Stage Evolving Graph Neural Network based Dynamic Recommendation with Life Cycles

Lin Meng
Department of Computer Science
Florida State University
lin@ifmlab.org

Haoran Yang
Department of Computer Science,
University of Technology Sydney
haoran.yang-2@student.uts.edu.au

Jiawei Zhang
Department of Computer Science
Unversity of California, Davis
jiawei@ifmlab.org

ABSTRACT

The dynamic recommender systems have always been one of the most interesting research topics among researchers. Existing methods 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 a life stage. With the rise of graph neural networks, researchers have recently proposed methods based on graph neural networks to solve the recommendation problem. However, such graph neural network-based works do not make use of temporal information. In this paper, we study the dynamic recommendation problem from the graph perspective and then 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 uses 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 adopts a memory mechanism to ensure that the evolutionary patterns can be maintained in the final representations of users and products. Due to the lack of information caused by data sparsity, SEGNN makes use of 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.

KEYWORDS

Dynamic Recommender System; Graph Neural Network

ACM Reference Format:
Lin Meng, Haoran Yang, and Jiawei Zhang. 2018. Stage Evolving Graph Neural Network based Dynamic Recommendation with Life Cycles. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Nowadays, online shopping becomes a popular way to shop for consumers. Existing online stores like Amazon, eBay make recommendations for their customers. The recommendation goal is to make the customers satisfy the products recommended by the online shopping mall. As customers' preferences change over time, the recommendation becomes challenging to make customers satisfied all the time. Thus, it is worth investigating the recommendation problem with temporal information. Formally, we name it the dynamic recommendation problem.

Collaborative filtering [2, 8] performs the best in the recommendation field, which only requires the records (*i.e.*, ratings) of users. Massive collaborative filtering-based works are proposed for dynamic recommendations to model the temporal effects. TimeSVD adds the time bias term into the conventional SVD [18]. [11] uses the Gaussian process to capture the users' preferences. In contrast, the other types of methods utilize additional information such as locations or clicks to enhance the performance of dynamic recommendation [13, 23]. However, these methods only consider one unified timeline (absolute timeline) [12], failing to realize that the preference or popularity evolutions are related to the life cycles of users and products. Based on the observation, BiCycle [12] is proposed to use life cycles instead of the absolute timeline, which can capture both the evolutionary trends for both users and products. However, it does not distinguish the importance of relations in a life stage for users and products. With the development of deep learning, works like NeuMF [7], NFM [6] uses the neural networks to learn the feature vectors of users and products, showing more powerful representation ability.

In this paper, we propose a bipartite graph neural network to solve the dynamic recommendation problem with life cycles by treating the data as a bipartite graph. The dynamic recommendation problem with life cycles is difficult to address due to the following challenges:

- **Representation Learning based on Life Cycles:** There are extensive CF-based works on user/product embeddings like [8, 19, 20] are based on the absolute timeline, while neural network-based representation methods [16, 22] cannot be used in the dynamic scenario directly. Nevertheless, the interests of users or the traits of products are different during their life cycles. 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, capturing 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 prevents accurate prediction of ratings [2]. Traditional methods (*i.e.*, CF) and deep learning methods suffer from such data sparsity

problems. Thus, how to learn a recommendation model with such sparse data is a challenging problem.

In this paper, we represent the historical user-product transaction data as a dynamic bipartite graph to solve the challenges mentioned earlier. Based on the graph, we propose the Stage Evolving Graph Neural Network (SEGNN). 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 inherent in life cycles. After learning the representations of life stages, we can predict the ratings between the users and products. To overcome the data sparsity, we train the model by a masked loss function to improve the performance.

2 TERMINOLOGY AND PROBLEM DEFINITION

This section defines a dynamic bipartite graph and corresponding dynamic rating matrix and formulates the studied problem.

