AutoSeqRec: Autoencoder for Efficient Sequential Recommendation

Sijia Liu[*][†]
Jiahao Liu[*][†]
sijialiu21@m.fudan.edu.cn
jiahaoliu21@m.fudan.edu.cn
Fudan University
Shanghai, China

Hansu Gu
Seattle, United States
hansug@acm.org

Dongsheng Li
Microsoft Research Asia
Shanghai, China
dongsli@microsoft.com

Tun Lu[‡][†]
Fudan University
Shanghai, China
lutun@fudan.edu.cn

Peng Zhang[†]
Fudan University
Shanghai, China
zhangpeng_@fudan.edu.cn

Ning Gu[†]
Fudan University
Shanghai, China
ninggu@fudan.edu.cn

ABSTRACT

Sequential recommendation demonstrates the capability to recommend items by modeling the sequential behavior of users. Traditional methods typically treat users as sequences of items, overlooking the collaborative relationships among them. Graph-based methods incorporate collaborative information by utilizing the user-item interaction graph. However, these methods sometimes face challenges in terms of time complexity and computational efficiency. To address these limitations, this paper presents AutoSeqRec, an incremental recommendation model specifically designed for sequential recommendation tasks. AutoSeqRec is based on autoencoders and consists of an encoder and three decoders within the autoencoder architecture. These components consider both the user-item interaction matrix and the rows and columns of the item transition matrix. The reconstruction of the user-item interaction matrix captures user long-term preferences through collaborative filtering. In addition, the rows and columns of the item transition matrix represent the item out-degree and in-degree hopping behavior, which allows for modeling the user's short-term interests. When making incremental recommendations, only the input matrices need to be updated, without the need to update parameters, which makes AutoSeqRec very efficient. Comprehensive evaluations demonstrate that AutoSeqRec outperforms existing methods in terms of accuracy, while showcasing its robustness and efficiency.

CCS CONCEPTS

• **Information systems → Collaborative filtering**.

[*]Both authors contributed equally to this research.
[†]Also affiliated to Shanghai Key Laboratory of Data Science, China, and Shanghai Institute of Intelligent Electronics & Systems, China.
[‡]Corresponding author.

KEYWORDS

sequential recommendation, autoencoder, collaborative filtering

ACM Reference Format:
Sijia Liu, Jiahao Liu, Hansu Gu, Dongsheng Li, Tun Lu, Peng Zhang, and Ning Gu. 2023. AutoSeqRec: Autoencoder for Efficient Sequential Recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3583780.3614788

1 INTRODUCTION

Recommender systems have the ability to model user preferences and help users discover content of their interest from a vast amount of information [8, 13, 21, 23, 24, 27, 33]. Meanwhile, as user interests evolve over time, recommender systems are required to dynamically update and promptly adapt their recommendation strategies to keep up with the changes, capturing shifts in user interests in real time [11, 14, 20, 22, 36, 39]. However, training a recommendation model from scratch incurs significant costs, making it impractical to retrain the model every time there are interaction updates. Therefore, the sequential recommendation model faces a critical challenge of efficiently capturing evolving user interests. To address this challenge, two key problems need to be solved: 1) accurately capturing the evolution of user interests, and 2) swiftly adapting to changes through incremental updates.

Existing sequential recommendation models suffer from limitations in both aspects. On the one hand, traditional sequential methods [1, 14, 34] typically treat users as a sequences of items they have interacted with, overlooking the collaborative relationships among them. Consequently, these methods fail to effectively leverage collaborative information, which is crucial for improving accuracy and personalization in recommendations. On the other hand, while some more recent graph-based approaches [3, 39] can incorporate collaborative information by leveraging user-item interaction graphs, these methods often face challenges in terms of time complexity and computational efficiency. To address these limitations, we propose a novel recommendation model called AutoSeqRec (**Auto**encoder for Efficient **Seq**uential **Rec**ommendation), which consists of the following two components.

To accurately capture the evolution of user interests, we propose a multi-encoder and decoder architecture. The first step involves constructing a user-item interaction matrix and an item transition matrix. These two matrices are then fed into a multi-information encoder, while three single-information decoders are employed for reconstruction purposes. The user-item interaction matrix contains the historical interactions between users and items. By reconstructing this matrix, AutoSeqRec can effectively model collaborative information, providing a representation of users' long-term preferences. On the other hand, the item transition matrix captures the typical previous/next item that users interact with before/after the current item. Through reconstructing this matrix, AutoSeqRec can capture users' real-time interactions, enabling a representation of their short-term interests. By combining collaborative information and item transition information, AutoSeqRec is able to accurately capture the evolution of user interests.

To promptly adapt to changes through incremental updates, we introduce an efficient incremental inference process that incorporates new user-item interactions. This process begins by representing the single-hop item transition probability based on the reconstructed item transition matrix. It then efficiently approximates the multi-hop transition information by utilizing the compressed output of the multi-information encoder. When there are new updates on interactions, AutoSeqRec only modifies the user-item interaction matrix and item transition matrix incrementally, while keeping the model parameters unchanged for inference. This approach enables AutoSeqRec to efficiently adapt to new changes at the incremental level of a single interaction, facilitating real-time recommendation updates with minimal computational overhead. The update process of AutoSeqRec shares some insights with a contemporaneous work, IMCrrorect [19]. However, unlike IMCrrorect, which focuses on recommendation unlearning tasks, AutoSeqRec focuses on incremental recommendation tasks.

Our main contributions are as follows:

- We design a novel autoencoder architecture called AutoSeqRec. By combining the user's long-term preference derived from collaborative information with the user's short-term interest modeled from item transition, our approach allows for a more accurate capture of user interest evolution and changes in behavior.
- We develop an efficient incremental inference process that incorporates new user-item interactions without changing model parameters. It facilitates real-time recommendation updates at the incremental level of a single interaction.
- Comprehensive evaluation on four datasets shows that AutoSeqRec can significantly outperform existing sequential recommendation methods in accuracy and efficiency. Moreover, AutoSeqRec can be integrated with other sequential recommendation methods to improve their performance.

2 PRELIMINARIES

In this section, we introduce AutoRec [25], a representative autoencoder framework for collaborative filtering. Then, we describe the problem and discuss the limitations of the previous autoencoder-based method in solving this problem.

2.1 Notations

In this paper, we denote user set as $\mathcal{U} = \{u_1, u_2 ..., u_m\}$ and item set as $\mathcal{I} = \{i_1, i_2, ..., i_n\}$, where m is the number of users and n is the number of items. Without loss of generality, we use u to represent a user and i to represent an item when their indices are not concerned. The user-item rating matrix $Y \in \mathbb{R}^{m \times n}$ is partially observed. Each user is represented by a partially observed vector $y^{(u)} = (Y_{u,1}, ..., Y_{u,n}) \in \mathbb{R}^n$, and each item is represented by a partially observed vector $y^{(i)} = (Y_{1,i}, ..., Y_{m,i}) \in \mathbb{R}^m$.

