}$= GSAU$\Big(\hat{\mathbf{X}}_u^{(m-1)}, \mathbf{X}_u^{(m)}, \Big(\mathbf{X}_p^{(1)}, \cdots, \mathbf{X}_p^{(N)}\Big), \big(\mathbf{R}^{(m1)}, \cdots, \mathbf{R}^{(mN)}\big)\Big)$;
  end
  for *item stage* n **do**
    $\hat{\mathbf{X}}_p^{(n)}$= GSAU$\Big(\hat{\mathbf{X}}_p^{(n-1)}, \mathbf{X}_p^{(n)}, \Big(\mathbf{X}_u^{(1)}, \cdots, \mathbf{X}_u^{(M)}\Big), \big(\mathbf{R}^{(1n)}, \cdots, \mathbf{R}^{(Mn)}\big)\Big)$;
  end
  Compute $\mathcal{L}$ with updated $\mathcal{X}_u$ and $\mathcal{X}_p$;
  Update the model variables via backpropagation;
  $i = i + 1$;
end

TABLE I
STATISTICS OF THE DATASETS.

Dataset	Users	products	Ratings	Density
ML 100K	943	1,682	100,000	6.30%
AMI	1,429	900	10,261	0.80%

V. EXPERIMENTS

In this section, we introduce the datasets and the compared methods we used in the experiments first. Then, the experimental settings and evaluation metrics are provided. Next, the experimental results on two real-world datasets are discussed. To conduct a comprehensive study of the SEGNN model, we also provide the parameter analysis, discussion on the effectiveness of our model on sparse data, and the ablation study on the life-stage attention layer and two gates for storage and forget in GSAU. We use two publicly accessible datasets. The statistics of them are summarized in Table I.

A. Comparison Methods

To demonstrate the effectiveness of our models, we compare the SEGNN from three aspects. (1) Traditional methods: We adopt **POP** [19] that recommend popular products to users, and **ItemKNN** [13] that aims to recommend similar products for a user based on their similarities, and **MF** [8]. (2) MF-based methods: **BiCycle** [11] that solves dynamic recommendation problem based MF, **NeuMF** [4] that adopts generalized matrix factorization by neural networks. (3) Graph-based methods: **SpectralCF** [22] that adopts a novel convolution operation in the spectral domain and **GAT** [15] that integrates the features of neighboring nodes with attention in static graph. We also propose three variants. **P-SENN** which only has the interaction attention layer; **SENN** which keep the interaction attention layer and life-stage attention layer in GSAU without the storage gate and forget gate; **SEGNN** that is the full version of proposed.

B. Experimental Setting And Evaluation Metrics

1) Experimental Setting: For all datasets, we sort all of them according to timestamps and then divide each dataset into 29 time windows as discussed in [11] with equal depth. We select time window $1 < c \leq k$ as train windows. Then, we obtain the top 20% data in test window c, together with all data in time windows $1, \cdots, c-1$ as the training set. The remaining 80% data in c is the test set. To make a fair comparison, we set the number of user life stages and the number of product life stages are $m = 4$ and $n = 2$, respectively. Additionally, we set the hidden size for all neural networks as 32. The learning rate of our models is 0.01. We set the parameter to evaluate the significance of unobserved interactions between users and products as $\alpha = 0.8$. For parameters in our models, we use Glorot initialization [3].

2) Evaluation Metrics: We adopt three commonly used metrics to evaluate the recommendation results of our methods and comparison methods. They are Recall (Rec@k), Precision (Prec@k) and Mean Reciprocal Rank (MRR), which are adopted by [4], [11], [22].

C. Experimental Results

We use the last ten windows as testing windows to conduct experiments, and the rest are the training windows. Due to the space limit, we only present the results of three windows out of ten. The experimental results are shown in Table II, where Prec@15, Rec@15, and MRR evaluate all comparison methods. Generally, the proposed model has the best performance due to its stable performance on all windows and metrics.