Definition 1. *(Dynamic Bipartite Graph) Given $G = (\mathcal{U}, \mathcal{P}, \mathcal{E})$ 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 define each user/item with a life cycle, and we suppose that each user has at most M life stages and each product has at most N stages. The l-th life stage of user u_i or product p_i is denoted by time window $[\tau_{sl}^{(i)}, \tau_{el}^{(i)})$. Especially, we consider the M-window $[\tau_{sM}^{(i)}, \infty)$ or $[\tau_{sN}^{(i)}, \infty)$ with no explicit end time. The time window partition is reasonable since users or products usually enter a stable stage eventually at some time point in a real-world evolutionary pattern. Based on the previous definition, we can have the dynamic rating matrix definition as follows:

Definition 2. *(Dynamic Rating Matrix) Given one 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 $[\tau_{sn}^{(j)}, \tau_{en}^{(j)})$, then 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 :*

$$\mathbf{R}^{(mn)}(i,j) = \begin{cases} r_{ij} & \textit{if } (u_i, p_j, r_{ij}, \tau) \in \mathcal{E}, \tau \in [\tau_{sm}^{(i)}, \tau_{em}^{(i)}) \wedge [\tau_{sn}^{(j)}, \tau_{en}^{(j)}) \\ 0 & \textit{otherwise} \end{cases}$$

2.1 Life Stage Data Partition

Before we give the problem formulation, we need to partition the data first. In this paper, we follow the life stage partition as in [12]. 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. There exist M^{th} and N^{th} stage for users and products, which are left with an open end. Based on the rules laid down, we can allocate all the historical interactions of users and products into M and N bins, respectively, where each bin associates with the corresponding life stage. In this case, we have $M \times N$ dynamic rating matrices based on the M user life stages and N product life stages. Given the dynamic bipartite graph $G = (\mathcal{U}, \mathcal{P}, \mathcal{E})$, we have a set of dynamic rating matrices $\left\{\mathbf{R}^{(mn)} \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{P}|} \mid m \in \{1, \cdots, M\} \wedge n \in \{1, \cdots, N\}\right\}$, where $\mathbf{R}^{(mn)}$ contains all interactions given by users of m^{th} stage and products of n^{th} stage, which can be denoted as follows:

$$\mathbf{R}^{(mn)}(i,j) = \begin{cases} r_{ij} & \text{if } (i, j, r_{ij}, \tau) \in \mathcal{E}, m = B_u(i,\tau), n = B_p(j,\tau) \\ 0 & \text{otherwise} \end{cases}$$

where $B_u(i,\tau)$ and $B_p(j,\tau)$ denote the functions that determine the life stage of user i and the life stage of product j at time τ, respectively. Here, we define them as:

$$B_u(i,\tau) = min(M, \lceil \frac{\tau - \tau_0^i}{l_u} \rceil)$$
$$B_p(j,\tau) = min(N, \lceil \frac{\tau - \tau_0^j}{l_p} \rceil)$$

where τ_0^i and τ_0^j denote the first observed time of user i and product j, respectively.

2.2 Problem Formulation

Unlike the conventional recommendation, we aim to solve the dynamic recommendation based on the bipartite graphs and life cycles, named the dynamic recommendation problem. Formally, we define the problem.

Definition 3. *(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 $\mathcal{R} = \left\{\mathbf{R}^{(11)}, \cdots, \mathbf{R}^{(1N)}, \cdots, \mathbf{R}^{(M1)} \cdots, \mathbf{R}^{(MN)}\right\}$ 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 products at their certain life stage in $[\tau_e, \tau_f)$. If the predicted u_i's 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 $[\tau_e, \tau_f)$.*

3 PROPOSED METHOD

In this paper, we propose the Stage Evolving Graph Neural Network named SEGNN. As shown in Algorithm 1, 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 1, 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 life stage pairs and identifies the life stage preferences for users and products, while the inter-stage representation learning adopts the memory gates to capture the evolutionary patterns. Also, to train the model, we

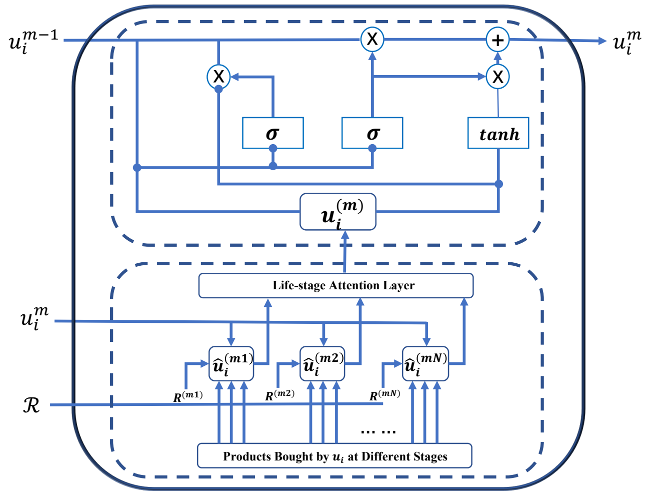


Figure 1: Gated Stage Attention Unit(GSAU) for User Representation. Product Representation can be formed in a similar way.

adopt a masked loss function to solve the sparsity problem. We will discuss each module in detail in the following subsections.