2.2 AutoRec

AutoRec [25] designs an item-based (or user-based) autoencoder which first takes as input each partially observed $y^{(i)}$ (or $y^{(u)}$), then projects it into a low-dimensional latent (hidden) space, and finally reconstruct $y^{(i)}$ ($y^{(u)}$) in the output space to predict missing ratings for static recommendation.

Taking item-based AutoRec as an example, which tries to solve the following reconstruction problem:

$$\min_{\theta} \sum_{\mathbf{y}^{(i)}} \|\mathbf{y}^{(i)} - h(\mathbf{y}^{(\mathrm{i})}; \theta)\|^2, \tag{1}$$

where $h(\cdot) = f(g(\cdot))$ includes two processes: encoding and decoding. The encoder $g(\cdot)$ and decoder $f(\cdot)$ are both composed of a single nonlinear layer. Then, the parameters θ are learned using backpropagation.

Given learned parameters $\hat{\theta}$, the predicted rating of item-based AutoRec for user u and item i is

$$\hat{Y}_{u,i} = \left(h\left(\mathbf{y}^{(i)}; \hat{\theta}\right)\right)_u. \tag{2}$$

2.3 Problem description

Each user-item interaction can be represented by a tuple $(u^{(\tau)}, i^{(\tau)})$, where $\tau = 1, 2, ...,$ indicates the sequential order in which the interaction occurred. Suppose there are N interactions in total, denoted as $\mathcal{S}^{(N)} =< (u^{(1)}, i^{(1)}), (u^{(2)}, i^{(2)}), ..., (u^{(N)}, i^{(N)}) >$. Now given a user u, we need to predict which item that user u will interact with in the $N + 1$-th interaction. The output should be a n-dimensional vector, and each dimension represents the possibility score of interaction between user u and the corresponding item.

AutoRec is a static recommendation method that cannot effectively utilize sequential information. Although a lot of extensions have been proposed [16, 18, 27, 31, 35, 37], there is no prior work to extend AutoRec to incremental recommendation scenarios to the best of our knowledge.

3 THE PROPOSED METHOD

In this section, we first introduce the overall architecture of AutoSeqRec, and then present the implementation of incremental inference of AutoSeqRec.

3.1 Architecture

In this subsection, we detail the AutoSeqRec, including its implementation of the learning process and inference process. The overall architecture of AutoSeqRec is shown in Figure 1.

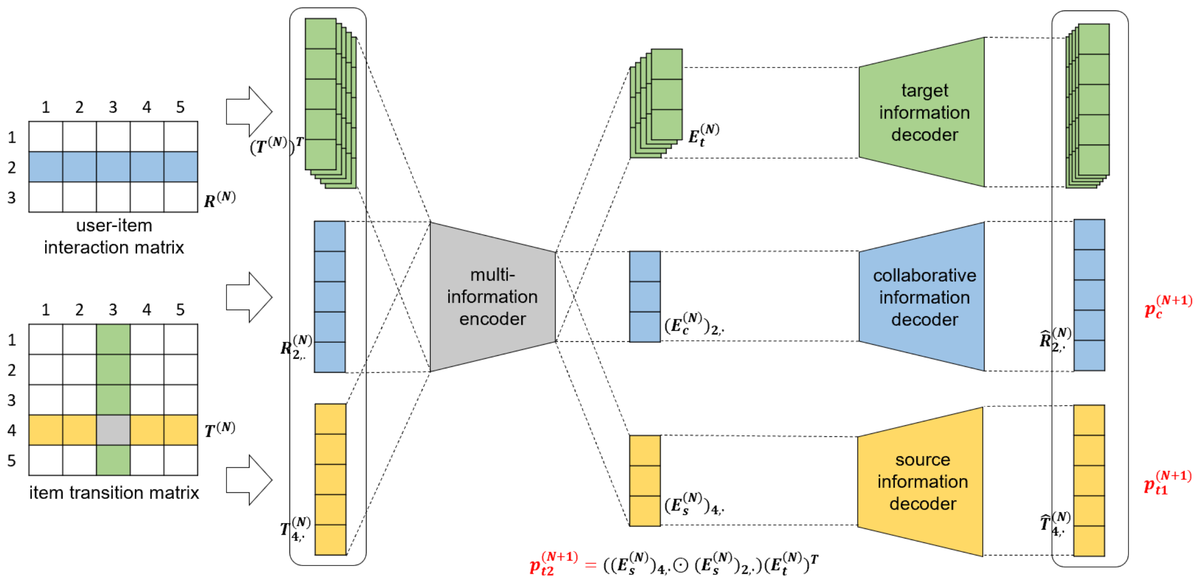
1 2 3 4 5
1
2
3
$R^{(N)}$
user-item interaction matrix
1 2 3 4 5
1
2
3
4
5
$T^{(N)}$
item transition matrix
$(T^{(N)})^T$
$R^{(N)}_{2,\cdot}$
$T^{(N)}_{4,\cdot}$
multi-information encoder
$E^{(N)}_t$
$(E^{(N)}_c)_{2,\cdot}$
$(E^{(N)}_s)_{4,\cdot}$
target information decoder
collaborative information decoder
source information decoder
$\hat{R}^{(N)}_{2,\cdot}$
$\hat{T}^{(N)}_{4,\cdot}$
$p^{(N+1)}_c$
$p^{(N+1)}_{t1}$
$p^{(N+1)}_{t2} = ((E^{(N)}_s)_{4,\cdot} \odot (E^{(N)}_s)_{2,\cdot})(E^{(N)}_t)^T$

Figure 1: The inference process of AutoSeqRec. Assuming that user 2 last interacted with item 4, this figure shows how AutoSeqRec obtains the prediction score of user 2 interacting each item in the $N+1$-th interaction.

3.1.1 Learning Process. We first construct the user-item interaction matrix $R^{(N)} \in \mathbb{R}^{m\times n}$ and item transition matrix $T^{(N)} \in \mathbb{R}^{n\times n}$, where the superscript indicates that there are a total of N interactions. The user-item interaction matrix $R^{(N)}$ stores the interaction records between users and items in $\mathcal{S}^{(N)}$, its entry at u-th row and i-th column is defined as:

$$R^{(N)}_{u,i} = \begin{cases} 1, & \text{user } u \text{ has interacted with item } i \text{ in } \mathcal{S}^{(N)}, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

We define the interaction sequence of user u in $\mathcal{S}^{(N)}$ as $\mathcal{S}^{(N)}_u =< i^{(1)}_u, i^{(2)}_u, ..., i^{(n^{(N)}_u)}_u >$, where $i^{(\tau)}_u$ is the item id of the τ-th interaction of user u, $n^{(N)}_u$ is the number of interactions of user u in $\mathcal{S}^{(N)}$. The item transition matrix $T^{(N)}$ stores the transition information between item pairs, its entry at i'-th row and i''-th column is defined as:

$$T^{(N)}_{i',i''} = \sum_{u=1,...,m} \sum_{\substack{i^{(\tau)}_u = i', i^{(\tau+1)}_u = i'' \\ \tau=1,...,n^{(N)}_u - 1}} 1. \tag{4}$$

Discussion about the input matrices. Each row of the user-item interaction matrix $R^{(N)}$ indicates which items the corresponding user has interacted with, which actually describes a user's personalized preference. We call the elements related to $R^{(N)}$ **collaborative information**. Each row of the item transition matrix $T^{(N)}$ records the data of which item a user will interact with next time statistically, after the user interacts with the corresponding source item. Since it is statistical data from all users, this information is irrelevant to a specific user. We call the elements related to $T^{(N)}$ **source information**. Correspondingly, each row of the transposition of the item transition matrix, $(T^{(N)})^\top$, records the data of which item the user interacted with last time if a user interacts with the corresponding target item. This is also statistical data from all users and does not contain personalized information. We call the elements related to $(T^{(N)})^\top$ **target information**.

