A Peep into the Future: Adversarial Future Encoding in Recommendation

Ruobing Xie*
WeChat, Tencent
Beijing, China
ruobingxie@tencent.com

Shaoliang Zhang*
WeChat, Tencent
Beijing, China
modriczhang@tencent.com

Rui Wang
WeChat, Tencent
Beijing, China
rysanwang@tencent.com

Feng Xia
WeChat, Tencent
Beijing, China
xiafengxia@tencent.com

Leyu Lin
WeChat, Tencent
Beijing, China
goshawklin@tencent.com

ABSTRACT

Personalized recommendation often relies on user historical behaviors to provide items for users. It is intuitive that future information also contains essential messages as supplements to user historical behaviors. However, we cannot directly encode future information into models, since we are unable to get future information in online serving. In this work, we propose a novel adversarial future encoding (AFE) framework to make full use of informative future features in different types of recommendation models. Specifically, AFE contains a future-aware discriminator and a generator. The future-aware discriminator takes both common features and future features as inputs, working as a recommendation prophet to judge user-item pairs. In contrast, the generator is considered as a challenger, which generates items with only common features, aiming to confuse the future-aware prophet. The future-aware discriminator can inspire the generator (to be deployed online) to produce better results. We further conduct a multi-factor optimization to enable a fast and stable model convergence via the direct learning and knowledge distillation losses. Moreover, we have adopted AFE on both a list-wise RL-based ranking model and a point-wise ranking model to verify its universality. In experiments, we conduct sufficient evaluations on two large-scale datasets, achieving significant improvements on both offline and online evaluations. Currently, we have deployed AFE on a real-world system, affecting millions of users. The source code is in https://github.com/modriczhang/AFE.

CCS CONCEPTS

• **Information systems → Recommender systems**.

KEYWORDS

recommendation, future information, GAN

*Both authors contributed equally to this research. Ruobing Xie is the corresponding author (ruobingxie@tencent.com).

ACM Reference Format:
Ruobing Xie, Shaoliang Zhang, Rui Wang, Feng Xia, and Leyu Lin. 2022. A Peep into the Future: Adversarial Future Encoding in Recommendation. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM ’22), February 21–25, 2022, Tempe, AZ, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3488560.3498476

1 INTRODUCTION

Personalized recommendation aims to provide appropriate items for users according to their preferences[16, 32]. In recommendation, user historical behaviors are one of the most important features to precisely capture user preferences, since the history is viewed as a mirror to reflect the future. There are lots of works that concentrate on learning from user historical behaviors via more sophisticated sequence-based models [21] and neural feature interaction models [4] to capture users’ diverse and variable preferences.

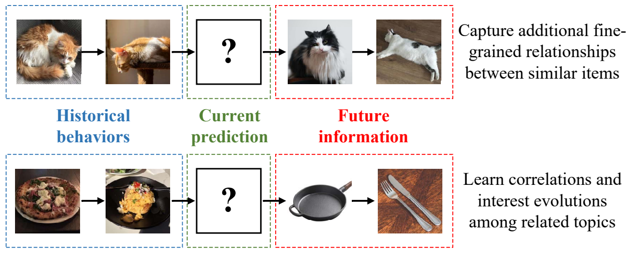
?
?
Capture additional fine-grained relationships between similar items
Historical behaviors
Current prediction
Future information
Learn correlations and interest evolutions among related topics

Figure 1: Examples and advantages of future information.

In real-world recommendation, it is straightforward and natural to predict items according to the historical information. However, the ***future information*** (i.e., information that can only be collected in the future) also contains essential messages as supplements to the current prediction, as long as we could peep into the future. As shown in Fig. 1, the effectiveness of the future information mainly locates in two aspects: (1) in the short term, users usually focus on one topic, whose historical and future behaviors are often similar to each other. For example, a user interested in cats may continuously seek cat-related information. The fine-grained relationships between these similar behaviors are bidirectional in the timeline. Hence, the future information can provide additional information to assist current recommendations. (2) In the long term, user’s concentrations may switch back and forth among multiple topics. In this case, modeling additional future information is more significant. It helps to capture the coarse-grained correlations between different but related topics, which enables models to predict user's interest evolution in advance (e.g., from gourmet to cooking in Fig. 1). In conclusion, a recommendation prophet can make full use of all user behaviors on both sides of the timeline, which greatly improves our understandings of user preferences.