3.1 Intra-Life-Stage Representation Learning

The intra-life-stage representation learning aims to find users' preferences or the traits of products for life-stage pairs. To learn such preferences for users at m^{th} life stage and product at n^{th} life stage, we adopt the interaction attention layer. Assume we have the 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 $\mathbf{R}^{(mn)}$, we adopt the attention mechanism as follows:

$$\mathbf{K}_{u1}^{(mn)}(i,j) = \begin{cases} \frac{\exp(\sigma(\mathbf{a}_{u1}^{(mn)^\top}[\mathbf{U}^{(m)}(i,:)\sqcup\mathbf{P}^{(n)}(j,:)]))}{\sum_{k=1}^{|\mathcal{P}|}\exp(\sigma(\mathbf{a}_{u1}^\top[\mathbf{U}^{(m)}(i,:)\sqcup\mathbf{P}^{(n)}(k,:)]))} & \text{if } \mathbf{R}^{(mn)}(i,j) > 0 \\ 0 & \text{otherwise} \end{cases}$$

where σ denotes the LeakyRelu activation function. $\mathbf{a}_{u1}^{(mn)} \in R^{2H}$ denotes the attention variable in the attention layer for users at the m^{th} user stage and the n^{th} product stage. $\mathbf{K}_{u1}^{(mn)}(i,j)$ denote weight scalars of the relations between user i and product j at user stage m and product stage n. Once we obtain all $\mathbf{K}_u^{(mn)}(i,:)$ for all purchased products, we can formulate the user representations at m^{th} user life stage and n^{th} product life stage as follows:

$$\bar{\mathbf{U}}^{(mn)}(i,:) = \gamma(\textstyle\sum_j \mathbf{K}_{u1}^{(mn)}(i,j)\mathbf{P}^{(n)}(i,:)), \quad \text{if } \mathbf{R}^{(mn)}(i,j) > 0$$

where the γ denotes the activation function ELU. Therefore, we obtain the learned $\bar{\mathbf{U}}^{(mn)}$ for users and $\bar{\mathbf{P}}^{(mn)}$ for products at m^{th} user stage and n^{th} product stage, respectively. Similarly, we can have all paired life stage representations $\bar{\mathbf{U}}^{(11)}, \cdots, \bar{\mathbf{U}}^{(MN)}$ for users and $\bar{\mathbf{P}}^{(11)}, \cdots, \bar{\mathbf{P}}^{(MN)}$ for products. Thus, after the interaction attention layer with one temporal adjacency matrix, we have N user representations for m^{th} user stage or M product representations for n^{th} product 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 i^{th} item life stage, the product will be purchased by users. Based on the observations, we adopt the life-stage attention layer to learn the preferences across the life stages. Formally, given $\left\{\bar{\mathbf{U}}^{(m1)}, \cdots, \bar{\mathbf{U}}^{(mN)}\right\}$ for user stage m, we formulate the life-stage attention layer for m^{th} life stage of users as:

$$\bar{\mathbf{U}}^{(m)}(i,:) = \left(\mathbf{a}_{u2}(1)\cdot\bar{\mathbf{U}}^{(m1)}(i,:)\right) \sqcup \cdots \sqcup \left(\mathbf{a}_{u2}(N)\cdot\bar{\mathbf{U}}^{(mN)}(i,:)\right).$$

where the $\mathbf{a}_{u2} \in \mathbb{R}^N$ denotes the weight vectors. To eliminate the minor information and maintain dimensional consistency, we adopt one fully-connected layer (*i.e.*, fc-layer), which is formulated as follows:

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

where the $\mathbf{W}_{fu}$ denotes the weights of corresponding fc-layers. By applying the life-stage attention layer to all life stages, we can have all life stage representations for users $\hat{\mathbf{U}}^{(1)}, \cdots, \hat{\mathbf{U}}^{(M)}$, and learn all life stage representations for products $\hat{\mathbf{P}}^{(1)}, \cdots, \hat{\mathbf{P}}^{(N)}$ in a similar way. So far, two attention layers learn the unique life stage representations for users and products but fail to capture the evolutionary feature during the learning process.

