Sequential Recommendation with Self-Attentive Multi-Adversarial Network

Ruiyang Ren[1,4], Zhaoyang Liu[2], Yaliang Li[2], Wayne Xin Zhao[3,4*], Hui Wang[1,4], Bolin Ding[2], and Ji-Rong Wen[3,4]
[1]School of Information, Renmin University of China
[2]Alibaba Group
[3]Gaoling School of Artificial Intelligence, Renmin University of China
[4]Beijing Key Laboratory of Big Data Management and Analysis Methods
{reyon.ren, hui.wang, jrwen}@ruc.edu.cn, {jingmu.lzy, yaliang.li, bolin.ding}@alibaba-inc.com, batmanfly@gmail.com

ABSTRACT

Recently, deep learning has made significant progress in the task of sequential recommendation. Existing neural sequential recommenders typically adopt a generative way trained with Maximum Likelihood Estimation (MLE). When context information (called *factor*) is involved, it is difficult to analyze when and how each individual factor would affect the final recommendation performance.

For this purpose, we take a new perspective and introduce adversarial learning to sequential recommendation. In this paper, we present a Multi-Factor Generative Adversarial Network (MFGAN) for explicitly modeling the effect of context information on sequential recommendation. Specifically, our proposed MFGAN has two kinds of modules: a Transformer-based generator taking user behavior sequences as input to recommend the possible next items, and multiple factor-specific discriminators to evaluate the generated sub-sequence from the perspectives of different factors. To learn the parameters, we adopt the classic policy gradient method, and utilize the reward signal of discriminators for guiding the learning of the generator. Our framework is flexible to incorporate multiple kinds of factor information, and is able to trace how each factor contributes to the recommendation decision over time. Extensive experiments conducted on three real-world datasets demonstrate the superiority of our proposed model over the state-of-the-art methods, in terms of effectiveness and interpretability.

CCS CONCEPTS

• **Information systems → Recommender systems**; • **Computing methodologies → Neural networks**.

KEYWORDS

Sequential Recommendation, Adversarial Training, Self-Attentive Mechanism

*Corresponding author.

SIGIR ’20, July 25–30, 2020, Virtual Event, China
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401111

ACM Reference Format:
Ruiyang Ren, Zhaoyang Liu, Yaliang Li, Wayne Xin Zhao, Hui Wang, Bolin Ding and Ji-Rong Wen. 2020. Sequential Recommendation with Self-Attentive Multi-Adversarial Network. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR ’20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397271.3401111

1 INTRODUCTION

Recommender systems aim to accurately characterize user interests and provide personalized recommendations in a variety of real-world applications. They serve as an important information filtering technique to alleviate the information overload problem and enhance user experiences. In most applications, users’ interests are dynamic and evolving over time. It is essential to capture the dynamics of sequential user behaviors for making appropriate recommendations.

In the literature, various methods [10, 14, 26] have been proposed for sequential recommender systems. Early methods usually utilize the Markov assumption that the current behavior is tightly related to the previous ones [26]. Recently, sequential neural networks such as recurrent neural network [4] and Transformer [27] have been applied to recommendation tasks as these networks can characterize sequential user-item interactions and learn effective representations of user behaviors [10, 14]. Besides, several studies have proposed to incorporate context information to enhance the performance of neural sequential recommenders [11, 12, 16]. The advantages of these sequential neural networks have been experimentally confirmed as they have achieved significant performance improvements.

Typically, existing neural sequential recommenders [10, 14] adopt a generative way to predict future items and learn the parameters using Maximum Likelihood Estimation (MLE). However, it has been found that MLE-based training is easy to suffer from issues such as data sparsity or exposure bias [23, 32] in sequence prediction. Especially, in such an approach, when context information (called as *factor* in this paper) is incorporated, it has to be integrated with the original sequential prediction component [11, 12, 16]. The consequence is that various factors (*e.g.,* price and brand of a product in the e-commerce scenario) are either mixed in the sequential context representations, or coupled with the black-box recommendation module. Therefore, we cannot accurately figure out when and how each individual factor would affect the final recommendation performance. These disadvantages weaken or even impede their applications in a wide range of decision making scenarios. It is important to explicitly and effectively characterize the effect of various factors in sequential recommender systems.

In the light of this challenge, we propose to use an adversarial training approach to developing sequential recommender systems. Indeed, the potential advantage of Generative Adversarial Network (GAN) has been shown in collaborative filtering methods [2, 28]. Different from prior studies, our novelty is to decouple *factor utilization* from the *sequence prediction* component via adversarial training. Following the GAN framework [7], we set two different components, namely generator and discriminator. In our framework, the generator predicts the future items for recommendation relying on user-item interaction data alone, while the discriminator judges the rationality of the generated recommendation sequence based on available information of various factors. Such an approach allows more flexibility in utilizing external context information in sequential recommendation, which is able to improve the recommendation interpretability.

To this end, in this paper, we present a novel *Multi-Factor Generative Adversarial Network (MFGAN)*. Specifically, our proposed MFGAN has two essential kinds of modules: (1) a Transformer-based generator taking user behavior sequences as input to recommend the possible next items, and (2) multiple factor-specific discriminators to evaluate the generated recommendations from the perspectives of different factors. Unlike the generator, the discriminator adopts a bi-directional Transformer-based architecture, and it can refer to the information of subsequent positions for sequence evaluation. In this way, the discriminator is expected to make more reliable judgement by considering the overall sequential characteristics *w.r.t.* different factors. Due to the discrete nature of item generation, the training of the proposed MFGAN method is realized in a reinforcement learning way by policy gradient. The key point is that we utilize the discriminator modules to provide the reward signal for guiding the learning of the generator.

Under our framework, various factors are decoupled from the generator, and they are utilized by the discriminators to derive supervision signals to improve the generator. To validate the effectiveness of the proposed MFGAN, we conduct extensive experiments on three real-world datasets from different domains. Experimental results show that the proposed MFGAN is able to achieve better performance compared to several competitive methods. We further show the multi-adversarial architecture is indeed useful to stabilize the learning process of our approach. Finally, qualitative analysis demonstrates that the proposed MFGAN can explicitly characterize the effect of various factors over time for sequential recommendation, making the recommendation results highly interpretable.

Our main contributions are summarized as follows:

• To the best of our knowledge, we are the first to introduce adversarial training into the sequential recommendation task, and design the unidirectional generator for prediction and bidirectional discriminator for evaluation.

• We propose a multi-discriminator structure that can decouple different factors and improve the performance of sequential recommendation. We analyze the effectiveness and the stability of the multi-adversarial architecture in our task.

• Extensive experiments conducted on three real-world datasets demonstrate the benefits of the proposed MFGAN over state-of-the-art methods, in terms of both effectiveness and interpretability.