However, it is extremely challenging to utilize future information in real-world recommendation. Different from bidirectional sequential models in NLP tasks [7] that can take the whole sequences as inputs, online recommendation *cannot* get future information since they happen in the future literally. In contrast, directly using future data as input features of the deployed model in offline training will result in overfitting on these future features, which will reduce the performance in the test set and online serving [1]. Moreover, incautiously using future information in testing will cause the fatal issue of data leakage, which is studied and should be avoided [22].

There are very few works that consider future information in recommendation. Recently, with the thriving of pre-training, some works introduce the masked language model (MLM) pre-training task in NLP [7] to sequence-based recommendation, which consider bidirectional information to learn better sequential models via self-supervised learning (SSL) [26, 40]. The MLM task randomly masks some items in the user behavior sequence during training, and then attempts to predict them with the remaining unmasked contexts [38, 39]. These models can be viewed as indirectly using a specific future feature (i.e., behaviors after the masked item) under SSL in sequence-based recommendation. However, there are optimization biases between the MLM task and the next-item prediction task of recommendation. Moreover, the MLM-based future modeling is limited to the specific future information in sequence modeling. Therefore, we attempt to build a universal framework for using any type of future information in general recommendation.

In this work, we build a novel **Adversarial future encoding (AFE)** framework to take advantage of useful future information in different types of recommendations. Specifically, AFE builds a **future-aware discriminator**, which is considered as the prophet to peep into the future. It takes both common features (e.g., user historical behaviors, item features, contexts) and future features as inputs, and outputs a ranking score of the current user-item pairs. Enhanced by the future information, the discriminator becomes more powerful to score the current user-item instances. In contrast, AFE also builds a **generator**, which works as a challenger to fight against the future-aware discriminator. It takes common features as inputs like conventional ranking models, and then generates "fake" items it thinks will be clicked to confuse the discriminator. Through the minimax game with a stronger future-aware discriminator, the generator could get a more sufficient training and provide better recommendations. In online, the generator will be deployed with common features as inputs. We successfully adopt this future-aware GAN framework on both a real-world list-wise RL-based recommendation model and a general point-wise ranking model. To ensure a stable and effective training, we further conduct a direct learning loss and a future-aware knowledge distillation (KD) loss during the adversarial training. The advantages of AFE are as follows: (1) AFE can capture more significant features from user historical behaviors under the spur of the future-aware discriminator. (2) The generator to be deployed in online does not directly take future features as inputs, which prevents the feature inconsistency and overfitting issues. Moreover, AFE is directly optimized via recommendation losses (e.g., the next item prediction task), which greatly reduces the optimization biases in the MLM pre-training task. (3) The future-aware GAN framework is universal. It is convenient to conduct AFE with almost all types of recommendation models.

In experiments, we evaluate AFE on two large-scale industrial datasets from a real-world system. AFE is deployed with both classical list-wise RL model [32] and point-wise ranking model [20] to verify its flexibility and universality. AFE achieves significant improvements compared to various types of competitive baselines on both offline and online evaluations. Moreover, we also conduct several ablation tests to demonstrate the effectiveness of different components. The contributions of AFE are concluded as follows:

- We systematically highlight the significance and challenges of using future information in general recommendation. To the best of our knowledge, we are the first to bring in future information via GAN in general recommendation.
- We propose a novel adversarial future encoding, which builds a future-aware discriminator to boost the generator's training. We also combine the adversarial training with a direct learning loss and a future-aware knowledge distillation loss.
- We achieve significant improvements on offline and online evaluations with different types of models. AFE has been deployed on a real-world system for millions of users.

2 RELATED WORKS

Real-world Recommendation. Matrix factorization (MF) [15] and Factorization machine (FM) [18] are classical recommendation models. Recently, neural models [3, 9, 11, 16, 28, 30] have been successfully verified in modeling (high-order) feature interactions. AutoInt [20] and AFN [4] further bring in self-attention and logarithmic transformation layer to capture useful feature interactions. Real-world recommendation algorithms usually need to provide an item list (rather than a single item) for each user request [32]. Differing from top-N recommendation [34], list-wise recommendation should further consider the item chemistries (e.g., diversity) and the quality of the whole list, where reinforcement learning (RL) methods have been successfully verified [29, 31]. [42] models the sequential interactions between users and a recommender system as a Markov decision process via deep RL. MaHRL [41] and HRL-Rec [32] further conduct hierarchical RL to model multi-goals abstraction and multiple channels. In this work, we also adopt an RL-based model with AFE on a real-world list-wise recommendation.