3.2 Inter-Life-Stage Representation Learning:

The evolutionary patterns contain essential information on 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. Formally, we can compute the storage coefficient z and the forget coefficient g as:

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

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

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

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

3.3 Framework Training with Masked Loss Function

To predict the rating, we concatenate the representations of two linked nodes, and one fully connected layer is used as the predictor. The prediction can be represented as:

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

where the $\mathbf{W}_{pred} \in \mathbb{R}^{2H}$ and $\mathbf{b}_{pred} \in \mathbb{R}$ 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, in reality, the recommender system usually suffers from severe data sparsity. To alleviate such a problem, we can make use of the products not rated by users. In our case, we assume that the products not purchased by users can still attract users to some extent. Here, we utilize a hyper-parameter α to control the extent that not purchased products attract users. If $\alpha = 0$, then users dislike those not purchased products (*i.e.*, the corresponding rating is 0). Now, the loss function for the SEGNN is

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

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

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

After calculating the loss, we adopt the backpropagation algorithm to update all the variables. The general procedure of SEGNN is shown in Algorithm 1.

Algorithm 1: SEGNN Model

```
Result: The life stages representations for users
        $\mathcal{X}_u = \{\mathbf{U}^{(1)}, \cdots, \mathbf{U}^{(M)}\}$ and products
        $\mathcal{X}_p = \{\mathbf{P}^{(1)}, \cdots, \mathbf{P}^{(N)}\}$
Input: Random Initialization for $\mathbf{U}^{(0)}$ and $\mathbf{P}^{(0)}$ and set $\mathcal{X}_u$
       and $\mathcal{X}_p$ zero; the dynamic rating matrices
       $\mathbf{R}^{(11)}, \cdots, \mathbf{R}^{(MN)}$
1  while $i < Epochs$ do
2      for user stage $m$ do
3          $\mathbf{U}^{(m)} = \text{GSAU}\Big(\hat{\mathbf{U}}^{(m-1)}, \mathbf{U}^{(m)}, \big(\mathbf{P}^{(1)}, \cdots, \mathbf{P}^{(N)}\big),$
           $\big(\mathbf{R}^{(m1)}, \cdots, \mathbf{R}^{(mN)}\big)\Big)$;
4      end
5      for item stage $n$ do
6          $\mathbf{P}^{(n)} = \text{GSAU}\Big(\hat{\mathbf{P}}^{(n-1)}, \mathbf{P}^{(n)}, \big(\mathbf{U}^{(1)}, \cdots, \mathbf{U}^{(M)}\big),$
           $\big(\mathbf{R}^{(1n)}, \cdots, \mathbf{R}^{(Mn)}\big)\Big)$;
7      end
8      Compute $\mathcal{L}$ with updated $\mathcal{X}_u$ and $\mathcal{X}_p$;
9      Update the model variables via backpropagation;
10     $i = i + 1$;
11 end
```

4 EXPERIMENTS

In the experiments, we use two publicly accessible datasets, ML100k and Amazon Musical Instruments (AMI). The information of two datasets is summarized in Table 1. To show the effectiveness of SEGNN, we use two traditional models including ItemKNN [14], MF (Matrix Factorization) [9], and a life cycle-based model BiCycle [12], a neural network-based model NeuMF [7], two graph-based models including SpectralCF [22], GAT [15] as the comparison methods.

Table 1: Statistics of the datasets.

Dataset	Users	products	Ratings	Density
ML 100K [4]	943	1,682	100,000	6.30%
AMI [5]	1,429	900	10,261	0.80%

4.1 Experimental Setting And Evaluation Metrics

4.1.1 Experimental Setting. For both datasets, we sort them according to timestamps and then divide each dataset into 29 time windows [12] with equal depth. We select time window $1 < c \leq k$ as train windows. Then, we obtain 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 $lr = 0.01$. We set the parameter to evaluate the significances of unobserved interactions between users and products as $\alpha = 0.8$. For parameters in our models, we utilize Glorot initialization [3].