2 PROBLEM DEFINITION

In this section, we first formulate the studied problem of sequential recommendation before diving into the details of the proposed method. Let $\mathcal{U}$ and $\mathcal{I}$ denote a set of users and items, respectively, where $|\mathcal{U}|$ and $|\mathcal{I}|$ are the numbers of users or items. Typically, a user u has a chronologically-ordered interaction sequence of items: $\{i_1, i_2, \dots, i_t, \dots, i_n\}$, where n is the total number of interactions and i_t is the t-th item that user u has interacted with. For convenience, we use $i_{j:k}$ to denote the subsequence of the entire sequence, *i.e.,* $i_{j:k} = \{i_j, \dots, i_k\}$, where $1 \leq j < k \leq n$. Besides, we assume that each item i is associated with m kinds of contextual information, corresponding to m factors, *e.g.,* artist, album and popularity in music recommender system.

Based on the above notations, we now define the task of sequential recommendation. Formally, given the historical behaviors of a user (*i.e.,* $\{i_1, i_2, \dots, i_t, \dots, i_n\}$) and the context information of items, our task aims to predict the next item that she/he is likely to interact with at the $(n+1)$-th step.

3 METHODOLOGY

In this section, we first give an overview of the proposed *Multi-Factor Generative Adversarial Network* (MFGAN) framework, and then introduce the design of the generator and discriminators. The details of the training process are also discussed in this section.

3.1 Multi-Factor Generative Adversarial Network Framework

Figure 1 presents the overview of our proposed MFGAN framework for sequential recommendation.

3.1.1 Basic Components. In this framework, we have two kinds of components undertaking different responsibilities for sequential recommendation:

(1) The upper component is the prediction component (*i.e.* generator G) which is a sequential recommendation model and successively generates the next items based on the current historical sequence. Note that the generator will not use any context information from the item side. It only makes the prediction conditioned on historical sequence data.

(2) The lower component is the evaluation component that is a set of m discriminators $\{D_1, D_2, \dots, D_m\}$ for judging the rationality of generated sequences by using the information from multiple perspectives. Each discriminator performs the judgement from a certain perspective based on the information of some corresponding factor. For example, in music recommender system, we may have multiple discriminators specially designed with category information, popularity statistics, artist and album of music, respectively.

3.1.2 Overall Procedure. Following standard GAN [7], the generator and multiple discriminators will play a min-max game. At the t-step, the generator first generates a predicted item $\hat{i}_t$ based on the historical sequence $\{i_1, \dots, i_{t-1}\}$. Then, each discriminator takes the t-length sequence $\{i_1, \dots, i_{t-1}, \hat{i}_t\}$ as the input and evaluates

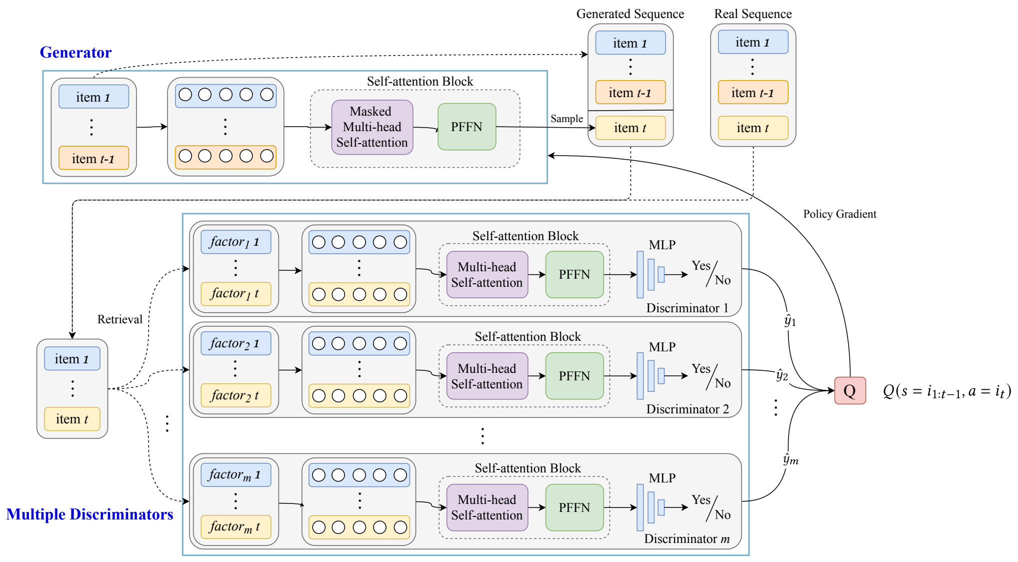


Figure 1: The overview of the proposed MFGAN model consisting of a generator and multiple discriminators. The upper and the bottom framed parts correspond to the generator and multi-discriminator components, respectively.

the rationality of the generated sequence using the information of some factor. The evaluation results are sent back to the generator to guide the learning of the generator at the next round. Correspondingly, the discriminator is updated by taking the generated sequence and actual sequence (*i.e.,* ground-truth user behaviors) as the training samples for improving its discriminative capacity. As such, the two components force each other to improve in a mutual reinforcement way.

3.1.3 Merits. There are three major merits of using such a framework for sequential recommendation. First, generally speaking, it is relatively difficult to train a capable generation-based sequential recommender using a direct optimization with a maximum likelihood loss (*e.g.,* exposure bias or data sparsity [23]). We utilize the discriminators to monitor the quality of the recommendation results of the generator, which are able to gradually improve the final recommendation performance. Second, it is more flexible to incorporate various kinds of factor information into discriminators, so that the generator can focus on the generation task itself. Such a way is more resistible to useless or noisy information from context data. It is easier to incorporate additional factors into an existing model. Third, instead of modeling all the factors in a discriminator, our framework decouples the effect of each factor by using multiple discriminators, which also improves the interpretability (*e.g.,* explaining why a special behavioral transition occurs) of the generated sequences.

To instantiate the framework, we adopt the recently proposed self-attentive neural architecture (*e.g.,* Transformer [27]) to develop the components of generator and discriminators, since it has been shown to be successful in various sequence-oriented tasks, including sequential recommendation [14]. While, it is flexible to implement our framework with other classes of models in practice. In the following sections, we will introduce the details of both components.

3.2 The Generator Component

In the MFGAN framework, let G_θ denote the generator component, parameterized by θ, where θ denotes the set of all the related parameters in G. We develop the generator for sequential recommendation model by stacking the embedding layer, self-attention block, and the prediction layer to generate the target items. Next, we describe each part in detail.

3.2.1 Embedding Layer. We maintain an item embedding matrix $\boldsymbol{M}_G \in \mathbb{R}^{|\mathcal{I}|\times d}$ to project original one-hot representations of items to d-dimensional dense representations. Given a n-length sequence of historical interactions, we apply a look-up operation from $\boldsymbol{M}_G$ to form the input embedding matrix $\boldsymbol{E} \in \mathbb{R}^{n\times d}$. Furthermore, we incorporate a learnable position encoding matrix $\boldsymbol{P} \in \mathbb{R}^{n\times d}$ to enhance the input representations. In this way, the input representations $\boldsymbol{E}_G \in \mathbb{R}^{n\times d}$ for the generator can be obtained by summing two embedding matrices: $\boldsymbol{E}_G = \boldsymbol{E} + \boldsymbol{P}$.