Future Modeling in Recommendation. There are some works that indirectly consider future behaviors in sequence-based recommendation [40]. These models are mostly inspired by the masked language model (MLM) of pre-training [7]. It randomly masks some tokens in a sequence, and then predicts them via the rest contexts. Some models [2, 21, 26, 38, 39] adopt the MLM task in sequence-based recommendation, indirectly considering the future behaviors after the current position via MLM in training. However, there are optimization biases between MLM and next-item prediction tasks. Moreover, the scenario (i.e., sequence-based recommendation) and future information type (i.e., user behaviors) of this future modeling are also limited, which is hard to use in general recommendation.

In general recommendation, there are very few works that consider future data. [17] considers other users' behaviors after the last historical behavior (but before the current time) of the target user as future information, which is not real future data. PFD [33] is the most related work, which adopts a KD block to model privileged features (i.e., important features of the current items that cannot be used in offline). However, PFD is not purposefully verified in future information. Differing from these methods, our AFE explicitly considers future features in a universal adversarial framework.

Adversarial Training in Recommendation. Generative adversarial network (GAN) proposes a minimax game framework for estimating generative models via adversarial networks [8]. IRGAN [24] brings GAN into information retrieval tasks. It conducts a generator to provide fake documents for the discriminator, using a policy gradient based RL to update the generator. Adversarial training has also been utilized in social recommendation [37], adversarial attack [5], and feature transfer [10]. In this work, we encode future information into the discriminator. It encourages the generator to win the minimax game against the strong prophet.

3 METHODOLOGY

3.1 Overall Framework

Fig. 2 displays the overall framework of AFE. We can find that the input features are roughly divided into two types, namely the common feature and the future feature. The ***common feature*** consists of classical features in recommendation such as user features, item features, recommendation contexts, and user historical behaviors, which are combined to model the current and previous states. In contrast, the ***future feature*** represents the future information of the user, item, and system. In AFE, the common features are fed to both generator and discriminator, while the future features are only given to the discriminator. During the minimax game, the generator aims to generate appropriate items that look like the ground-truth clicked items, whereas the future-aware discriminator will try to clearly distinguish the real clicked items from the generated "fake" clicked items with the help of additional future features. The discriminator is regarded as a prophet armed with powerful future features, which multiplies the challenges of the generator in confusing the discriminator but also encourages the generator to learn better representations. Besides the GAN training, we further conduct a multi-factor optimization containing a direct learning loss and a future-aware KD loss, which ensures a more effective and stable training with complicated practical models.

3.2 Future Information

We define the *future information* as all information that can only be collected in the future. The future information is an inclusive and broad concept, which could be any types and forms of useful information deriving from users, items, recommendation contexts, and systems in the future of the current prediction. For example, it could consist of user's multi-type behaviors (e.g., click, unclick), item popularity, updated user profiles, and hot topics in the future.

Specifically in this work, we consider the target user's future click behaviors and the future clicked taxonomy information (containing tag, category, item type) as the future information. For a user's future click behavior $\{d_{t+1}, \cdots, d_{t+n}\}$ after time t, we first adopt their trainable ID embeddings $\{\boldsymbol{d}_{t+1}, \cdots, \boldsymbol{d}_{t+n}\}$ of items as the ID feature field of the future information. For the taxonomy feature field, we first project all tags, categories, and types of these future clicked items into taxonomy embeddings. Next, all tag embeddings are then aggregated via an average pooling to get the tag field embedding $\boldsymbol{t}_g$. The category and item type field embedding $\boldsymbol{t}_c$ and $\boldsymbol{t}_y$ are built the same as $\boldsymbol{t}_g$. Item tag and category model user's fine-grained and coarse-grained interests respectively, and item type reflects user's preferences on different content forms (e.g., video, article). The future feature f_f contains $3 + n$ fields of future items and taxonomies, noted as $\{\boldsymbol{d}_{t+1}, \cdots, \boldsymbol{d}_{t+n}, \boldsymbol{t}_g, \boldsymbol{t}_c, \boldsymbol{t}_y\}$. We use these trainable ID embeddings and taxonomies as future information in AFE, since they (1) reflect useful future information from different aspects, (2) perform well in online scenarios, and (3) are easy to collect for training and serving. Note that we only consider the future behaviors in the same session of the current prediction in order to improve efficiency and remove noise. It is also convenient to adopt other future features in our AFE framework.

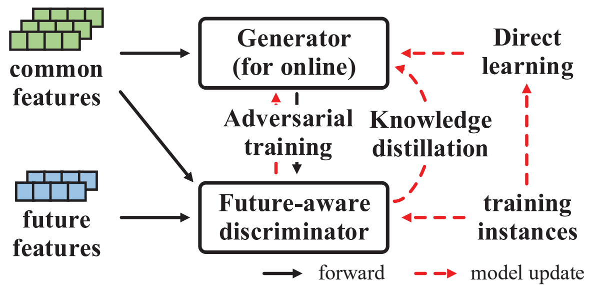

Figure 2: Overall framework of AFE.