4.1.2 Evaluation Metrics. We adopt three commonly used metrics to evaluate the performance. They are Recall of Top k (Rec@k), Precision of Top k (Prec@k), and Mean Reciprocal Rank (MRR).

4.2 Experimental Results

In the experiments, we use the last ten windows as testing windows, and the rest are the training windows. Due to the space limit, we only present the results of three windows out of ten due to space limitations. The experimental results are shown in Table 2, 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. The traditional baselines lack interpretability. Additionally, traditional baselines fail to take temporal factors into consideration. Thus, it is hard for them to capture the temporal continuity of the graph. Prec@15, Recall@15, and MRR in Table 2 indicate that our proposed model performs better than these traditional baselines.

Then, we compare our proposed model 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 consider the different importances of products. The results support our intuition of better interprets of deep models. Both SpectralCF and NeuMF adopt neural networks. SpectralCF learns the features via spectral domain, while NeuMF uses both 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

Table 2: Prec@15(rank), Rec@15(rank) and MRR(rank). The best performer is in boldface.

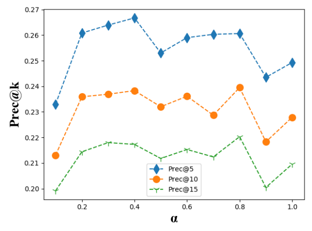
		Window 1			Window 2			Window 3			Ave.
Dataset	Method	Prec@15	Recall@15	MRR	Prec@15	Recall@15	MRR	Prec@15	Recall@15	MRR	Rank
ML100K	SEGNN	**0.1692(1)**	**0.1029(1)**	**0.0390(1)**	0.1683(2)	0.0965(2)	0.0303(4)	0.1775(2)	**0.1113(1)**	**0.0361(1)**	**1.6**
	GAT	0.1652(2)	0.0986(2)	0.0332(2)	**0.1758(1)**	**0.1206(1)**	**0.0454(1)**	0.1756(3)	0.0975(2)	0.0289(4)	2
	SpectralCF	0.0106(7)	0.0101(7)	0.0222(4)	0.0124(7)	0.0134(7)	0.0114(6)	0.0128(7)	0.0101(7)	0.0084(7)	6.5
	NeuMF	0.1234(4)	0.0468(4)	0.0190(5)	0.1042(4)	0.0666(4)	0.0309(3)	0.1408(5)	0.0477(5)	0.0231(5)	4.3
	Bicycle	0.1552(3)	0.0663(3)	0.0258(3)	0.1608(3)	0.0853(3)	0.0367(2)	0.1465(4)	0.0631(4)	0.0293(3)	3.1
	MF	0.1045(5)	0.0366(5)	0.0140(6)	0.0842(5)	0.0522(5)	0.0221(5)	**0.1850(1)**	0.0870(3)	0.0312(2)	4.1
	ItemKNN	0.0448(6)	0.0169(6)	0.0097(7)	0.0383(6)	0.0155(6)	0.0073(7)	0.0451(6)	0.0219(6)	0.0097(6)	6.2
AMI	SEGNN	**0.0152(1)**	**0.0809(1)**	**0.0303(1)**	**0.0271(1)**	**0.1335(1)**	**0.0478(1)**	**0.0196(1)**	0.1074(2)	**0.0382(1)**	**1.1**
	GAT	**0.0152(1)**	0.0718(4)	0.0285(4)	0.0248(2)	0.1201(2)	0.0474(2)	**0.0196(1)**	**0.1077(1)**	0.0381(2)	2.1
	SpectralCF	0.0016(6)	0.0084(6)	0.0040(7)	0.0027(6)	0.0126(7)	0.0117(5)	0.0025(7)	0.0135(7)	0.0074(7)	6.4
	NeuMF	0.0012(7)	0.0034(7)	0.0052(6)	0.0024(7)	0.0215(6)	0.0058(7)	0.0047(6)	0.0364(6)	0.0085(6)	6.4
	Bicycle	0.0145(3)	0.0721(2)	0.0289(2)	0.0236(3)	0.1082(3)	0.0462(3)	0.0181(3)	0.0863(3)	0.0370(3)	2.7
	MF	0.0145(3)	0.0721(2)	0.0289(2)	0.0236(3)	0.1082(3)	0.0462(3)	0.0181(3)	0.0863(3)	0.0370(3)	2.7
	ItemKNN	0.0067(5)	0.0457(5)	0.0182(5)	0.0041(5)	0.0354(5)	0.0115(6)	0.0072(5)	0.0523(5)	0.0230(5)	5.1