3.2.2 Self-attention Block. Based on the embedding layer, we stack multiple self-attention blocks. A self-attention block generally consists of two sub-layers, a multi-head self-attention sub-layer and a point-wise feed-forward network. Instead of attending information of user sequences with a single attention function, multi-head self-attention mechanism has been adopted for effectively extracting the information from different representation subspaces. Specifically, multi-head self-attention is defined as below:

$$\begin{aligned} \text{MultiHeadAtt}(\boldsymbol{F}^l) &= [head_i; head_2; \dots, head_h]\boldsymbol{W}^O, \\ head_i &= \text{Attention}(\boldsymbol{F}^l\boldsymbol{W}_i^Q, \boldsymbol{F}^l\boldsymbol{W}_i^K, \boldsymbol{F}^l\boldsymbol{W}_i^V), \end{aligned} \tag{1}$$

where the $\boldsymbol{F}^l$ is the input for the l-th layer. When $l = 0$, we set $\boldsymbol{F}^l$ to the input $\boldsymbol{E}_G$ of the generator. Let h denote the number of heads. The projection matrix $\boldsymbol{W}_i^Q \in \mathbb{R}^{d\times d/h}$, $\boldsymbol{W}_i^K \in \mathbb{R}^{d\times d/h}$, $\boldsymbol{W}_i^V \in \mathbb{R}^{d\times d/h}$ and $\boldsymbol{W}^O \in \mathbb{R}^{d\times d}$ are the corresponding learnable parameters for each attention head. The attention function is implemented by scaled dot-product operation:

$$\text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d/h}})\mathbf{V}, \tag{2}$$

where $\mathbf{Q} = \boldsymbol{F}^l\boldsymbol{W}_i^Q, \mathbf{K} = \boldsymbol{F}^l\boldsymbol{W}_i^K$, and $\mathbf{V} = \boldsymbol{F}_l\boldsymbol{W}_i^V$, which are the linear transformations of the input embedding matrix. The temperature $\sqrt{d/h}$ is the scale factor to avoid large values of the inner product. In sequential recommendation, we can only utilize the information before current time step, and we apply the mask operation to the output of the multi-head self-attention function, removing all connections between $\mathbf{Q}_i$ and $\mathbf{K}_j$ (for all cases of $j > i$).

As shown in Eq. (1), the multi-head attention function is mainly built on the linear projections. We further endow the non-linearity of the self-attention block by applying a point-wise feed-forward network as:

$$\text{PFFN}(\boldsymbol{F}^l) = [\text{FFN}(\boldsymbol{F}_1^l)^\top; \cdots; \text{FFN}(\boldsymbol{F}_n^l)^\top], \tag{3}$$
$$\text{FFN}(\boldsymbol{x}) = \max(0, \boldsymbol{x}\boldsymbol{W}_1 + \boldsymbol{b}_1)\boldsymbol{W}_2 + b_2, \tag{4}$$

where $\boldsymbol{W}_1, \boldsymbol{b}_1, \boldsymbol{W}_2, b_2$ are the trainable parameters and not shared across layers.

3.2.3 Prediction Layer. At the final layer of the generator, we calculate the user's preference over the item set through the softmax function:

$$G_\theta(i_t|i_{1:t-1}) = \text{softmax}(\boldsymbol{F}_n^L\boldsymbol{M}_G^\top)_{[i_t]}, \tag{5}$$

where L is the number of self-attention blocks and $\boldsymbol{M}_G$ is the maintained item embedding matrix defined in Section 3.2.1.

3.3 Factor-specific Discriminator Components

As mentioned before, we consider m kinds of factor information that is useful to improve the sequential recommendation. Instead of directly feeding them into the generator, we set a unique discriminator for each factor, such that various kinds of context information can be utilized and decoupled via the factor-specific discriminators.

Specially, we have m discriminators $D_\Phi = \{D_{\phi_1}, D_{\phi_2}, \dots, D_{\phi_m}\}$, in which the j-th discriminator is parameterized by ϕ_j. The function of each discriminator is to determine whether the generated recommendation sequence by the generator is rational or not. This is cast as a binary classification task, *i.e.,* discriminating between generated or actual recommendation sequence. We assume that different discriminators are equipped with different parameters and work independently.

3.3.1 Embedding Layer. Considering a specific discriminator D_{ϕ_j}, we first construct an input embedding matrix $\boldsymbol{E}_D^j \in \mathbb{R}^{n\times d}$ for a n-length sequence by summing the factor-specific embedding matrix $\boldsymbol{C}^j$ and the positional encoding matrix $\boldsymbol{P}$, namely $\boldsymbol{E}_D^j = \boldsymbol{C}^j + \boldsymbol{P}$. To construct the $\boldsymbol{C}^j$, we adopt a simple yet effective method: first discretize the possible values of a factor into several bins, then set a unique embedding vector for each bin, and finally derive $\boldsymbol{C}^j$ using a look-up operation by concatenating the embeddings for the bin IDs from the input sequence.

3.3.2 Architecture. To develop the discriminator, we adopt the similar architecture of the generator. In our framework, the generator predicts the recommended sequence, and the discriminators are mainly used to improve the generator. Hence, we adopt a relatively weak architecture with only one self-attention block for avoiding the case that the discriminator is too strong and cannot send suitable feedback to the generator. The one-layer self-attention block is computed as:

$$\boldsymbol{A}^j = \text{MultiHeadAtt}(\boldsymbol{E}_D^j), \tag{6}$$
$$\boldsymbol{H}^j = \text{PFFN}(\boldsymbol{A}^j). \tag{7}$$

Note that unlike the generator, the self-attention block of the discriminator can refer to the information of subsequent positions when trained at the t-th position. Hence, the discriminator adopts a *bi-directional* architecture by removing the mask operation. In this way, the discriminator can model the interaction between any two positions, and make a more accurate judgement by considering the overall sequential characteristics. While, the generator does not utilize such bi-directional sequential characteristics. As such, the discriminator is expected to provide additional supervision signals, though it shares the similar architecture with the generator.

Finally, the degree of the rationality for the generated recommendation sequence is measured by a Multiple-Layer Perceptron (MLP):

$$\hat{y}_j = \text{MLP}(\boldsymbol{H}_n^j), \tag{8}$$

where $\hat{y}_j$ is the predicted degree of the rationality from the the MLP component based on the output of the self-attention block $\boldsymbol{H}^j$. A rationality score reflects the probability that a sequence is from actual data distribution judged by some discriminator.