We first study the effectiveness of our models by comparing them with traditional baselines: MF, ItemKNN, and POP. These traditional baselines try to illustrate the interactions between users and products via matrix factorization or k-Nearest Neighbors [6]. However, two of these ways are lack of interpretability. Secondly, traditional baselines fail to take temporal factors into consideration. Thus, it is hard for them to capture the temporal continuity of user features or product features. Prec@15, Recall@15 and MRR in Table II indicate that our proposed models perform better than these traditional baselines.

Then, we compare our proposed models with BiCycle. This method is the first framework introducing the concept of *life cycles*. With temporal information, BiCycle performs better than traditional baselines in most situations. However, BiCycle treats all products equally important while SEGNN considers the different importances of products. The results support our intuition of a better interpretation of deep models. Both SpectralCF and NeuMF adopt graph neural networks. SpectralCF learns the features via the spectral domain, while NeuMF uses generalized matrix factorization and multi-layer perceptron. The results show that these two models have the worst performance, showing that the two models cannot perform well with temporal information. GAT is the one that is the most similar to our proposed methods. Though GAT achieves several relatively high scores in our experiments, there are gaps between GAT and our models. This result indicates that temporal dynamics and the significance of unobserved interactions can promote the performance of GAT. SEGNN also introduces an attention mechanism; however, we not only utilize neighbor features but also apply them to capture temporal dynamics among different life stages.

We should also note that BiCycle and MF perform quite well on the specific test windows, though our methods still outperform them in general. It is also worth mentioning that two traditional methods outperform our proposed models on some metrics, the gaps between these two traditional methods and our proposed methods are very small. AMI has the lowest density among all the three datasets we used. According to [2], deep models may perform worse on the datasets with a relatively small scale or low density. Later, in section V-E, we show the results of compared deep models on ML100K with different sizes of the training set to illustrate our methods performing better on a sparse training set.

D. Parameter Analysis

In this section, we study the parameter used to evaluate the significance of unobserved interactions between users and products. We conduct experiments with different values of parameter α on ML100K via using model SEGNN. Figure 3 shows the result.

Many works, like [4], [8], [11], take the same measurements to handle observed and unobserved interactions between users and products, which is the same as the situation with $\alpha = 1$ in our experimental settings. According to Figure 3, we note that $\alpha = 1$ is not the best performer. Obviously, it is not proper to treat observed and unobserved equally. Our models can achieve higher results by carefully choosing α.

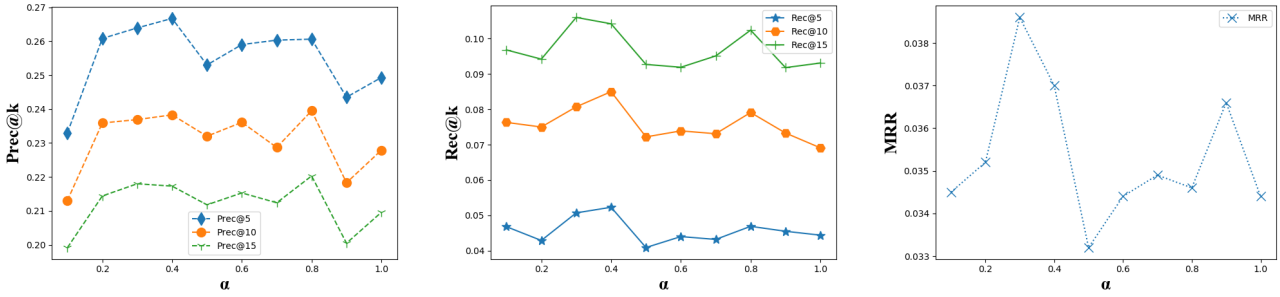
E. Quality of Recommendation on Sparser Data with Deep Models

Performances on sparse datasets are essential metrics to evaluate the effectiveness of deep models. To verify that gathering neighbor features and introduce temporal dynamics that tackle the sparsity problem, we conduct experiments to show the quality of recommendation of NeuMF, SpectralCF, GAT, and SEGNN on a sparser dataset in this section. We randomly sample subsets on the whole training set, which are obtained by methods described in section V-B1. The sizes of the subsets are $\{10\%, 20\%, \cdots, 90\%, 100\%\}$ of the whole training set, respectively. Since we conduct experiments on one dataset, we choose the last ten time windows as test windows to ensure the reliability of the results. Other experimental settings are the same as the settings in section V-B1.