In this paper, we implement the multi-information encoder (denoted as $f(\cdot)$) with a single nonlinear layer, more complex forms of the multi-information encoder are left for future work:

$$f(X) = \sigma(XW_e + \mathbf{b}_e), \tag{5}$$

where $X \in \mathbb{R}^{*\times n}$ is an input matrix with an unlimited number of rows, $W_e \in \mathbb{R}^{n\times k}$ is the weight matrix, $\mathbf{b}_e \in \mathbb{R}^k$ is the bias term, and $\sigma(\cdot)$ is the activation function. The user-item interaction matrix $R^{(N)}$, item transition matrix $T^{(N)}$ and its transpose $(T^{(N)})^\top$ are input into the multi-information encoder to obtain collaborative information embedding $E^{(N)}_c$, source information embedding $E^{(N)}_s$, and target information embedding $E^{(N)}_t$, respectively:

$$E^{(N)}_c = f(R^{(N)}) \in \mathbb{R}^{m\times k}, \tag{6}$$

$$E^{(N)}_s = f(T^{(N)}) \in \mathbb{R}^{n\times k}, \tag{7}$$

$$E^{(N)}_t = f((T^{(N)})^\top) \in \mathbb{R}^{n\times k}. \tag{8}$$

Discussion about multi-information encoder. Firstly, the user-item interaction matrix, item transition matrix, and its transpose share consistent semantics. Each dimension represents a corresponding item, allowing them to be simultaneously compressed by an encoder. Secondly, the multi-information encoder converts the input matrices into embedding representations with semantic meaning. This implies that if two users have similar interaction histories, i.e., their respective rows in the user-item interaction matrix are similar, their outputs after passing through the encoder will also exhibit similarity. Likewise, if the transition patterns of two items (whether as source or target items) are comparable, indicating similar rows or columns in the item transition matrix, their outputs after going through the multi-information encoder will reflect this similarity. Lastly, since the multi-information encoder incorporates collaborative information, source information, and target information simultaneously, the compressed representation generated by the multi-information encoder can effectively capture and consider all three types of information concurrently.

Next, we input collaborative information embedding $E_c^{(N)}$, source information embedding $E_s^{(N)}$, target information embedding $E_t^{(N)}$ into collaborative information decoder (denoted as $g_c(\cdot)$), source information decoder (denoted as $g_s(\cdot)$) and target information decoder (denoted as $g_t(\cdot)$) respectively, to reconstruct the user-item interaction matrix $R^{(N)}$, the item transition matrix $T^{(N)}$ and its transpose $(T^{(N)})^\top$:

$$\tilde{R}^{(N)} = g_c(E_c^{(N)}) \in \mathbb{R}^{m\times n}, \tag{9}$$

$$\tilde{T}^{(N)} = g_s(E_s^{(N)}) \in \mathbb{R}^{n\times n}, \tag{10}$$

$$(\tilde{T}^{(N)})^\top = g_t(E_t^{(N)}) \in \mathbb{R}^{n\times n}. \tag{11}$$

Again, we only adopt the simplest setting, where each of the three decoders consists of a single nonlinear layer:

$$g_c(X) = \sigma(XW_c + \mathbf{b}_c), \tag{12}$$

$$g_s(X) = \sigma(XW_s + \mathbf{b}_s), \tag{13}$$

$$g_t(X) = \sigma(XW_t + \mathbf{b}_t), \tag{14}$$

where $W_c, W_s, W_t \in \mathbb{R}^{k\times n}$ are the weight matrices, $\mathbf{b}_c, \mathbf{b}_s, \mathbf{b}_t \in \mathbb{R}^n$ are the bias terms.

Finally, we train AutoSeqRec by reducing the reconstruction loss, using the Frobenius norm as the measure of the differences between the input matrices and the reconstructed matrices. Specifically, we denote the reconstruction loss of the collaborative information decoder as $\mathcal{L}_c$, denote the reconstruction loss of the source information decoder as $\mathcal{L}_s$, and denote the reconstruction loss of the target information decoder as $\mathcal{L}_t$, which are defined as follows, respectively:

$$\mathcal{L}_c = ||R^{(N)} - \tilde{R}^{(N)}||_F, \tag{15}$$

$$\mathcal{L}_s = ||T^{(N)} - \tilde{T}^{(N)}||_F, \tag{16}$$

$$\mathcal{L}_t = ||(T^{(N)})^\top - (\tilde{T}^{(N)})^\top||_F. \tag{17}$$

By minimizing the reconstruction loss, we aim to enable the decoders to recover the original data as accurately as possible. The final loss function consists of three kinds of losses:

$$\mathcal{L} = \mathcal{L}_c + \mathcal{L}_t + \mathcal{L}_s. \tag{18}$$

Through backpropagation during training, we iteratively update model parameters to minimize reconstruction loss. Since the reconstructions of three types of information are equally important, we do not need to set any hyperparameters to balance among them.

3.1.2 Inference Process. As aforementioned, the user-item interaction matrix reflects user personalized information. In static scenarios, we can directly use the reconstructed user-item interaction matrix $\tilde{R}^{(N)}$ as the prediction result. The collaborative information-based probability score of user u interacting with all items in the

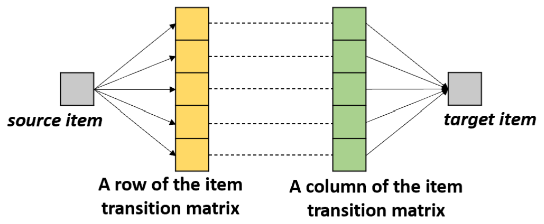
source item
target item
A row of the item transition matrix
A column of the item transition matrix

Figure 2: Explanation of two-hops transition. A row of the item transition matrix records the data of which item a user will interact with next time, after the user interacts with the source item. A column of the item transition matrix records the data of which item the user interacted with last time if a user interacts with the target item. The inner product of the two vectors represents the probability score of the source item reaching the target item after two hops.

$N + 1$-th interaction can be expressed as:

$$\mathbf{p}_c^{(N+1)} = \tilde{R}_{u,\cdot}^{(N)} \in \mathbb{R}^n. \tag{19}$$

Each dimension of $\mathbf{p}_c$ corresponds to an item.