Since we have m discriminators *w.r.t.* different factors, we can obtain a set of predicted rationality scores $\{\hat{y}_1, \hat{y}_2, \dots, \hat{y}_m\}$. As will be illustrated later, these rationality scores can be used for supervision signals to guide the learning of the generator.

3.4 Multi-adversarial Training

As described previously, there is one generator G_θ and multiple discriminators $D_\Phi = \{D_{\phi_1}, D_{\phi_2}, \dots, D_{\phi_n}\}$. The generator G_θ successively predicts the next item based on historical sequence data, and the discriminators try to discriminate between the predicted sequence and the actual sequence. In this part, we present the multi-adversarial training algorithm for our approach.

3.4.1 RL-based Formalization. Because sampling from the item set is a discrete process, gradient descent cannot be directly applied to solve the original GAN formulation for our recommendation task. As such, following [32], we first formalize the sequential recommendation task in a reinforcement learning (RL) setting. At the t-step, the *state s* is represented by the previously recommended sub-sequence $i_{1:t-1} = \{i_1, i_2, \dots, i_{t-1}\}$; the *action a* is to select the next item i_t for recommendation, controlled by a *policy* π that is defined according to the generator: $\pi(a = i_t|s) = G_\theta(i_t|i_{1:t-1})$; when an action is taken, it will transit from s_t to a new state s', corresponding to the sub-sequence $i_{1:t} = \{i_1, i_2, \dots, i_t\}$; taking an action will lead to some *reward r*. The key point is that we utilize the discriminator components to provide the reward signal for guiding the learning of the generator. We define the expected return $Q(s, a)$ for a pair of state and action, namely the Q-function, as below

$$Q(s = i_{1:t-1}, a = i_t) = \sum_{j=1}^{m} \omega_j \hat{y}_j, \tag{9}$$

where $\hat{y}_j$ is the rationality score (Eq. (8)) of current sequence according to the j-th discriminator, and ω_j is the combination coefficient defined through a λ-parameterized softmax function:

$$\omega_j = \frac{\exp(\lambda \hat{y}_j)}{\sum_{j'=1}^{m} \exp(\lambda \hat{y}_{j'})}, \tag{10}$$

where λ is a tuning parameter that will be discussed later.

As the discriminator is updated iteratively, it gradually pushes the generator to its limit, which will generate more realistic recommended items. Through multiple-factor enhanced architecture, the generator can obtain guidance of sequential characteristics in the interaction sequence from different perspectives.

3.4.2 Learning Algorithm. After the task is formulated as a RL setting, we can apply the classic policy gradient to learn the model parameters of the generator. Formally, the objective of the generator $G_\theta(i_t | i_{1:t-1})$ is to maximize the expected reward at the t-th step:

$$\mathcal{J}(\theta) = \mathbb{E}[R_t | i_{1:t-1}; \theta] = \sum_{i_t \in \mathcal{I}} G_\theta(i_t | i_{1:t-1}) \cdot Q(i_{1:t-1}, i_t),$$

where $G_\theta(i_t | i_{1:t-1})$ and $Q(i_{1:t-1}, i_t)$ are defined in Eq. (5) and Eq. (9), respectively. R_t denotes the reward of a generated sequence. The gradient of the objective function $\mathcal{J}(\theta)$ *w.r.t.* the generator's parameters θ can be derived as:

$$\begin{aligned}
\nabla_\theta \mathcal{J}(\theta) &= \nabla_\theta \sum_{i_t \in \mathcal{I}} G_\theta(i_t | i_{1:t-1}) \cdot Q(i_{1:t-1}, i_t) \\
&= \sum_{i_t \in \mathcal{I}} \nabla_\theta G_\theta(i_t | i_{1:t-1}) \cdot Q(i_{1:t-1}, i_t) \\
&= \sum_{i_t \in \mathcal{I}} G_\theta(i_t | i_{1:t-1}) \nabla_\theta \log G_\theta(i_t | i_{1:t-1}) \cdot Q(i_{1:t-1}, i_t) \\
&= \mathbb{E}_{i_t \sim G_\theta(i_t | i_{1:t-1})} [\nabla_\theta \log G_\theta(i_t | i_{1:t-1}) \cdot Q(i_{1:t-1}, i_t)].
\end{aligned} \tag{11}$$

We update the parameters of the generator using gradient ascent as follows:

$$\theta \leftarrow \theta + \gamma \nabla_\theta \mathcal{J}(\theta), \tag{12}$$

where γ is the step size of the parameter update.

After updating the generator, we continue to optimize each discriminator D_{ϕ_j} by minimizing the following objective loss:

$$\min_{\phi_j} -\mathbb{E}_{i_{1:t} \sim P_{\text{data}}} [\log D_{\phi_j}(i_{1:t})] - \mathbb{E}_{i_{1:t} \sim G_\theta} [\log(1 - D_{\phi_j}(i_{1:t}))]\}, \tag{13}$$

where P_{data} is the real data distribution.

Algorithm 1 presents the details of the training algorithm for our approach. The parameters of G_θ and multiple discriminators D_Φ are pretrained correspondingly. For each G-step, we generate the recommended item based on the previous sequence $i_{1:t-1}$, and then update the parameter by policy gradient with the reward provided from multiple discriminators. For each D-step, the recommended sequence is considered as the negative samples and we take the actual sequence from training data as positive ones. Then the discriminators are updated to discriminate between positive and negative sequences accordingly. Such a process is repeated until the algorithm converges.

Algorithm 1 The learning algorithm for our MFGAN framework.

```
Require: generator $G_\theta$; discriminators $D_\Phi = \{D_{\phi_1}, \dots, D_{\phi_m}\}$;
    user-item interactive sequence dataset $\mathcal{S}$
1: Initialize $G_\theta$, $D_\Phi$ with random weights $\theta$, $\Phi$
2: Pre-train $G_\theta$ using MLE
3: Generate negative samples using $G_\theta$ for training $D_\Phi$
4: Pre-train $D_\Phi$ via minimizing cross-entropy
5: repeat
6:     for $G$-steps do
7:         Generate the predicted item $i_t$ using $i_{1:t-1}$
8:         Obtain the generated sequence $i_{1:t}$
9:         Compute $Q(s = i_{1:t-1}, a = i_t)$ by Eq. (9)
10:        Update generator parameters $\theta$ via policy gradient Eq. (12)
11:    end for
12:    for $D$-steps do
13:        Use $G_\theta$ to generate negative examples
14:        Train $m$ discriminators $D_\Phi$ by Eq. (13)
15:    end for
16: until Convergence
```

3.5 Discussion and Analysis

In this section, we analyze the effectiveness and the stability of the multi-adversarial architecture in our task.