our model. This result indicates that temporal dynamics and significances of unobserved interactions can promote the performance of GAT. SEGNN also introduces the attention mechanism. However, we utilize neighbor features and apply them to capture temporal dynamics among different life stages of users and products.

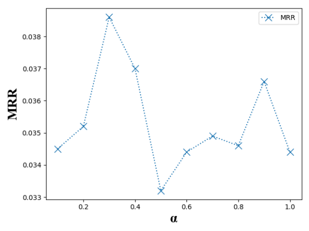
We should also note that BiCycle and MF perform well on the specific test windows, though our methods still outperform them in general. It is also worth mentioning that two traditional methods outperform our proposed model on some metrics. The gaps between these two traditional methods and our proposed methods are small.

4.3 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 the dataset ML100K via using model SEGNN. Figure 2 shows the result. Many works like [7, 9, 12] 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 2, we note that $\alpha = 1$ is not the best performer. It is not proper to treat observed and unobserved equally. Our model can achieve higher results by carefully choosing α.



(a) Prec@k on the dataset ML100K



(b) MRR@k on the dataset ML100K

Figure 2: Prec@k, Rec@k and MRR of SEGNN on the Dataset ML100K with Different α

4.4 Ablation Study

Table 3: Improvement of Proposed Methods with Comparing with The Previous Version

Methods	ML100K			AMI		
	Prec@15	MRR	Imp.	Prec@15	MRR	Imp.
P-SENN	0.1709	0.0329	N/A	0.0197	0.0318	N/A
SENN	0.1738	0.0318	-0.8232%	0.0197	0.0379	9.591%
SEGNN	0.1776	0.0365	8.482%	0.0206	0.0388	3.471%

In this section, we study the effectiveness of the life-stage attention layer and the memory mechanism for capturing the evolutionary patterns added. We adopt two variants P-SENN without life stage attention layer and memory mechanism and SENN only without memory mechanism. Averaged Prec@15 and MRR are selected as metrics. We also give *Improve* to illustrate the improvement of our proposed methods with extra components compared with the previous version. The definition of *Improve* is as follows:

$$Imp. = \frac{1}{2} \cdot (IP + IM)$$

where IP and IM denote the improvement on Prec@15 and MRR, respectively. Table 3 shows that the life-stage attention layer can improve the performance a lot for AMI and have little impact on ML100K. It also shows that the memory mechanism does have a positive impact on both datasets.

5 RELATED WORK

The related work of our proposed model contains two parts: dynamic and neural network-based recommender system and graph-based recommender system.

Dynamic and Neural Network-based Recommender System: Matrix factorization-based methods have been proposed for dynamic recommender systems for a long time. TimeSVD [18] is the very first method to introduce temporal information to fulfill the drawbacks of traditional collaborative filtering methods. The bayesian factorization model [19] is a better model with introducing temporal dynamics. Temporal dynamics are essential factors for recommender systems. For utilizing temporal factors better, BiCycle [12] 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. Other neural network-based methods are also proposed for recommender systems. Neural Collaborative Filtering (NeuMF) is the collaborative signal by multi-layer perceptron. SpectralCF [22] is the first method directly learning from the spectral domain of user-product bipartite graphs. [17] introduces a convolutional neural network and deep belief network to assist representation learning.

Graph-based Recommender System: To learn latent factors of users and products from graphs, several researchers have proposed graph-based recommender systems. In Graph Neural Networks, Graph Attention Networks (GATs) recently attract many researchers' interests. There are a series of GATs like [1, 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.