According to the results, we note that SEGNN and NeuMF perform better with the percentage of the training set growing, which means that our method and NeuMF are sensitive to the sparseness of the training set. Performances of SpectralCF and GAT are quite stable. Hence, SpectralCF and GAT can be trained by sparser datasets to achieve relatively good performance. We do not need to feed dense datasets to these models. Figure 4 shows that though SEGNN is not the best performer in the very beginning, its scores are quite high at the beginning and grow quickly with the scale of the training set growing. Soon after, SEGNN becomes the best performer. This phenomenon is more obvious on Rec@5 and MRR. Additionally, this phenomenon is reasonable because SEGNN can obtain more extra information from less data growth due to two attention layers and the gates for storage and forget. The result indicates that our proposed model can tackle the dataset sparsity problem. The results verify our intuition that deep models should perform well on sparse datasets by introducing extra information, for example, temporal dynamics and neighbor features. In this case, we explore ideas and methods to discover more extra information under the situation that lacks metadata.

TABLE II
PREC@15(RANK), REC@15(RANK) AND MRR(RANK). THE BEST PERFORMER IS IN **BOLDFACE**.

		Window 1			Window 2			Window 3			Ave.
Dataset	Method	Prec@15	Recall@15	MRR	Prec@15	Recall@15	MRR	Prec@15	Recall@15	MRR	Rank
ML100K	SEGNN	0.1692(2)	**0.1029(1)**	**0.0390(1)**	0.1683(2)	0.0965(2)	0.0303(5)	0.1775(3)	0.1113(2)	0.0361(2)	**2.2**
	SENN	**0.1721(1)**	0.0966(3)	0.0370(2)	0.1608(4)	0.0670(5)	0.0221(7)	0.1784(2)	0.1100(3)	0.0333(3)	3.3
	P-SENN	0.1622(4)	0.0758(4)	0.0360(3)	0.1667(3)	0.0842(4)	0.0335(3)	0.1775(3)	**0.1194(1)**	**0.0385(1)**	2.9
	GAT	0.1652(3)	0.0986(2)	0.0332(4)	**0.1758(1)**	**0.1206(1)**	**0.0454(1)**	0.1756(5)	0.0975(4)	0.0289(6)	3.0
	SpectralCF	0.0106(10)	0.0101(10)	0.0222(6)	0.0124(10)	0.0134(10)	0.0114(9)	0.0128(10)	0.0101(10)	0.0084(10)	9.4
	NeuMF	0.1234(7)	0.0468(7)	0.0190(8)	0.1042(7)	0.0666(6)	0.0309(4)	0.1408(8)	0.0477(8)	0.0231(8)	7.0
	Bicycle	0.1552(5)	0.0663(5)	0.0258(5)	0.1608(4)	0.0853(3)	0.0367(2)	0.1465(7)	0.0631(7)	0.0293(5)	4.8
	MF	0.1045(8)	0.0366(8)	0.0140(9)	0.0842(8)	0.0522(8)	0.0221(7)	**0.1850(1)**	0.0870(5)	0.0312(4)	6.4
	ItemKNN	0.0448(9)	0.0169(9)	0.0097(10)	0.0383(9)	0.0155(9)	0.0073(10)	0.0451(9)	0.0219(9)	0.0097(9)	9.2
	POP	0.1463(6)	0.0570(6)	0.0193(7)	0.1267(6)	0.0564(7)	0.0231(6)	0.1474(6)	0.0641(6)	0.0256(7)	6.3
AMI	SEGNN	**0.0152(1)**	**0.0809(1)**	**0.0303(1)**	**0.0271(1)**	**0.1335(1)**	0.0478(3)	**0.0196(1)**	0.1074(2)	0.0382(1)	**1.3**
	SENN	0.0139(5)	0.0696(5)	0.0283(5)	0.0260(2)	0.1233(2)	**0.0481(1)**	0.0191(3)	0.1038(3)	0.0373(3)	3.2
	P-SENN	0.0139(5)	0.0696(5)	0.0283(5)	0.0260(2)	0.1233(2)	**0.0481(1)**	0.0191(3)	0.1038(3)	0.0373(3)	3.2
	GAT	**0.0152(1)**	0.0718(4)	0.0285(4)	0.0248(4)	0.1201(4)	0.0474(4)	**0.0196(1)**	**0.1077(1)**	0.0381(2)	2.8
	SpectralCF	0.0016(9)	0.0084(9)	0.0040(10)	0.0027(9)	0.0126(10)	0.0117(8)	0.0025(10)	0.0135(10)	0.0074(10)	9.4
	NeuMF	0.0012(10)	0.0034(10)	0.0052(9)	0.0024(10)	0.0215(9)	0.0058(10)	0.0047(9)	0.0364(9)	0.0085(9)	9.4
	Bicycle	0.0145(3)	0.0721(2)	0.0289(2)	0.0236(5)	0.1082(5)	0.0462(5)	0.0181(5)	0.0863(5)	0.0370(5)	4.1
	MF	0.0145(3)	0.0721(2)	0.0289(2)	0.0236(5)	0.1082(5)	0.0462(5)	0.0181(5)	0.0863(5)	0.0370(5)	4.1
	ItemKNN	0.0067(8)	0.0457(8)	0.0182(8)	0.0041(8)	0.0354(8)	0.0115(9)	0.0072(8)	0.0523(8)	0.0230(8)	8.1
	POP	0.0091(7)	0.0514(7)	0.0196(7)	0.0195(7)	0.0970(7)	0.0398(7)	0.0150(7)	0.0825(7)	0.0370(5)	6.8