3.3 Adversarial Future Encoding

The key challenge of this work is how to make full use of future information. We propose a novel and universal AFE framework, which indirectly learns from informative future features via GAN.

3.3.1 Generator and Future-aware Discriminator. Specifically, for a user u and a target item d, the ranking model should give a ranking score of the user-item pair. The common feature of the past and present is noted as f_c, while the future feature is represented as f_f. Precisely, we build the generator and discriminator as:

- **Generator.** The generator attempts to generate appropriate items according to the common features f_c. We use $p_\theta(d|u, f_c)$ to represent the probability of item d clicked by u given by the generator with parameter θ. The goal of the generator is to simulate the user-item click probability distribution of real data. The generator will be deployed for online recommendation.
- **Future-aware discriminator.** The future-aware discriminator aims to distinguish real clicked items from both unclicked items and generated "fake" clicked items, with the help of both common features and future features. We use $h_\phi(u, d|f_c, f_f)$ to represent the ranking score of item d for user u with parameter ϕ. Enhanced by the future information, the discriminator can fully stimulate the potential of the generator.

Through the adversarial minimax game, the generator and future-aware discriminator can improve each other. Note that we do not use future features in both offline training and online serving for the online deployed model (i.e., the generator). The formalization of $p_\theta(\cdot)$ and $h_\phi(\cdot)$ can be flexible according to the practical demands. We also give a specification of adopting AFE on a real-world list-wise RL-based recommendation model in Sec. 3.4.

Classical GAN-based recommendations such as IRGAN [24] encourage the generators to select high-quality negative samples via the minimax game, which aims to enhance the discriminator for recommendation inspired by the catfish effect. Different from these models, AFE builds a much stronger future-aware discriminator (which cannot be deployed in online) to improve the generator. It is because that in general GAN-based tasks, the discriminator is often (and should be) stronger than the generator for a stable training [6, 37]. However, if equipped with the future features, the generator is likely to have a much more powerful modeling ability than the discriminator, which may disturb the training of the discriminator. Hence, we encode future features into the discriminator, and deploy the generator in online serving.

3.3.2 Minimax Optimization Objective. Inspired by [24], we also train the generator and the discriminator via a minimax game. We use $D_{pos}(u)$ and $D_{neg}(u)$ to represent the clicked and unclicked item distributions of user $u \in U$ in the training data. Without loss of generality, the optimization objective is formalized as follows:

$$\begin{aligned} L^{G^*,D^*} = \min_\theta \max_\phi \sum_{u\in U} (&\mathbb{E}_{d\sim D_{pos}(u)}[\log(\sigma(h_\phi(u,d|f_c,f_f)))] \\ &+ \mathbb{E}_{d\sim D_{neg}(u)}[\log(1-\sigma(h_\phi(u,d|f_c,f_f)))] \\ &+ \mathbb{E}_{d\sim p_\theta(d|u,f_c)}[\log(1-\sigma(h_\phi(u,d|f_c,f_f)))]). \end{aligned} \tag{1}$$

σ is the sigmoid function. $d \sim p_\theta(d|u, f_c)$ indicates the items generated by the generator. We regard all generated d as the positive instance if $d \in D_{pos}(u)$. Note that the negative samples consist of both the generated "fake" clicked items and the real unclicked items. During the minimax game, the generator and discriminator are learned iteratively by maximizing (for discriminator) and minimizing (for generator) the objective in Eq. (1).

Specifically for the discriminator, the objective is to maximize the log-likelihood of correctly judging positive and negative instances in Eq. (1). It is convenient to adopt stochastic gradient descent (SGD) based updating to approximate the $\arg\max_\phi$. In contrast, for the generator, its objective is to minimize Eq. (1), leading the generated items to confuse the discriminator. Different from GANs in CV [8], the generator of AFE provides discrete items instead of continuous feature embeddings. Therefore, we adopt a policy gradient based RL method (i.e., REINFORCE [25]) for the generator's training, which is similar to [24]. Formally, the gradient $\nabla_\theta L^G(u)$ for is written as:

$$\nabla_\theta L^G(u) \simeq \frac{1}{K}\sum_{k=1}^{K} \nabla_\theta \log p_\theta(d_k|u,f_c)\log(1-\sigma(h_\phi(u,d_k|f_c,f_f))). \tag{2}$$

For each user, we conduct a sampling approximation to select top K items given by the generator, and feed them to the discriminator as negative samples. With the REINFORCE terminology, the term $\log p_\theta(d_k|u, f_c)$ is considered as the policy, while the term $\log(1 - \sigma(h_\phi(u, d_k|f_c, f_f)))$ is regarded as the reward. Through Eq. (2), the generator will be learned to give a higher generation probability for d_k that has a higher ranking score given by the discriminator (i.e., generating fake items that successfully confuse the discriminator). f_c and f_f are also updated via Eq. (1). Besides the adversarial loss, we also conduct a direct learning and a future-aware knowledge distillation for better training, which are shown in Sec. 3.5.