6 CONCLUSION

In this paper, we propose a novel model SEGNN incorporating life cycles for the dynamic recommendation problem. SEGNN learns the representations of users and products by the GSAU first. 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 alleviate the data sparsity issue, we take advantage of the mask matrix to decide the degree of the utilization of not rated products. The experimental results and analysis shows the effectiveness of the proposed model.

REFERENCES

[1] Sami Abu-El-Haija, Bryan Perozzi, Rami Al-Rfou, and Alex Alemi. 2018. Watch Your Step: Learning Node Embeddings via Graph Attention. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems* (Montr&#233;al, Canada) *(NIPS'18)*. Curran Associates Inc., USA, 9198–9208. http://dl.acm.org/citation.cfm?id=3327546.3327592

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

[3] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research)*, Yee Whye Teh and Mike Titterington (Eds.), Vol. 9. PMLR, Chia Laguna Resort, Sardinia, Italy, 249–256. http://proceedings.mlr.press/v9/glorot10a.html

[4] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* 5, 4, Article 19 (Dec. 2015), 19 pages. https://doi.org/10.1145/2827872

[5] Ruining He and Julian McAuley. 2016. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. In *Proceedings of the 25th International Conference on World Wide Web* (Montr&#233;al, Qu&#233;bec, Canada) *(WWW '16)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 507–517. https://doi.org/10.1145/2872427.2883037

[6] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 355–364.

[7] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web* (Perth, Australia) *(WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 173–182. https://doi.org/10.1145/3038912.3052569

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

[9] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (Aug. 2009), 30–37. https://doi.org/10.1109/MC.2009.263

[10] John Boaz Lee, Ryan Rossi, and Xiangnan Kong. 2018. Graph Classification Using Structural Attention. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery &#38; Data Mining* (London, United Kingdom) *(KDD '18)*. ACM, New York, NY, USA, 1666–1674. https://doi.org/10.1145/3219819.3219980

[11] Xin Liu. 2015. Modeling users' dynamic preference for personalized recommendation. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

[12] X. Liu, Y. Song, C. Aggarwal, Y. Zhang, and X. Kong. 2017. BiCycle: Item Recommendation with Life Cycles. In *2017 IEEE International Conference on Data Mining (ICDM)*. 297–306. https://doi.org/10.1109/ICDM.2017.39

[13] Makbule Gulcin Ozsoy, Faruk Polat, and Reda Alhajj. 2016. Time Preference Aware Dynamic Recommendation Enhanced with Location, Social Network and Temporal Information. In *Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining* (Davis, California) *(ASONAM '16)*. IEEE Press, Piscataway, NJ, USA, 909–916. http://dl.acm.org/citation.cfm?id=3192424.3192595

[14] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th International Conference on World Wide Web* (Hong Kong, Hong Kong) *(WWW '01)*. ACM, New York, NY, USA, 285–295. https://doi.org/10.1145/371920.372071

[15] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rJXMpikCZ

[16] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *Proceedings of the 42Nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Paris, France) *(SIGIR'19)*. ACM, New York, NY, USA, 165–174. https://doi.org/10.1145/3331184.3331267

[17] Xinxi Wang and Ye Wang. 2014. Improving Content-based and Hybrid Music Recommendation Using Deep Learning. In *Proceedings of the 22Nd ACM International Conference on Multimedia* (Orlando, Florida, USA) *(MM '14)*. ACM, New York, NY, USA, 627–636. https://doi.org/10.1145/2647868.2654940

[18] Liang Xiang and Qing Yang. 2009. 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)*. IEEE Computer Society, Washington, DC, USA, 450–457. https://doi.org/10.1109/WI-IAT.2009.78

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

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

[21] Yizhou Zhang, Yun Xiong, Xiangnan Kong, and Yangyong Zhu. 2016. NetCycle: Collective Evolution Inference in Heterogeneous Information Networks. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. ACM, New York, NY, USA, 1365–1374. https://doi.org/10.1145/2939672.2939742

[22] Lei Zheng, Chun-Ta Lu, Fei Jiang, Jiawei Zhang, and Philip S. Yu. 2018. Spectral Collaborative Filtering. In *Proceedings of the 12th ACM Conference on Recommender Systems* (Vancouver, British Columbia, Canada) *(RecSys '18)*. ACM, New York, NY, USA, 311–319. https://doi.org/10.1145/3240323.3240343

[23] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1059–1068.