As mentioned before, we train the MFGAN model in an RL way by policy gradient. Since we have multiple discriminators, from each discriminator we receive a reward to guide the training process of the generator. Recall that we use a λ-parameterized softmax function to combine the reward signals from multiple discriminators in Eq. (10). By using such a parameterization, our reward function can be implemented in several forms:

(1) $\lambda \rightarrow -\infty$: it selects the discriminator with the minimum reward, *i.e., min*;

(2) $\lambda \rightarrow +\infty$: it selects the discriminator with the maximum reward, *i.e., max*;

(3) $\lambda = 0$: it becomes a simple average over all the discriminators, *i.e., mean*;

(4) Others: it adopts a "soft" combination of multiple reward values.

Among the four cases, we first study two extreme strategies, namely *max* and *min*. As shown in [6], it is too harsh for the generator by adopting the minimum reward from the discriminators. Let $p_G(x)$ denote the distribution induced by the generator. The low reward only indicates the position where to decrease $p_G(x)$, and does not specifically indicate the position where to increase $p_G(x)$. In addition, decreasing $p_G(x)$ will increase $p_G(x)$ in other regions of distribution space $\mathcal{X}$ (keeping $\int_{\mathcal{X}} p_G(x) = 1$), and the correctness of this region cannot be guaranteed. Hence, the *min* strategy is not good to train our approach. Conversely, *max* always selects the maximum reward that is able to alleviate the above training issue of *min*. However, since multiple factors are involved in the discriminators, some "prominent" factor will dominate the learning process, leading to insufficient training of other factors.

Compared with the former two cases, the latter two cases seem to be more reasonable in practice. They consider the contributions from all the discriminators. Specially, we can have an interesting observation: the gradient $\nabla_\theta \mathcal{J}(\theta)$ of the generator calculated in Eq. (11) is more robust for model learning due to the use of multiple discriminators. The major reason is that the Q-function in Eq. (9) is equal to zero *if and only if* $D_{\phi_j} = 0$ for all j, when all the discriminators give zero reward. Therefore, using multiple discriminators is indeed useful to stabilize the learning process of our approach. In our experiments, we do not observe a significant difference between the last cases on our task. Hence, we adopt the simpler *mean* way to set our reward function.

Our work is closely related to general sequence prediction studies with adversarial training [32] or reinforcement learning [23]. Similar to SeqGAN [32], we set up two components with different roles of *generator* and *discriminator*, respectively. Also, it is easy to make an analogy between the two roles and the concepts of "*actor*" and "*critic*" in the actor-critic algorithm in RL [23]. Compared with previous sequential recommendation models [11, 12, 16], our approach takes a novel perspective that decouples various factors from the prediction component. In our model, each discriminator has been fed with the information of some specific factor. Such a way is able to enhance the interpretability of the generator, that is to say, the reward values of discriminators can be treated as the importance of influencing factors at each time step. With the proposed framework, we believe there is much room to consider more advanced implementations or functions for generator and discriminators for improving sequential recommendation.

4 EXPERIMENTS

In this section, we first setup the experiments, then report major comparison results and other detailed analysis.

4.1 Dataset Construction

We construct experiments on three public datasets from different domains, including MovieLens-1M movie [9], Yahoo! music[1] and Steam [14] game. Since the Yahoo! dataset is very large, we randomly sample a subset of ten thousand users from the entire dataset. We group the interaction records by users, sort them by the timestamps ascendingly, and form the interaction sequence for each user. Following [26], we only keep the k-core dataset, filtering out unpopular items and inactive users with interaction records which are fewer than k. We set $k = 5, 10, 5$ for the MovieLens-1M, Yahoo! and Steam datasets, respectively.

The three datasets contain several kinds of context information. We extract such context information as *factors*:

(1) For MovieLens-1M dataset, we select category, popularity, and knowledge graph information as factors. Note that we treat the knowledge base (KB) information as a single factor, since we would like to develop a strong factor. We use the KB4Rec dataset [34] to obtain *item-to-entity* alignment mapping, and then obtain the KB information from Freebase [8]. We adopt the classic TransE [1] to learn the factor representation for KB information.

[1]https://webscope.sandbox.yahoo.com/catalog.php?datatype=r.

(2) For Yahoo! dataset, we select category, popularity, artist and album as factors.

(3) For Steam dataset, we select category, popularity, price and developer as factors.

Note that for all the three datasets, we have incorporated a special kind of factor, *i.e.,* item ID. Although it has been utilized in the generator, it can be utilized by the discriminator in a different way, *i.e.,* using a bi-directional Transformer architecture. Item ID can be used as a reference to compare the usefulness of the other kinds of factor information. Following [14], we split each user interaction sequence into three parts: the last item in the interaction sequence is treated as test data, the item just before the last is used for validation and the rest data is considered as training data. The statistics of three datasets after preprocessing are summarized in Table 1.

Table 1: Statistics of the three datasets.

Dataset	#Users	#Items	#Interactions	#Factors
Movielens-1m	6,040	3,361	996,834	4
Music	10,000	136,630	3,732,463	5
Game	51,995	12,037	1,789,953	5

4.2 Experimental Settings

4.2.1 Evaluation Protocol. To assess whether our method can improve the sequential recommendation, we adopt a variety of evaluation metrics widely used in previous works [12, 13, 26]: Mean Reciprocal Rank (MRR), top-k Normalized Discounted cumulative gain (NDCG@10) and Hit Ratio (HR@10). Since the item set is large, it is time-consuming to enumerate all the items as candidate. For each positive item in the test set, we pair it with 100 sampled items that the user has not interacted with as negative items. The evaluation metrics can be computed according to the rankings of these items. We report the average score over all test users.

4.2.2 Implementation Details. For our proposed MFGAN, we use two self-attention blocks in the generator and one self-attention block in discriminators. In both generator and discriminators, item embeddings are shared in embedding layer and prediction layer. We implement MFGAN with *Tensorflow*, and use *Adam* optimizer. At the pre-training stage, we first train the generator and discriminators to converge separately. At the adversarial training stage, we alternatively train the generator and discriminators with 100 epochs and 1 epoch, respectively. The learning rates are set to 0.001 for MovieLens-1M dataset, and 0.0002 for Steam and Yahoo! datasets, and the dropout rates are all set to 0.2 for the three datasets. For each factor, we discretize its possible values into several bins if needed. Each bin is associated with an embedding vector. The item embedding size and all the factor embedding sizes are set to 50, the KB embedding size with TransE is set to 100. The mini-batch sizes are set to 128 in generator and 16 in discriminators.

4.2.3 Comparison Methods. Here, we compare our propose approach MFGAN against a number of competitive baselines. Our baselines include related methods on general and sequential recommendation with or without context information:

• **PopRec**: This is a method that sorts items by their popularity to assign each item a rank, tending to recommend highly popular items to users.

Table 2: Performance comparison of different methods for sequential recommendation task on three datasets. We use bold and underline fonts to denote the best performance and second best performance method in each metric respectively.