3.4 Specification on List-wise Recommendation

AFE is an effective and universal framework. However, it is non-trivial to adopt AFE on a real-world list-wise recommendation due to its practical complexity. In this subsection, we show the implementation details of conducting AFE on a list-wise RL-based model. It is also easy to adopt AFE on other point-wise recommendation models (e.g., AutoInt [20], see the offline evaluation in Sec. 5.4).

3.4.1 Overall List-wise RL Framework with AFE. List-wise recommendation aims to generate an item list for each request, which is widely adopted in practice [42]. Reinforcement learning (RL) is naturally suitable for considering long-term rewards in the list-wise prediction [32]. Similarly, we model the list-wise recommendation as a sequential item prediction task with a value-based RL model Double-DQN [23]. The key notions are defined as follows:

- **State** s_t: the t-th state in list-wise RL should not only model the user preferences and the current item's features, but also model the whole list's information. s_t is built from user profiles, user historical behaviors, recommendation contexts, the t-th item features, and previous $t-1$ item features in the list. Note that the list-wise RL discriminator further considers the future features as supplements in its state encoder.
- **Action** a_t: the t-th action a_t represents generating an item for the t-th position in the recommended list.
- **Reward** r_t: the t-th reward r_t is measured via the click number of the t-th item in the recommended list.

The list-wise RL model will sequentially recommend items having the highest Q values until the entire list is generated.

In AFE, we build two similar list-wise RL models as the generator and future-aware discriminator. At each position, the generator considers the common features as inputs, and generates the "fake" clicked items according to the Q value distribution of items. The discriminator, enhanced by future information, will distinguish the "fake" clicked items from real ones. Through the iterative adversarial training in Eq. (1), the generator is optimized to recommend high-quality items that can achieve high Q values from the discriminator.

3.4.2 Model Architecture. Inspired by [32], we take the previous $t-1$ item features $\{f_1, \cdots, f_{t-1}\}$ already recommended by the list-wise RL as the input sequence for the current prediction at the t-th position. In the generator, the i-th item feature embedding $\boldsymbol{f}_i^G$ is built from the interactions between multiple feature fields, including the user profile set U_p (e.g., age), the user historical behavior feature set U_c (e.g., clicks), the i-th recommendation contexts set C_i (e.g., position), and the i-th item profile set D_i (e.g., tag). First, we follow a recent powerful feature interaction modeling method AFN [4] to build the i-th item feature embedding $\boldsymbol{f}_i^G$ as follows:

$$\boldsymbol{f}_i^G = \mathrm{AFN}(\boldsymbol{U}_p, \boldsymbol{U}_c, \boldsymbol{C}_i, \boldsymbol{D}_i), \tag{3}$$

where AFN(·) indicates the Logarithmic transformation layer. Next, we conduct a GRU [12] to model the sequential information as:

$$s_t^G = \mathrm{MLP}(\mathrm{GRU}(\{f_1^G, \cdots, f_{t-1}^G\})). \quad (4)$$

s_t^G is the state embedding that well represents the t-th state in the list-wise RL. MLP(·) is a Multi-layer perceptron. Finally, we combine the state embedding with the target item embedding $\boldsymbol{d}_t$ (i.e., the action), and feed them into another MLP layer as follows:

$$q^G(s_t, a_t) = \mathrm{ReLU}(\mathrm{MLP}(\mathrm{Concat}(\boldsymbol{s}_t^G, \boldsymbol{d}_t))). \quad (5)$$

$q^G(s_t, a_t)$ is the predicted Q value given by the generator. In the list-wise recommendation, the Q value is defined as the expected return of the current and future items in the list. We have:

$$Q^G(s_t, a_t) = \mathbb{E}_{s_{t+1}, r_t \sim E}[r_t + \gamma Q^G(s_{t+1}, a_{t+1})]. \quad (6)$$

r_t is the reward calculated as the clicked number of the t-th item. γ is the discount factor that decreases the importances of low-rank items in list-wise recommendation. In training and online serving, the item d_t with the highest $q^G(s_t, a_t)$ is regarded as the t-th action. Fig. 3 shows the model architecture of the list-wise AFE.