(a) Prec@k on the dataset ML100K (b) Rec@k on the dataset ML100K (c) MRR@k on the dataset ML100K

Fig. 3. Prec@k, Rec@k and MRR of SEGNN on the Dataset ML100K with Different α

TABLE III
ABLATION STUDY ON THREE MODULES

Methods	ML100K			AMI		
	Prec@15	MRR	Imp.	Prec@15	MRR	Imp.
P-SENN	0.1709	0.0329	-	0.0197	0.0318	-
SENN	0.1738	0.0318	0.823%	0.0197	0.0379	0.305%
SEGNN	0.1776	0.0365	8.483%	0.0206	0.0388	3.472%

F. Ablation Study

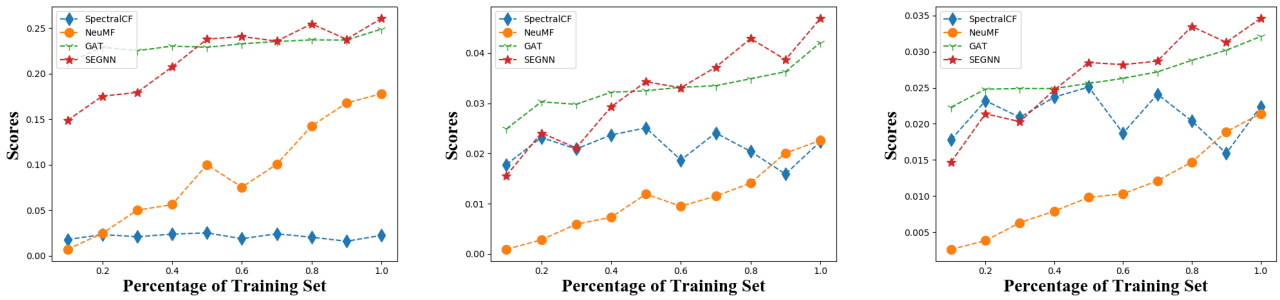
We also study the effectiveness of the life-stage attention layer as well as two gates for capturing the evolutionary patterns added to P-SENN. We conduct several experiments on the two datasets. We choose mean results to show the improvement for quantitative evaluation. Averaged Prec@15 and MRR are selected as metrics. We also give *Improve* (*i.e.*, Imp. is the average improvement of two metrics) to illustrate the improvement of our proposed methods with extra components with comparing with the previous version.