Models	Movie			Music			Game		
	NDCG@10	HR@10	MRR	NDCG@10	HR@10	MRR	NDCG@10	HR@10	MRR
PopRec	0.2290	0.4200	0.1940	0.3984	0.6322	0.3408	0.4129	0.6738	0.3477
BPR	0.3063	0.5576	0.2505	0.4538	0.6695	0.3992	0.4203	0.6919	0.3511
FM	0.1935	0.3522	0.1690	0.2196	0.4165	0.1812	0.1383	0.2818	0.1216
IRGAN	0.2263	0.4201	0.1903	0.2857	0.5120	0.2332	0.3942	0.6525	0.3307
FPMC	0.3754	0.5990	0.3237	0.4330	0.5768	0.4022	0.3699	0.6167	0.3107
GRU	0.4014	0.6233	0.3763	0.4843	0.7778	0.4053	0.2383	0.4212	0.2037
GRU$_F$	0.4130	0.6409	0.3552	0.5470	0.8112	0.3961	0.2806	0.4860	0.2369
SASRec	0.5849	0.8132	0.5214	0.8356	0.9293	0.8074	0.5895	0.8351	0.5203
MFGAN	**0.6185**	**0.8318**	**0.5587**	**0.8556**	**0.9417**	**0.8286**	**0.6035**	**0.8488**	**0.5346**

• **BPR** [25]: This is a classic personalized ranking algorithm that optimizes the pairwise ranking loss function of latent factor model with implicit feedback.

• **FM** [24]: It uses a generic matrix factorization to learn the coefficients of combined features, considering the interactions between different features.

• **IRGAN** [28]: This method combines generative and discriminative information retrieval via adversarial training, in which a simple matrix factorization is used for the discriminator.

• **FPMC** [26]: It combines Matrix Factorization and Markov Chain, which can simultaneously capture sequential information and long-term user preference.

• **GRU** [10]: It is a GRU-based sequential recommender with session-parallel mini-batch training, which employs ranking-based loss functions. We implement an enhanced version by replacing one-hot vectors with pre-trained BPR vectors.

• **GRU$_F$** [11]: It proposes to incorporate additional feature vector as the input of GRU networks, which incorporates auxiliary features to improve sequential recommendation.

• **SASRec** [14]: It is a next-item sequential recommendation method based on the Transformer architecture, which adaptively considers interacted items for prediction. This method is the state-of-the-art baseline for sequential recommendation.

• **MFGAN**: This is our approach introduced in Section 3.

Besides GRU$_F$, there are other baselines that utilize context information. However, we have empirically found that the Transformer architecture is more superior than other sequence neural networks in sequential recommendation. Indeed, SASRec is significantly better than quite a few context-aware recommendation algorithms with other RNN architectures. Therefore, our focus is how our approach improves over SASRec, and we omit other context-aware baselines. Another note is that in comparison our MFGAN approach only utilizes the generator to recommend items, *i.e.,* context information or discriminators will not be used at the test stage.

4.3 Results and Analysis

The results of different methods for sequential recommendation are presented in Table 2. It can be observed that:

(1) Among non-sequential recommendation baselines, we can find that BPR outperforms other methods, and non-sequential recommendation methods overall perform worse than sequential recommendation methods. Specially, factorization machine (FM) does not work very well in this task, since it still adopts the regression loss and cannot effectively capture the preference order over two items by a user. Besides, popularity seems to be a robust baseline that gives substantial performance on our datasets. A major reason is that there may exist the "*rich-gets-richer*" phenomenon in product adoption, and using k-core preprocessing on our datasets further enhances this trend.

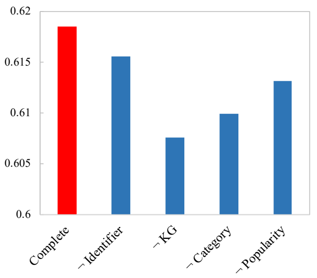
(2) Among sequential recommendation baselines, the Markov Chain-based method performs better on sparse dataset (*i.e.,* the game dataset) than dense datasets (*i.e.,* the other two datasets). As for two neural methods, GRU adopts the RNN-based architecture, serving as a standard comparison method for sequential recommendation. With available context information, GRU$_F$ further improves over the GRU method, indicating context information is useful in our task. Furthermore, SASRec utilizes the powerful Transformer architecture to develop the sequential recommender, achieving the best performance among all the baselines. In natural language processing, self-attention architecture has shown its superiority in various tasks [5, 27]. Such an architecture is particularly useful when dealing with sequence data, which also works well in sequential recommendation.

(3) Finally, we compare our proposed model MFGAN with the baseline methods. It is clear to see that MFGAN is consistently better than these baselines by a large margin. Our base architecture is a pre-trained self-attentive generator. Different from the above models, we adopt the multi-adversarial architecture to guide the learning of the generator with various kinds of factor information. Note that our generator has not directly used any context information, and it is guided by the signals from the discriminators. Such a comparison indicates the multi-adversarial training approach is effective to improve the performance of self-attention architecture for sequential recommendation.

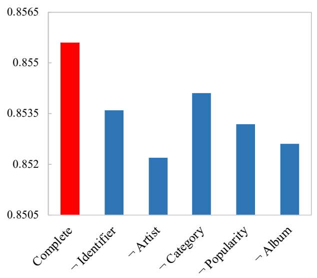
4.4 Detailed Analysis on Our Approach

In this section, we further conduct a series of detailed experiments to analyze the effectiveness of our approach.

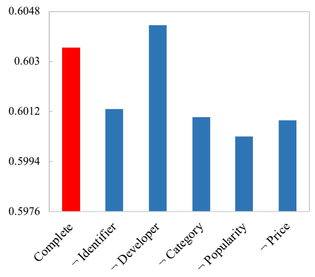
4.4.1 Effect of Different Factors. In this part, we study the effect of different kinds of factor information in our approach. We first prepare the complete MFGAN model, and then remove one kind of factor information at each time. In this way, we can study how each individual factor contributes to the final performance for sequential



(a) MovieLens-1m movie dataset.



(b) Yahoo! music dataset.



(c) Steam game dataset.

Figure 2: Effect of different kinds of factor information in our framework for sequential recommendation. We report the results using the NDCG@10 metric. "¬" indicates that we remove the corresponding factor information, while the rest factor information is kept.

recommendation. Figure 2 presents the NDCG@10 results of factor removal experiments for MFGAN. As we can see, all the factors seem to be useful to improve the performance of our MFGAN, except the factor *developer* on the Steam game dataset. The most useful factors are *knowledge graph*, *artist* and *popularity* for the three datasets, respectively. Interestingly, item ID is not always the most useful feature for discriminators, indicating the importance of other kinds of factor information.