However, in dynamic scenarios, the reconstructed interaction matrix $\tilde{R}^{(N)}$ can only reflect users' long-term preferences and cannot accurately reflect users' temporal interests. To capture users' short-term interests, we further utilize item transition matrix. For user u, whose last interacted item is $i_u^{(n_u^{(N)})}$ (for the convenience of narration, we use i to refer to $i_u^{(n_u^{(N)})}$), we take the i-th row of the reconstructed item transition matrix as the short-term interest of user u. Then the single-hop transition probability score of user u is:

$$\mathbf{p}_{t1}^{(N+1)} = \tilde{T}_{i,\cdot}^{(N)} \in \mathbb{R}^n. \tag{20}$$

Each dimension of $\mathbf{p}_{t1}$ corresponds to an item.

$\mathbf{p}_{t1}^{(N+1)}$ only considers the information of single-hop transition. In order to more comprehensively consider the information of multi-hop transitions, a simple solution is to multiply $\mathbf{p}_{t1}^{(N+1)}$ and the item transition matrix $\tilde{T}^{(N)}$ continuously. As shown in the Figure 2, $\mathbf{p}_{t1}^{(N+1)}$ reflects the probability score of the items that will be reached from a source item, and each column of the item transition matrix reflects the probability score of the items that a target item may come from. So one multiplication can represent the probability score of reaching the target item from the source item after two hops. However, the time complexity of this approach can be very high when the number of items is large.

We took a more efficient approach. As mentioned before, similar inputs will have similar outputs after passing through the multi-information encoder, which in turn will have similar decoder outputs. So instead of operating on the reconstruction matrix output by the decoder, we can operate on the embedded representation output by the multi-information encoder. Specifically, we perform a Hadamard product operation on the collaborative information embedding of user u ($(E_c^{(N)})_{u,\cdot}$) and $(E_s^{(N)})_{i,\cdot}$ to obtain a personalized transition embedding:

$$\tilde{\mathbf{p}}_{t2}^{(N+1)} = (E_s^{(N)})_{i,\cdot} \odot (E_c^{(N)})_{u,\cdot} \tag{21}$$

Furthermore, we multiply the $\tilde{\mathbf{p}}_{t2}^{(N+1)}$ by the target information embedding $E_t^{(N)}$ to obtain the two-hop transition information:

$$\mathbf{p}_{t2}^{(N+1)} = \tilde{\mathbf{p}}_{t2}^{(N+1)} (E_t^{(N)})^\top \in \mathbb{R}^n. \tag{22}$$

The multi-hop transition information can be obtained by multiplying the transpose of the target information embedding, $(E_t^{(N)})^\top$, multiple times.

Finally, we take the weighted sum of three types of probability scores (collaborative information probability score $\mathbf{p}_c^{(N+1)}$, one-hop transition probability score $\mathbf{p}_{t1}^{(N+1)}$, and two-hops transition probability score $\mathbf{p}_{t2}^{(N+1)}$) as the final prediction result, as follows:

$$\mathbf{p}^{(N+1)} = \lambda_1 \mathbf{p}_c^{(N+1)} + \lambda_2 \mathbf{p}_{t1}^{(N+1)} + (1 - \lambda_1 - \lambda_2)\mathbf{p}_{t2}^{(N+1)}. \tag{23}$$

The coefficients λ_1 and λ_2 control the importance of different sources of information in the final prediction results.

3.2 Incremental Inference

In this subsection, we introduce how AutoSeqRec effectively handles dynamic data and updates recommendation results in real time. Specifically, we first analyze why the autoencoder-based CF model can naturally achieve incremental recommendation, and then we give the concrete implementation of incremental recommendation of AutoSeqRec.

3.2.1 Analysis. Representation learning-based recommendation methods, e.g., matrix factorization [13], first obtain the embeddings of users and items, and then perform personalized ranking of items by calculating the similarity (e.g., inner product) between users and items. Different from these representation learning-based methods, the autoencoder-based collaborative filtering method is a process of obtaining predicted interactions directly from observed interactions. For example, AutoRec directly inputs the user-item interaction matrix into the autoencoder, and takes the reconstructed user-item interaction matrix as the prediction result. Autoencoder-based methods are often used to solve the problem of static recommendation.

Actually, once an autoencoder is trained, it is equipped to handle various user interests. Now, let us focus on the dynamic recommendation process for two users a and b. Assume that user a likes to watch comedy movies in the early days, and user b likes to watch science fiction movies during the same period. Because the historical interaction records of the two users are different (a interacts with more comedy movies, b interacts with more science fiction movies), when a recommendation model based on autoencoder is built, a will be recommended comedy movies, b will be recommended science fiction movies. Both user a and user b can obtain the desired personalized recommendation in that period, and the root cause of their different recommendation results is the difference in historical interaction (the difference between the corresponding rows of the interaction matrix). If in the later stage, a's interest has changed to science fiction movies. At this time, the interaction record of a (interacting more science fiction movies instead of comedy movies) is input into the recommendation model, and a will be recommended science fiction movies. The autoencoder-based model has this ability because it has learned how to recommend science fiction movies to users (e.g., user b) from the training data. To sum up, the collaborative filtering method based on autoencoder actually supports incremental recommendation naturally. We only need to change the input matrix, and the weight of autoencoder does not need to be updated during the incremental inference process.

Table 1: The statistics of the datasets used in the experiments.

	# Users	# Items	# Interactions	# Unique Times
ML-100K	943	1,349	99,287	49,119
ML-1M	6,040	3,416	999,661	458,254
Garden	1,686	962	13,272	1,888
Video	5,130	1,685	37,126	1,946

3.2.2 Implementation. We believe that AutoSeqRec has the ability to handle various user interests and encode the historical interactions of users at different moments. Therefore, we only need to change the input user-item interaction matrix R and item transition matrix T, and then follow the same inference process to make predictions. Specifically, assuming user u interacts with item i at $N+1$-th interaction, we can use the following steps to predict the next interaction: 1) update the user-item interaction matrix $R^{(N+1)}$ and item transition matrix $T^{(N+1)}$ to contain the latest interaction state between user u and item i. 2) input the updated user-item interaction matrix and item transition matrix into the AutoSeqRec to obtain the recommendation result. After the model training is completed, we can make accurate next-step interaction predictions based on the existing interaction history. In addition, we can execute the above prediction process as many times as needed to achieve continuous prediction of the user's next interaction item. This flexibility and efficiency enable AutoSeqRec to quickly respond to user needs and provide personalized and high-quality recommendation services in practical applications.

4 EXPERIMENTS

In this section, we conduct comprehensive evaluation to answer the following research questions (RQs):

- **RQ1** Does AutoSeqRec outperform state-of-the-art methods in future interaction prediction task on dynamic graphs?
- **RQ2** How do different components of AutoSeqRec affect model performance?
- **RQ3** Does the desgin of AutoSeqRec help to reduce running time?
- **RQ4** Does the hidden embedding size and coefficients λ_1, λ_2 affect predictions?
- **RQ5** Can AutoSeqRec be seamlessly integrated with other sequential recommendation models to improve their performance?