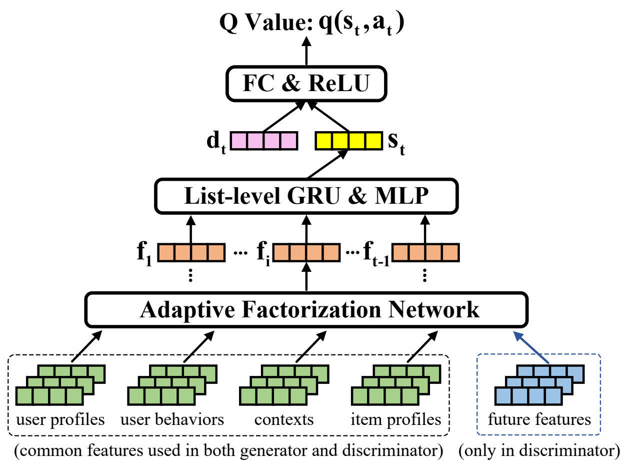

Figure 3: Model architecture of list-wise AFE.

For the future-aware discriminator, when building the i-th item feature embedding f_i^D, we further consider the i-th *future feature set* F_{f_i} built in Sec. 3.2 besides the common features. Formally, we calculate $\boldsymbol{f}_i^D$ encoded with future features $\boldsymbol{F}_{f_i}$ as follows:

$$f_i^D = \mathrm{AFN}(\boldsymbol{U}_p, \boldsymbol{U}_c, \boldsymbol{C}_i, \boldsymbol{D}_i, \boldsymbol{F}_{f_i}). \quad (7)$$

Next, we follow the same GRU and MLP structure in Eq. (4) to get $\boldsymbol{s}_t^D$, and follow Eq. (5) with different model parameters to generate the predicted Q value $q^D(s_t, a_t)$ of discriminator as follows:

$$q^D(s_t, a_t) = \mathrm{ReLU}(\mathrm{MLP}(\mathrm{Concat}(\boldsymbol{s}_t^D, \boldsymbol{d}_t'))). \quad (8)$$

Here, d_t' may come from both real impressed items and fake clicked items given by the generator. We also model $Q^D(s_t, a_t)$ with the TD error in Eq. (6) using the same γ and reward r_t.

3.5 Multi-factor Optimization

In real-world scenarios, the complicated list-wise RL models are challenging to have a stable convergence. The challenges even multiply when we enhance them with the GAN-based AFE framework. Hence, besides the adversarial loss in Sec. 3.3.2, we further bring two additional losses including a direct learning loss and a future-aware knowledge distillation loss for a stable and effective training. They can also be easily adopted on other AFE implementations.

3.5.1 Minimax adversarial loss. We follow Eq. (1) to play the minimax game. We set E_{real} as the real training dataset and $(s_t, a_t) \in E_{real}$ as the real state-action pair. We also use $G_{top_K}(s_t|\theta)$ to represent the top K "fake" clicked item set of state s_t given by the generator. The minimax adversarial loss $L^{G,D}$ is then defined as:

$$\begin{aligned} L^{G,D} = \min_\theta \max_\phi \sum_{u \in U} (&\mathbb{E}_{s_t, a_t \sim E_{real}}[(y_t - q^D(s_t, a_t|\phi))^2] \\ &+ \mathbb{E}_{a_t \sim G_{top_K}(s_t|\theta)}[\log(1 - \sigma(q^D(s_t, a_t|\phi) - b_q))]). \end{aligned} \quad (9)$$

Here, θ and ϕ are parameters in the generator and discriminator. $q^D(s_t, a_t|\phi)$ is the predicted Q value in Eq. (8), and b_q is a Q value bias. The first term of the loss indicates the classical mean squared error (MSE) for value-based RL training in the discriminator, which is updated via all list-wise training instances. We define the y_t as the t-th target Q value, which is formalized as:

$$y_t = r_t + \gamma q^D(s_{t+1}, \mathrm{argmax}_a q^D(s_{t+1}, a|\phi)|\bar{\phi}). \quad (10)$$

$\bar{\phi}$ is the target discriminator's parameters from Double-DQN [23]. r_t is the t-th reward. The second term in Eq. (9) aims to distinguish generated "fake" items from real clicked items and give them lower Q values. Note that if an generated item is clicked by user in E_{real}, it will be viewed as a positive instance and not used in $G_{top_K}(s_t|\theta)$.