4.4.2 Comparisons of Model Variants. In our model, we propose several techniques to improve the performance of our approach. Here, we construct comparison experiments to examine the effect of these techniques: (1) *Uni-directional* using a uni-directional self-attention architecture in discriminators; (2) *SDSF* using a single discriminator considering item ID as the only factor; (3) *SDAF* using a single discriminator incorporating all factor embeddings using simple vector concatenation. The first variant is used to illustrate the effect of the bi-directional architecture, and the last two variants are used to illustrate the effect of using multiple discriminators.

Table 3: Variant comparisons of our MFGAN framework on MovieLens-1m dataset.

Architecture	NDCG@10	HR@10	MRR
MFGAN	**0.6185**	**0.8318**	**0.5587**
SASRec	0.5849	0.8132	0.5214
Uni-Directional	0.6158	0.8299	0.5558
SDSF	0.6032	0.8212	0.5413
SDAF	0.6039	0.8221	0.5455

Table 3 presents the comparisons of these variants against our complete approach and SASRec. We can see that all the variants are worse than the complete approach. It indicates that the bi-directional architecture and multi-discriminator adversarial training are effective to improve the performance. However, the improvement with bi-directional architecture seems to be small in numerical values. A possible reason is that the uni-directional Transformer architecture is ready very competitive. While, the idea that utilizes the bi-directional architecture in discriminators has important implications. As future work, it will be interesting to design other architectures for generator and discriminator, respectively.

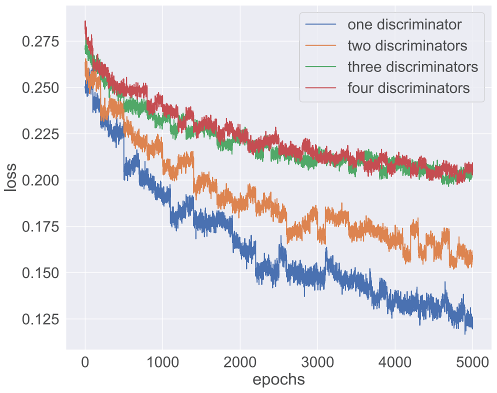


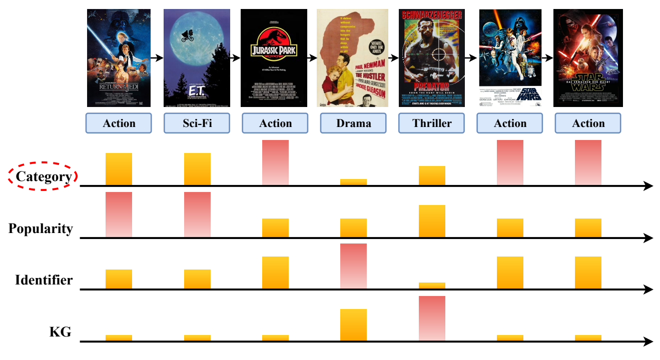
Figure 3: Convergence of loss with different numbers of discriminators in MovieLens-1m dataset.

Note that the *SDSF* variant can be considered as an enhanced implementation of SeqGAN [32], where the generator and the discriminator have been improved using the Transformer architecture. SeqGAN is originally proposed for general sequence generation, while it is easy to adapt to our task.

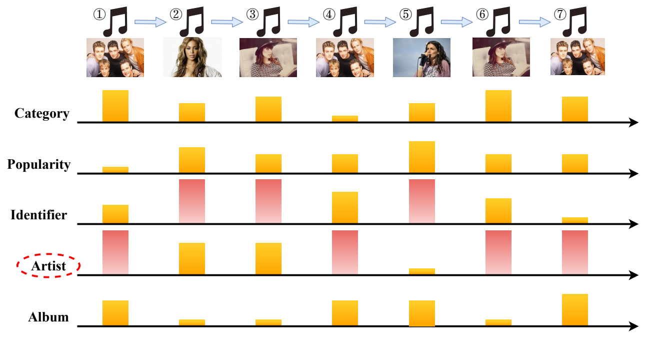
Besides the performance improvement, as discussed in Section 3, using multiple discriminators is likely to stabilize the training of adversarial learning approaches. For this purpose, we further compare the convergence of our objective function with a varying number of discriminators during training process. As we can see in Fig. 3, using more discriminators is faster to achieve a relatively stable loss. The red and green lines (corresponding to use three or four discriminators) become stable after about 2000 epochs, while the other lines are not stable even after 4000 epochs. Another interesting observation is that using more discriminators seems to yield a larger loss. We speculate that it has the similar effect of regularization, which prevents the model parameters from overfitting on training data.

4.5 Qualitative Analysis on the Recommendation Interpretability

Previous experiments have verified that our model is able to generate high-quality sequential recommendations. Another major



(a) A case from MOVIELENS-1M movie dataset.



(b) A case from YAHOO! music dataset.

Figure 4: Two qualitative cases with our approach. In each figure, the upper part is the interaction sequence, and the lower part corresponds to the lines of reward signal scores from some factors. The height of the histogram represents the reward score within the interval $(0, 1)$. We use the red color to highlight the highest bar (*i.e.,* the dominant factor) at each time step. For the two cases, we mainly focus on the *category* and *artist* factor, respectively.

benefit of our work is that our recommendations are highly interpretable due to the decoupled factors in the multi-adversarial framework. Here, we present two illustrative examples in Fig. 4, showing how our approach improves the recommendation interpretability with decoupled factors.

The first case (Fig. 4(a)) presents an interaction sequence of a sample user from MOVIELENS-1M dataset. As we can see, the first two movies "*Star War: episode VI*" and "*E.T. the Extra-Terrestrial*" have received higher reward signals from *popularity*, which is the prominent factor at the beginning stage. Interestingly, seen from the *category* line, the 3-rd, 6-th and 7-th movies are mainly driven by the category factor. Given the the last two movies "*Star War: episode IV*" and "*Star War: episode V*", the user, influenced by early popularity, seems to be a Star Wars fan or an action movie fan.

The second case (Fig. 4(b)) presents an interaction sequence of a sample user from YAHOO! music dataset. Similarly, we can check the effect of multiple factors at different time steps. Here, we mainly analyze the *artist* factor, focusing on the 4-th, 6-th and 7-th songs. Along the interaction sequence, this user switches among different artists. Our model is able to capture such listening patterns: when the user listens to a song of a previous artist (*i.e.,* previously appearing in the interaction sequence) occurs, the corresponding *artist* factor becomes dominant in our model.

As a comparison, existing methods mainly focus on integrating various kinds of context information into unified representations. The two examples have shown the advantage of decoupling various factors for sequential recommendation.

5 RELATED WORK

In this section, we review studies closely related to our work in two aspects.