4.1 Experimental Setup

4.1.1 Dataset. We use four publicly available datasets to evaluate the performance of AutoSeqRec. MovieLens [6] is a movie rating dataset that is widely used in dynamic graph tasks. We use two versions of MovieLens: ML-100K and ML-1M. ML-100K contains 100K interactions collected during a seven-month period from September 19th, 1997 to April 22nd, 1998. ML-1M contains 1M interactions from users who joined MovieLens in 2000. Amazon [7] contains ratings on the e-commerce platform Amazon.com, and we adopt two sub-datasets, Garden and Video. We follow the same preprocessing pipeline in previous works [10, 28] that users and items with fewer than 5 interactions are discarded. The statistical information of datasets is shown in Table 1.

4.1.2 Setting. We run all the experiments on a server equipped with one NVIDIA TESLA T4 GPU and Intel(R) Xeon(R) Gold 5218R CPU. All the code of this work is implemented with Python 3.7.4 [1]. Three input matrices including user-item interaction matrix, item transition matrix and the transposition of item transition matrix are first divided into mini-batches separately, and then we union these mini-batches and randomly shuffle them. In the training process, we use the corresponding loss function according to which matrix is used in the mini-batches union. The hyper-parameters are tuned via grid search. More specifically, we search autoencoder's hidden dimension size in [32, 64, 128, 256] and coefficients λ_1, λ_2 from values in $[0.0, 0.1, 0.2, \cdots, 1.0]$.

4.1.3 Baseline Models. We compare AutoSeqRec with the following state-of-the-art interaction graph modelling methods:

- LightGCN [8] is a representative collaborative filtering algorithm based on GNNs, which ignores time information.
- RRN [34] models user and item interaction sequences with separate RNNs for sequential recommendation.
- JODIE [14] also generates node embeddings using RNNs, and can also estimate user embeddings trajectories.
- CoPE [39] uses an ordinary differential equation-based GNN to model the evolution of network.
- FreeGEM [20] is a state-of-the-art parameter-free method for dynamic graph learning tasks.

4.2 Future interaction prediction (RQ1)

The future prediction prediction task is defined as follows: given all interactions till time t, we predict which item will user u interact with at time t? We use the same data splitting as JODIE's [14] and CoPE's [39] to separate the dataset by time. Specifically, we use the earliest 80% interactions for training, the following 10% data for validation, and the latest 10% data for test. We adopt mean reciprocal rank (MRR) and Recall@10 to evaluate the performance of future interaction prediction, in which MRR is the average of the reciprocal rank and Recall@10 is the fraction of interactions in which the ground truth items are within the top 10 recommendations, following JODIE, CoPE and FreeGEM.

Table 2 shows the results of AutoSeqRec compared with the above five state-of-the-art methods. We observe that AutoSeqRec significantly outperforms all compared methods in all datasets across both metrics Recall@10 and MRR on the four datasets. We calculate the relative improvement (Rel. Imp.) of AutoSeqRec over the baseline as (performance of AutoSeqRec – performance of baseline)/(performance of baseline). Across all datasets, the minimum improvement of AutoSeqRec is 2.00% - 50.00% in terms of MRR and 10.62% - 52.94% in terms of Recall@10. We notice that LightGCN performs the worst on most datasets. This is largely due to the fact that we split the dataset by time according to this experiment setting. Since AutoSeqRec makes full use of user-item collaborative information and one-hop or even two-hops of item transition information, AutoSeqRec outperforms the other methods.

[1]The codes are available at https://github.com/sliu675/AutoSeqRec

4.3 Ablation studies (RQ2)

In this section, we first conduct ablation study on the operations applied to the three types of information from the encoder. Then, we test whether the decoder outputs of user-item interaction matrix and item transition matrix are helpful or not.

4.3.1 Encoder hidden outputs operations. This experiment test the effectiveness of 1) the Hadamard product operation on the collaborative information embedding and source information embedding and 2) the inner product operation between the personalized transition embedding and the target information embedding. We use a, b, c to represent the collaborative information embedding of user $u((E_c)_{u,\cdot})$, u's last interacted item i's source information embedding $((E_s)_{i,\cdot})$ and the target information embedding E_t. We observe that combining a, b and c obtain the best performance among all the combinations as shown in Table 3. Moreover, operation $a \cdot c$ performs better than $b \cdot c$ on Amazon datasets, while the opposite is true on MovieLens dataset. One possible reason is that the two Amazon datasets used in our experiments are more sparse than the MovieLens datasets, which may lead to less robust statistics in the item transition matrix. Therefore, the collaborative information from the user-item interaction matrix becomes more important.

4.3.2 Decoder outputs of interaction matrix and transition matrix. We test if adding the decoder outputs of user-item interaction matrix and item transition matrix can improve the performance or not, compared with only using encoder hidden outputs. We add the reconstructed interaction matrix and transition matrix separately, and then add these two reconstructed matrices together. Specifically, we use a weighted sum to combine all the information (hidden outputs, reconstructed interaction matrix and transition matrix). As shown in Table 4, we observe that using all information is better than any other combination, confirming the usefulness of collaborative information and multi-hops transition information.

4.4 Efficiency Comparison (RQ3)

We use the future interaction prediction task to study the efficiency of JODIE, CoPE, FreeGEM and AutoSeqRec. Other baselines like RRN show comparable efficiency with JODIE [14] and thus are omitted. FreeGEM requires multiple runs of offline/online SVD, JODIE and CoPE both run 50 epochs and choose the best performing model on the validation set as the optimal model.

As shown in Table 5, AutoSeqRec is at least 16.3x faster than JODIE, at least 64.7x faster than CoPE, and at least 1.1x faster than the parameters-free method FreeGEM. This experiment confirms that AutoSeqRec has higher computational efficiency than the compared methods, maily due to 1) the interaction matrix and transition matrix updates are incremental and 2) the encoder-decoder architecture in AutoSeqRec is much more light-weight compared with RNNs (in JODIE) and GNNs (in CoPE).

Table 2: Accuracy comparison on the future interaction prediction task.

Method	ML-100K		ML-1M		Video		Garden	
	MRR	Recall@10	MRR	Recall@10	MRR	Recall@10	MRR	Recall@10
LightGCN	0.012	0.025	0.014	0.031	0.019	0.036	0.025	0.087
RRN	0.042	0.084	0.038	0.071	0.039	0.074	0.079	0.169
JODIE	0.046	0.095	0.037	0.070	0.044	0.081	0.059	0.142
CoPE	0.049	0.111	0.040	0.085	0.036	0.078	0.082	0.176
FreeGEM	0.040	0.079	0.040	0.077	0.058	0.086	0.100	0.179
AutoSeqRec	**0.068**	**0.150**	**0.060**	**0.130**	**0.064**	**0.107**	**0.102**	**0.227**
Rel. Imp.	38.78%	35.14%	50.00%	52.94%	10.34%	10.62%	2.00%	26.82%

Table 3: Ablation study on the encoder hidden outputs operations.