3.5.2 Direct Learning Loss. Since we deploy the generator in online system, we conduct a direct RL-based training on the generator for direct optimization. Similar to the MSE loss of the discriminator in Eq. (9), we design the direct learning loss L^{MSE} as follows:

$$\begin{aligned} L^{MSE} &= \sum_{u \in U} (\mathbb{E}_{s_t, a_t \sim E_{real}}[(y_t' - q^G(s_t, a_t|\theta))^2] \\ y_t' &= r_t + \gamma q^G(s_{t+1}, \mathrm{argmax}_a q^G(s_{t+1}, a|\theta)|\bar{\theta}). \end{aligned} \quad (11)$$

$\bar{\theta}$ is the target generator's parameters. This loss enables the generator to directly learn from real positive and negative instances besides the minimax game with the discriminator, which improves both effectiveness and robustness of AFE's training.

3.5.3 Future-aware Knowledge Distillation Loss. In Sec. 3.3, the future information mainly influences the generator via the adversarial training. To improve the efficiency of future encoding, we further conduct the knowledge distillation (KD) [13] to transfer future information from the discriminator to the generator. Precisely, the discriminator and generator of AFE are also viewed as the teacher and student of KD respectively. We combine the MSE-based KD loss of q values with the Hint [19] of intermediate embeddings to calculate the future-aware KD loss L^{KD} as follows:

$$L^{KD} = \sum_{t=1}^{|E_{real}|} (||q^G(s_t, a_t|\theta) - q^D(s_t, a_t|\phi)||_2^2 + \lambda_h ||\boldsymbol{h}_i^G - \boldsymbol{h}_i^D||_2^2). \quad (12)$$

$\boldsymbol{h}_i^G$ and $\boldsymbol{h}_i^D$ are the intermediate embeddings after the penultimate MLP layer in Eq. (5) and Eq. (8) of the generator and discriminator. λ_h is a KD weight empirically set to be 1. The KD loss L^{KD} can be regarded as a KD-based future encoding, which works as a supplement to the adversarial future encoding in Sec. 3.3. It encourages the generator to provide similar q values and state embeddings as the discriminator's, which also indirectly teaches the generator to find hidden relations between historical and future information.

3.5.4 Overall Loss. Finally, we jointly train the discriminator and the generator under the weighted combination of all three losses above to form the final loss L as follows:

$$L = \lambda_1 L^{G,D} + \lambda_2 L^{MSE} + \lambda_3 L^{KD}. \tag{13}$$

$\lambda_1 : \lambda_2 : \lambda_3$ is empirically set as $1 : 4 : 1$, emphasizing the impacts of real feedbacks. The effectiveness of different components and losses are evaluated in Sec. 5.6. To demonstrate the universality of AFE, we further adopt AFE on a classical ranking model AutoInt [20] with the same multi-factor optimization in Eq. (13).

4 ONLINE DEPLOYMENT

We have deployed AFE on a real-world recommendation system named *WeChat Top Stories*, which is a list-wise recommendation widely used by millions of users. As stated in Sec. 3.4, the generator of AFE is deployed on the ranking module. Hence, the online efficiency of AFE is nearly equal to other list-wise RL models. It takes common features as inputs and outputs an item list containing 10 items. No future information is used in online. Following [29, 32], we also conduct the online exploration instead of simulators to collect real user feedbacks for RL exploration. Precisely, for a small subset of online traffic, AFE will randomly recommend one of the top-N best items rather than the best one as a recommendation trial. We implement AFE with Linux and GCC and conduct a distributed training, using 50 ps machines (2 core, 4G memory) and 150 worker machines (4 core, 8G memory). We spend nearly 5 hours for daily training, and the online computation cost is acceptable in list-wise recommendation. The training stability is confirmed in practice with the help of the direct learning loss and the KD loss.

5 EXPERIMENTS

In this section, we conduct sufficient experiments to answer the following research questions: (**RQ1**): How does AFE perform against the state-of-the-art baselines in offline evaluation (see Sec. 5.4)? (**RQ2**): How does AFE perform in real-world online recommendation with various practical metrics (see Sec. 5.5)? (**RQ3**): What are the effects of different components in AFE (see Sec. 5.6)?

5.1 Datasets

We evaluate AFE on two large-scale industrial datasets, including a list-wise *WTS-1B* dataset and a point-wise *Article-333M* dataset.