Sequential Recommendation. Early works on sequential recommendation are mainly based on Markov Chain (MC) assumption. For instance, Rendle et al. [26] fuse the matrix factorization and first-order Markov Chain for modeling global user preference and short-term interests, respectively. Another line to model user behaviors is resorting to the recurrent neural network, which has achieved great success on sequential modeling in a variety of applications. Hidasi et al. [10] firstly introduce Gated Recurrent Units (GRU) to the session-based recommendation. A surge of following variants modify it by introducing pair-wise loss function [11], attention mechanism [18], memory network [3], hierarchical structure [22], etc. Other architectures or networks have also been used [19, 20], achieving good performance. Moreover, context information is often used to improve the recommendation performance and interpretability [11–13, 29]. Recently, self-attention network has achieved significant improvement in a bunch of NLP tasks [5, 27] and inspired a new direction on applying the self-attention mechanism to sequential recommendation problem [14].

Generative Adversarial Network. Generative Adversarial Network (GAN) [7] was originally proposed to mimic the generation process of given data samples. Typically, GAN consists of two components: the generative model learns to map from a latent space to a data distribution of interest, while the discriminative model distinguishes candidates produced by the generator from the true data distribution. A surge of follow-up works either improve the GAN framework by introducing more advanced training strategies, like f-divergence [21], Wasserstein distance [30], MMD constraints [17] or explore diverse applications under GAN framework [15, 32, 33]. In recommender systems, the idea of GAN has already been explored to some extent. IRGAN [28] is firstly proposed to unify the generative model and discriminative model in the field of information retrieval. Chae et al. [2] follow this line and further improve the training scheme by sampling a real-valued vector instead of a single item index. Moreover, personalized determinantal point process is utilized to improve recommendation diversity via adversarial training [31].

Our work is related to the above studies, while has a different focus. We are the first to apply adversarial training to sequential recommender systems. By designing a multi-adversarial network, our model is able to explicitly characterize the effect of each individual factor on sequential recommendation over time, which improves the recommendation interpretability.

6 CONCLUSIONS

In this paper, we have proposed a Multi-Factor Generative Adversarial Network (MFGAN) for sequential recommendation. In our framework, the generator taking user behavior sequences as input is used to generate possible next items, and multiple factor-specific discriminators are used to evaluate the generated sub-sequence from the perspectives of different factors. We have constructed extensive experiments on three real-world datasets. Experimental results have shown that our approach outperforms several competitive baselines. Especially, we have found that using multiple discriminators is useful to stabilize the training of adversarial learning, and also enhance the interpretability of recommendation algorithms.

Currently, we consider a simple setting where multiple discriminators are separately designed. As future work, we will investigate how to design a more principled way to share discrimination information across different discriminators. We will also consider incorporating explicit user preference in the discriminators.

7 ACKNOWLEDGEMENT

This work was partially supported by the National Natural Science Foundation of China under Grant No. 61872369 and 61832017, the Fundamental Research Funds for the Central Universities, the Research Funds of Renmin University of China under Grant No.18XNL G22 and 19XNQ047, Beijing Academy of Artificial Intelligence (BAAI) under Grant No. BAAI2020ZJ0301, and Beijing Outstanding Young Scientist Program under Grant No. BJJWZYJH012019100020098. Xin Zhao is the corresponding author.

REFERENCES

[1] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*. 2787–2795.

[2] Dong-Kyu Chae, Jin-Soo Kang, Sang-Wook Kim, and Jung-Tae Lee. 2018. Cfgan: A generic collaborative filtering framework based on generative adversarial networks. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 137–146.

[3] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential recommendation with user memory networks. In *Proceedings of the eleventh ACM International Conference on Web Search and Data Mining*. 108–116.

[4] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. 1724–1734.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 4171–4186.

[6] Ishan P. Durugkar, Ian Gemp, and Sridhar Mahadevan. 2017. Generative Multi-Adversarial Networks. In *5th International Conference on Learning Representations*.

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[8] Google. 2016. Freebase Data Dumps. https://developers.google.com/freebase/data.

[9] 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.

[10] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *4th International Conference on Learning Representations*.

[11] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. 2016. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 241–248.

[12] Jin Huang, Zhaochun Ren, Wayne Xin Zhao, Gaole He, Ji-Rong Wen, and Daxiang Dong. 2019. Taxonomy-aware multi-hop reasoning networks for sequential recommendation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 573–581.

[13] Jin Huang, Wayne Xin Zhao, Hongjian Dou, Ji-Rong Wen, and Edward Y Chang. 2018. Improving sequential recommendation with knowledge-enhanced memory networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 505–514.

[14] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 197–206.

[15] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew P. Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. 2017. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition*. 105–114.

[16] Chenliang Li, Xichuan Niu, Xiangyang Luo, Zhenzhong Chen, and Cong Quan. 2019. A Review-Driven Neural Model for Sequential Recommendation. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. 2866–2872.

[17] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. 2017. Mmd gan: Towards deeper understanding of moment matching network. In *Advances in neural information processing systems*. 2203–2213.

[18] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural Attentive Session-based Recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1419–1428.

[19] Fuyu Lv, Taiwei Jin, Changlong Yu, Fei Sun, Quan Lin, Keping Yang, and Wilfred Ng. 2019. SDM: Sequential deep matching model for online large-scale recommender system. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2635–2643.

[20] Chen Ma, Peng Kang, and Xue Liu. 2019. Hierarchical gating networks for sequential recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 825–833.

[21] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. 2016. f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization. In *Advances in neural information processing systems*. 271–279.

[22] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing Session-based Recommendations with Hierarchical Recurrent Neural Networks. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. 130–137.

[23] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence Level Training with Recurrent Neural Networks. *4th International Conference on Learning Representations* (2016).

[24] Steffen Rendle. 2012. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)* 3, 3 (2012), 1–22.

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

[26] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th International World Wide Web Conference*. 811–820.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in neural information processing systems*. 5998–6008.

[28] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 515–524.

[29] Shaoqing Wang, Cuiping Li, Kankan Zhao, and Hong Chen. 2017. Context-aware recommendations with random partition factorization machines. *Data Science and Engineering* 2, 2 (2017), 125–135.

[30] Jiqing Wu, Zhiwu Huang, Janine Thoma, Dinesh Acharya, and Luc Van Gool. 2018. Wasserstein divergence for gans. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 653–668.

[31] Qiong Wu, Yong Liu, Chunyan Miao, Binqiang Zhao, Yin Zhao, and Lu Guan. 2019. PD-GAN: Adversarial Learning for Personalized Diversity-Promoting Recommendation. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. 3870–3876.

[32] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[33] Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. 2017. Adversarial Feature Matching for Text Generation. In *Proceedings of the 34th International Conference on Machine Learning*. 4006–4015.

[34] Wayne Xin Zhao, Gaole He, Kunlin Yang, Hong-Jian Dou, Jin Huang, Siqi Ouyang, and Ji-Rong Wen. 2019. KB4Rec: A Data Set for Linking Knowledge Bases with Recommender Systems. *Data Intelligence* 1, 2 (2019), 121–136.