Setting	ML-100K		ML-1M		Video		Garden	
	MRR	Recall@10	MRR	Recall@10	MRR	Recall@10	MRR	Recall@10
$b \cdot c$	0.050	0.113	0.031	0.064	0.016	0.026	0.058	0.123
$a \cdot c$	0.038	0.079	0.020	0.037	0.030	0.060	0.086	0.203
$(a \odot b) \cdot c$	**0.058**	**0.130**	**0.037**	**0.076**	**0.040**	**0.086**	**0.097**	**0.206**

Table 4: Ablation study on the decoder outputs of user-item interaction matrix and item transition matrix.

Setting	ML-100K		ML-1M		Video		Garden	
	MRR	Recall@10	MRR	Recall@10	MRR	Recall@10	MRR	Recall@10
only hidden	0.058	0.130	0.037	0.076	0.040	0.086	0.097	0.206
hidden + interaction matrix	0.062	0.138	0.040	0.081	0.038	0.078	0.091	0.215
hidden + transition matrix	0.060	0.132	0.058	0.126	0.062	0.106	0.101	0.226
all	**0.068**	**0.150**	**0.060**	**0.130**	**0.064**	**0.107**	**0.102**	**0.227**

Table 5: Running time comparison on the four datasets.

	ML-100K	ML-1M	Video	Garden
JODIE	8h47min56s (298.8x)	374h7min53s (1989.5x)	46min18s (38.6x)	16min51s (16.3x)
CoPE	16h51min5s (572.3x)	1025h24min6s (5452.7x)	1h38min33s (82.1x)	1h6min52s (64.7x)
FreeGEM	3min12s (1.8x)	5h54min18s (31.4x)	2min46s (2.3x)	1min7s (1.1x)
AutoSeqRec	1min46s	11min17s	1min12s	1min2s

4.5 Sensitivity Analysis (RQ4)

Here, we analyze how the key hyper-parameters affect the performance of AutoSeqRec. We analyze the following hyper-parameters: the embedding size of the auto-encoder and the coefficients λ_1 and λ_2 for balancing the importance of different prediction scores in Equation 23.

4.5.1 *Embedding Size.*

We analyze the impact of the embedding size in the autoencoder on the performance. To do this, we vary the embedding size of the autoencoder from 32 to 256 and calculate the MRR and Recall@10 scores for future interaction prediction on the Amazon Garden dataset and MovieLens ML-100K dataset. The trends on other datasets are similar. The results are shown in Figure 3 (a). As the embedding dimension increases from 32 to 128, we find that the embedding size has a relatively positive effect on the performance of AutoSeqRec, i.e., the performance of AutoSeqRec consistently improve with increasing dimension size. This experiment demonstrates that AutoSeqRec does not suffer from the common overfitting issue as other deep models.

4.5.2 *The Coefficients λ_1 and λ_2.*

λ_1 controls the importance of collaborative information and λ_2 controls the importance of one-hop transition. The coefficients of collaborative information, one-hop transition and two-hops transition add up to one. Thus, once these two coefficients λ_1 and λ_2 are determined, the third one is immediately determined as well. We observe that using a single type of information is not optimal. To make sure each source of information is used, we vary the coefficients from 0.1 to 0.8 and plot the heatmaps of Recall@10 on the ML-100K and Video datasets. As shown in Figure 3 (b) and Figure 3 (c), the performance of AutoSeqRec is not very sensitive to λ_1 and λ_2, i.e., we can observe near optimal performance with several different configurations.

4.6 Integrating AutoSeqRec with Other Sequential Recommendation Methods (RQ5)

The major innovations in AutoSeqRec, e.g., the embedding learned from multi-information encoder, the score functions (Equation 21 and 22), and the combination of collaborative and transition information, are orthogonal to many existing sequential recommendation methods. Therefore, we believe AutoSeqRec can be integrated

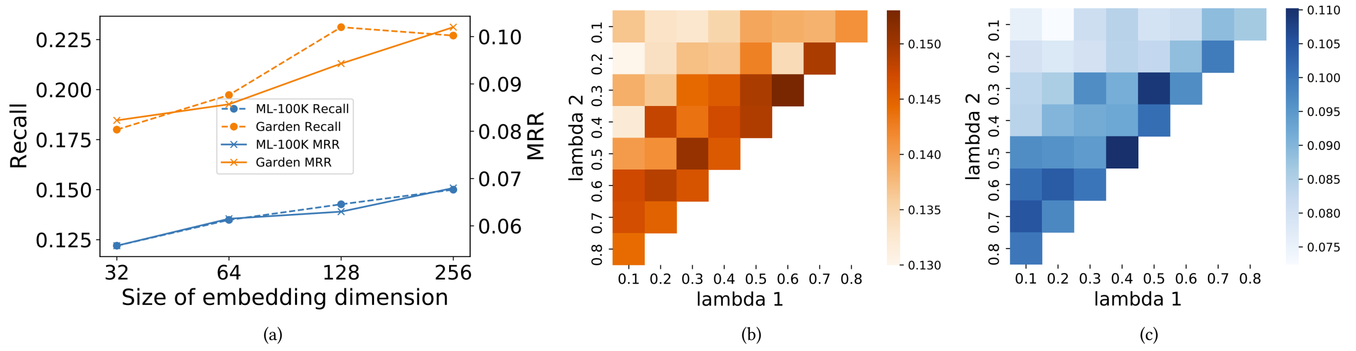

Figure 3: Sensitivity analysis of AutoSeqRec on three hyperparameters: (a) MRR and Recall@10 with different embedding size on ML-100K and Garden, (b) the heat-map of Recall@10 with different coefficients λ_1 and λ_2 on ML-100K, (c) the heat-map of Recall@10 with different coefficients λ_1 and λ_2 on Video.

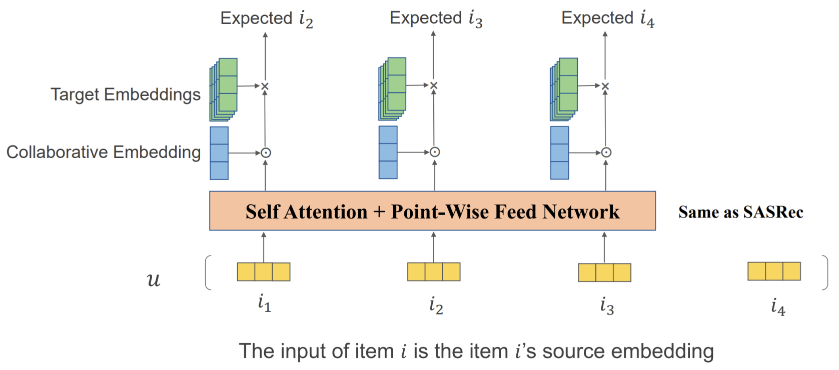

Figure 4: The process of integrating AutoSeqRec with SASRec. Collaborative information embedding, source information embedding and target information embedding are pre-trained by AutoSeqRec.