According to the results in Table III, SENN performs lightly worse than the P-SENN on the dataset ML100K, and achieves better results on the dataset AMI. When we calculate the neighbor features, we adopt a hidden layer in our deep model to do that instead of adopting a simple linear weighted average. That is because users in different life stages may have different preferences for products in different life stages.

For the two-gate component, as the experimental results show, we note that the component does have a positive impact on the experimental results of both datasets. Thus, we believe that the memory mechanism captures the evolutionary patterns in iteratively updating the user or product representations.

VI. CONCLUSION

In this paper, we propose a novel model SEGNN incorporating life cycles for the dynamic recommendation problem. SEGNN first learn the representations of users and products



(a) Prec@5 with Different Training Percentage (b) Rec@5 with Different Training Percentage (c) MRR@5 with Different Training Percentage

Fig. 4. Prec@k, Rec@k (k=5) and MRR of Four Deep Models on Training Sets with Different Sampling Ratios.

by the GSAU. GSAU learns the representations of each life stage for both users and products by two attention layers as well as two gates. The two attention layers learn the preferences lying in interactions and life stages, respectively, while the two gates capture the evolving trends of users and products. To get enough information about ratings, we take advantage of the mask matrix to decide the extent of the utilization of unobserved edges. Extensive experiments are conducted on two real-world datasets. The experimental results and analyses show the effectiveness of the proposed model.

VII. Acknowledgement

This work is partially supported by NSF through grants IIS-1763365 and IIS-2106972.

References

[1] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge & Data Engineering*, pages 734–749, 2005.

[2] Jayme Garcia Arnal Barbedo. Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification. *Computers and Electronics in Agriculture*, 153:46 – 53, 2018.

[3] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[4] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 173–182, Republic and Canton of Geneva, Switzerland, 2017. International World Wide Web Conferences Steering Committee.

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

[6] J. M. Keller, M. R. Gray, and J. A. Givens. A fuzzy k-nearest neighbor algorithm. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-15(4):580–585, July 1985.

[7] Yehuda Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 447–456. ACM, 2009.

[8] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, August 2009.

[9] Neal Lathia, Stephen Hailes, and Licia Capra. Temporal collaborative filtering with adaptive neighbourhoods. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 796–797, New York, NY, USA, 2009. ACM.

[10] John Boaz Lee, Ryan Rossi, and Xiangnan Kong. Graph classification using structural attention. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, pages 1666–1674, New York, NY, USA, 2018. ACM.

[11] X. Liu, Y. Song, C. Aggarwal, Y. Zhang, and X. Kong. Bicycle: Item recommendation with life cycles. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 297–306, Nov 2017.

[12] Bithika Pal and Mamata Jenamani. Kernelized probabilistic matrix factorization for collaborative filtering: Exploiting projected user and item graph. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, pages 437–440, New York, NY, USA, 2018. ACM.

[13] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM.

[14] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, pages 111–112, New York, NY, USA, 2015. ACM.

[15] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[16] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. In *Proceedings of the 42Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, pages 165–174, New York, NY, USA, 2019. ACM.

[17] Liang Xiang and Qing Yang. Time-dependent models in collaborative filtering based recommender system. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, WI-IAT '09, pages 450–457, Washington, DC, USA, 2009. IEEE Computer Society.

[18] Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff Schneider, and Jaime G. Carbonell. *Temporal Collaborative Filtering with Bayesian Probabilistic Tensor Factorization*, pages 211–222. 2010.

[19] Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. Personalized entity recommendation: A heterogeneous information network approach. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14, pages 283–292, New York, NY, USA, 2014. ACM.

[20] Chenyi Zhang, Ke Wang, Hongkun Yu, Jianling Sun, and Ee-Peng Lim. *Latent Factor Transition for Dynamic Collaborative Filtering*, pages 452–460.

[21] Yizhou Zhang, Yun Xiong, Xiangnan Kong, and Yangyong Zhu. Netcycle: Collective evolution inference in heterogeneous information networks. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1365–1374, New York, NY, USA, 2016. ACM.

[22] Lei Zheng, Chun-Ta Lu, Fei Jiang, Jiawei Zhang, and Philip S. Yu. Spectral collaborative filtering. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, pages 311–319, New York, NY, USA, 2018. ACM.