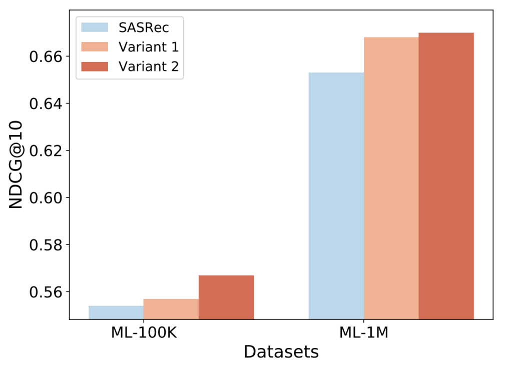

Figure 5: The recommendation accuracy of integrating AutoSeqRec with SASRec on the two MovieLens datasets.

with them to further improve their performance. Here, we take SASRec [10], a well-known sequential recommendation method, as an example to illustrate how to integrate AutoSeqRec with other sequential recommendation methods as shown in Figure 4.

4.6.1 Setting. We use the sequential recommendation task defined in SASRec [10]. All the data pre-processing and partitioning protocols are the same as SASRec: 1) we treat the presence of a review or rating as implicit feedback, 2) we use timestamps to determine the sequential order of actions, and 3) we split the historical sequence of each user into three parts: the most recent one interaction for test, the penultimate interaction for validation, and all remaining interactions for training.

4.6.2 The Process of Integrating AutoSeqRec with SASRec. SASRec [10] is a sequential recommendation method based on the Transformer architecture [32]. We first briefly introduce its architecture. Firstly, SASRec reads in the embedding sequence of items that a user has interacted with. Then, after several Attention blocks (including self-attention layer and point-wise feed forward), we get the item embedding sequence after feature transformation. Finally, the item embedding sequence after feature transformation will be input into the prediction layer (e.g., multi-layer perceptron), which will treat each item as a class, and train the model by optimizing the classification task.

While maintaining the main architecture (the Attention blocks) of SASRec, we use the collaborative information embedding E_c, source information embedding E_s and target information embedding E_t, which are outputs from multi-information encoder, to improve the input and prediction layers of SASRec. Firstly, we will use the rows corresponding to source information embedding E_s as input to SASRec, instead of the randomly initialized embedding. Then, after passing through several Attention blocks, we will obtain the source information embedding after feature transformation. Finally, in the prediction phase, we adopt a consistent approach with AutoSeqRec. Specifically, as Equation (21), we first perform a Hadamard product between E_c and the source information embedding after feature transformation to obtain the personalized transition embedding, and then multiply personalized transition embedding with the target information embedding E_t as Equation (22) to further consider multi-hops transition information.

Similar to the ablation studies, we design two ablative variants for this study to further understand the effectiveness of our design.

Specifically, we call that the model use collaborative information embedding as Variant 2, and that without collaborative information embedding as Variant 1.

4.6.3 Analysis. We compare Variant 1 and Variant 2 with SASRec on two MovieLens datasets (ML-1M and ML-100k). We follow the evaluation strategy in prior works [9, 12] to avoid heavy computation on all user-item pairs. For each user u, we randomly sample 50 negative items, and rank these items with the ground-truth item. The hidden dimension we use in our variants is 64. We use NDCG@10, which can simultaneously evaluate the ranking and recall of a prediction.

For more clear comparison, we plot the performance of SASRec and the two variants on the bar chart as shown in Figure 5. We can observe that both variants are better than SASRec. The improvement of Variant 1 is 0.54%-2.30% in terms of NDCG@10 and the improvement of Variant 2 is 2.35%-2.60% in terms of NDCG@10. The experimental results indicate that multiplying the i-th row of the source information embedding by the target information embedding in Equation 22 is helpful.

Furthermore, compared with Variant 1, Variant 2 achieves relatively higher accuracy, i.e., higher NDCG@10 on both datasets. In other words, adding the operation, a Hadamard product on the collaborative information embedding of user u, is not only beneficial to the future interaction prediction task, but also beneficial to the sequential recommendation task defined in SASRec. Thus, we further validate the rationality of multiplying the i-th row of the source information embedding by the target information embedding, and operating a Hadamard product on the collaborative information embedding of user u.

From this case study, we can further conclude that 1) the embedding learned by AutoSeqRec are effective for sequential recommendation, 2) our score functions are helpful for improving accuracy, and 3) it is beneficial to combine collaborative information with transition information for sequential recommendation tasks.

5 RELATED WORK

In this section, we first introduce the sequential recommendation methods, including recurrent neural network based methods, temporal point process based methods and graph based methods. Then, we introduce the autoencoder-based recommendation methods.

5.1 Sequential Recommendation

Many sequential recommendation methods are based on recurrent neural networks. RRN [34] models user and item interaction sequences with separate RNNs. Time-LSTM [41] adopts time gates to represent the time intervals. LatentCross [1] incorporates contextual data into embeddings. DeepCoevolve [4] and JODIE [14] generate node embeddings using two intertwined RNNs, and JODIE can also estimate user embeddings trajectories.

Another line of sequential recommendation work is based on temporal point processes. Know-Evolve [30] and HTNE [42] model interactions between users and items as multivariate point processes and Hawkes processes, respectively. Shchur et al. [26] directly model the conditional distribution of inter-event times, and therefore more accurately control the changes in user interests. DSPP [2] incorporates topology and long-term dependencies into the intensity function. All these methods take time information into account by intensity functions.

Recently, more advancing graph based sequential recommenders are proposed. TDIG-MPNN [3] captures both global and local information on the graph. SDGNN [29] uses the state changes of neighbor nodes to learn the node embedding. CoPE [39] uses an ordinary differential equation-based GNN to model the evolution of network.

5.2 Autoencoder-based Collaborative Filtering

Sedhain et al. [25] proposed AutoRec, the first autoencoder framework for collaborative filtering. Many researchers extend autoencoders to collaborative filtering for implicit feedback. Liang et al. [18] extended variational autoencoders to collaborative filtering for implicit feedback. Based on the Autorec and Transformer architectures, Dang et al. [5] proposed an attentional autoencoder for implicit recommendation. Zhu et al. [40] proposed a mutually-regularized dual collaborative variational auto-encoder (MD-CVAE) for recommendation.

Moreover, some works incorporate content information to enhance the performance of autoencoder. Li et al. [17] proposed a model that learns a shared feature space from heterogeneous data, leveraging heterogeneous auxiliary information to address the data sparsity problem of recommender systems. To solve the cold start problem, Lee et al. [15] developed a hybrid CF technique incorporating CF with content information. For implicit trust relationships, Zeng et al. [38] proposed an autoencoder model based on implicit trust relationship between users, combining the intrinsic relationship of both the implicit trust information and user-item interaction behavior for collaborative recommendation.

6 CONCLUSIONS

Efficiently capturing changing interests in real-time recommendation is the key challenge faced by sequential recommendation methods. This paper proposes AutoSeqRec, which is an autoencoder-based incremental method for sequential recommendation. AutoSeqRec consists of an encoder and three decoders within an autoencoder architecture. The encoder takes as input the rows of the user-item interaction matrix, the rows of the item transition matrix, and the columns of the item transition matrix. By simultaneously capturing different types of information, the encoder generates informative representations of user interests and item transitions. The three decoders reconstruct these three types of information, enabling a comprehensive understanding of user preferences and short-term interests. We conduct comprehensive experiments to verify the superiority of AutoSeqRec, and the experimental results show that AutoSeqRec can outperform the state-of-the-art methods in accuracy while achieving significant improvement in efficiency. In the future, we will try more complex encoder and decoder architectures, such as Transformers [10].

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (NSFC) under Grants 62172106 and 61932007.

REFERENCES

[1] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H Chi. 2018. Latent cross: Making use of context in recurrent recommender systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 46–54.

[2] Jiangxia Cao, Xixun Lin, Xin Cong, Shu Guo, Hengzhu Tang, Tingwen Liu, and Bin Wang. 2021. Deep structural point process for learning temporal interaction networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 305–320.

[3] Xiaofu Chang, Xuqin Liu, Jianfeng Wen, Shuang Li, Yanming Fang, Le Song, and Yuan Qi. 2020. Continuous-time dynamic graph learning via neural interaction processes. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 145–154.

[4] Hanjun Dai, Yichen Wang, Rakshit Trivedi, and Le Song. 2016. Deep coevolutionary network: Embedding user and item features for recommendation. *arXiv preprint arXiv:1609.03675* (2016).

[5] Hoang-Vu Dang and Dung Ngo. 2019. Attentional autoencoder for weighted implicit collaborative filtering. In *Proceedings of the 2019 2nd International Conference on Computational Intelligence and Intelligent Systems*. 168–172.

[6] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5, 4 (2015), 1–19.

[7] 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*. 507–517.

[8] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.

[9] 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*. 173–182.

[10] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 197–206.

[11] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. 2020. Representation Learning for Dynamic Graphs: A Survey. *J. Mach. Learn. Res.* 21, 70 (2020), 1–73.

[12] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 426–434.

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

[14] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 1269–1278.

[15] Kiwon Lee, Hyeonsoo Jo, Hyoji Kim, and Yong H Lee. 2019. Basis learning autoencoders for hybrid collaborative filtering in cold start setting. In *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 1–6.

[16] Sheng Li, Jaya Kawale, and Yun Fu. 2015. Deep collaborative filtering via marginalized denoising auto-encoder. In *Proceedings of the 24th ACM international on conference on information and knowledge management*. 811–820.

[17] Tianyu Li, Yukun Ma, Jiu Xu, Björn Stenger, Chen Liu, and Yu Hirate. 2018. Deep heterogeneous autoencoders for collaborative filtering. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1164–1169.

[18] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 world wide web conference*. 689–698.

[19] Jiahao Liu, Dongsheng Li, Hansu Gu, Tun Lu, Jiongran Wu, Peng Zhang, Li Shang, and Ning Gu. 2023. Recommendation Unlearning via Matrix Correction. *arXiv preprint arXiv:2307.15960* (2023).

[20] Jiahao Liu, Dongsheng Li, Hansu Gu, Tun Lu, Peng Zhang, and Ning Gu. 2022. Parameter-free Dynamic Graph Embedding for Link Prediction. *Advances in Neural Information Processing Systems* 35 (2022), 27623–27635.

[21] Jiahao Liu, Dongsheng Li, Hansu Gu, Tun Lu, Peng Zhang, Li Shang, and Ning Gu. 2023. Personalized Graph Signal Processing for Collaborative Filtering. In *Proceedings of the ACM Web Conference 2023*. 1264–1272.

[22] Jiahao Liu, Dongsheng Li, Hansu Gu, Tun Lu, Peng Zhang, Li Shang, and Ning Gu. 2023. Triple Structural Information Modelling for Accurate, Explainable and Interactive Recommendation. *arXiv preprint arXiv:2304.11528* (2023).

[23] Xia Ning and George Karypis. 2011. Slim: Sparse linear methods for top-n recommender systems. In *2011 IEEE 11th international conference on data mining*. IEEE, 497–506.

[24] 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*. 285–295.

[25] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th international conference on World Wide Web*. 111–112.

[26] Oleksandr Shchur, Marin Biloš, and Stephan Günnemann. 2019. Intensity-free learning of temporal point processes. *arXiv preprint arXiv:1909.12127* (2019).

[27] Harald Steck. 2019. Embarrassingly shallow autoencoders for sparse data. In *The World Wide Web Conference*. 3251–3257.

[28] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM international conference on web search and data mining*. 565–573.

[29] Sheng Tian, Tao Xiong, and Leilei Shi. 2021. Streaming Dynamic Graph Neural Networks for Continuous-Time Temporal Graph Modeling. In *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1361–1366.

[30] Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. 2017. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In *international conference on machine learning*. PMLR, 3462–3471.

[31] Vojtěch Vančura, Rodrigo Alves, Petr Kasalický, and Pavel Kordík. 2022. Scalable Linear Shallow Autoencoder for Collaborative Filtering. In *Proceedings of the 16th ACM Conference on Recommender Systems*. 604–609.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[33] 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*. 165–174.

[34] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent recommender networks. In *Proceedings of the tenth ACM international conference on web search and data mining*. 495–503.

[35] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the ninth ACM international conference on web search and data mining*. 153–162.

[36] Jiafeng Xia, Dongsheng Li, Hansu Gu, Jiahao Liu, Tun Lu, and Ning Gu. 2022. FIRE: Fast incremental recommendation with graph signal processing. In *Proceedings of the ACM Web Conference 2022*. 2360–2369.

[37] Feng Yuan, Lina Yao, and Boualem Benatallah. 2019. Adversarial collaborative auto-encoder for top-n recommendation. In *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

[38] Wei Zeng, Jiwei Qin, and Chunting Wei. 2021. Neural Collaborative Autoencoder for Recommendation With Co-Occurrence Embedding. *IEEE Access* 9 (2021), 163316–163324.

[39] Yao Zhang, Yun Xiong, Dongsheng Li, Caihua Shan, Kan Ren, and Yangyong Zhu. 2021. CoPE: Modeling Continuous Propagation and Evolution on Interaction Graph. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2627–2636.

[40] Yaochen Zhu and Zhenzhong Chen. 2022. Mutually-regularized dual collaborative variational auto-encoder for recommendation systems. In *Proceedings of the ACM Web Conference 2022*. 2379–2387.

[41] Yu Zhu, Hao Li, Yikang Liao, Beidou Wang, Ziyu Guan, Haifeng Liu, and Deng Cai. 2017. What to Do Next: Modeling User Behaviors by Time-LSTM.. In *IJCAI*, Vol. 17. 3602–3608.

[42] Yuan Zuo, Guannan Liu, Hao Lin, Jia Guo, Xiaoqian Hu, and Junjie Wu. 2018. Embedding temporal network via neighborhood formation. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2857–2866.