WTS-1B. We build a new dataset *WTS-1B* extracted from a widely-utilized recommendation system named WeChat Top Stories. This industrial dataset is used to evaluate AFE in the list-wise recommendation scenario. Precisely, we randomly sample nearly 38 million users and get their 1.3 billion behaviors of impression (including click and unclick) on nearly 14 million multi-type items. All impression behaviors are naturally arranged in recommendation lists and sessions, where each list contains 10 items. Note that the dataset also contains the user behaviors on items generated by the online exploration strategy, which are fairly used by all models.

Article-333M. To evaluate the point-wise recommendation, we resample 7.8 million article items from WeChat Top Stories, and collect all their 333 million behaviors of the same 38 million users to form the Article-333M dataset. This point-wise dataset directly flattens all user behaviors in lists, ignoring list-wise information. The rest features are the same as the WTS-1B dataset.

For two datasets, we consider the instances in the former few days as the train set, and the rest instances as the test set. The instance proportion of the train set and the test set is nearly 7 : 3. All data are preprocessed via data masking to protect user's privacy. Note that only behaviors in the train set are used as our future information to avoid the fatal data leakage issue. We conduct AFE on the list-wise RL-based model in Sec. 3.4 for the list-wise recommendation with WTS-1B, and implement AFE on AutoInt [20] for the point-wise recommendation with Article-333M. Table 1 shows the detailed statistics of two datasets.

Table 1: Statistics of list-wise and point-wise datasets.

Dataset	#user	#item	#instance
WTS-1B	37,725,789	13,752,412	1,329,663,154
Article-333M	37,725,789	7,752,187	332,716,677

5.2 Competitors

We implement several competitive methods as baselines, which are roughly grouped into three categories as follows. For fair comparisons, all models share the same common features and training instances as AFE. Future-aware models also have the same future features as AFE in both list-wise and point-wise scenarios.

Point-wise ranking models. We implement several classical ranking models as baselines for two datasets, which are widely verified in industry. These models include FM [18], Wide&Deep [3], NFM [11], AFM [27] and DeepFM [9]. We also implement two recent strong ranking models AutoInt [20] and AFN [4]. AutoInt uses self-attention to model feature field interactions. AFN is a recent SOTA ranking model using the logarithmic neural network.

List-wise RL-based models. For the list-wise recommendation, we implement two RL-based models for further comparisons. Specifically, we implement a Double DQN model [23] with the same model architecture used in Sec. 3.4 (i.e., the generator of AFE) without the adversarial framework and future information, noted as DDQN+. We also implement the current SOTA list-wise recommendation model HRL-Rec [32] for more challenging comparisons. HRL-Rec is specially designed for integrated recommendation, which contains a channel selector (for domain-level preferences) and an item recommender (for item-level preferences). The generator of AFE can be regarded as an enhanced version of the item recommender in HRL-Rec with AFN used as the feature extractor. All list-wise RL models use the same features and rewards as AFE for fair comparisons.

Future-aware models. Since no future information could be observed in online, we can only conduct indirect future-aware learning in offline. However, there is no existing work that considers future information in general recommendation. Inspired by PFD [33], we extend the idea of feature-based knowledge distillation to future encoding, replacing the privileged features with our future features. For fair comparisons, we use the same common and future features as AFE. We also conduct PFD on the same model architecture of AFE in two datasets. The enhanced PFD is noted as PFD (DDQN+), which can be viewed as an ablation version of AFE without GAN. Note that PFD (DDQN+) and AFE only differ in the usages of future information (i.e., knowledge distillation V.S. KD+GAN). We do not compare with MLM-enhanced sequence-based models [38], since they are limited to the specific future information of future behaviors and the task of sequence-based recommendation.

5.3 Experimental Settings

In AFE, the generator and discriminator share the same neural network and common features. We have 15 feature fields in common features and 8 feature fields in future features after field aggregation. The future information is built according to Sec. 3.2, and is only used in offline discriminator. The dimensions of feature fields are 16 and 8, and the dimensions of state embeddings are 160 and 120 for the discriminator and generator. We conduct 2-layer MLP layers for q value and state embedding calculations. The discount factor γ is set as 0.6 for future rewards. In training, we adopt Adam [14] for optimization with the batch size set as 256. The generator provides top 1 items for adversarial training. Similarly, AFE uses the same multi-factor losses and common/future features for AutoInt in the point-wise scenario. The dimensions of all models in the point-wise scenarios are also the same. We conduct a grid search for parameter selection. All models share the same common features (if beneficial) and experimental settings. We should highlight that all future information is not used in online serving.

5.4 Offline Evaluation (RQ1)

We first adopt AFE on both list-wise and point-wise recommendation models with two large-scale datasets.

5.4.1 Evaluation Protocols. Following [20, 32], we use the classical Area Under Curve (AUC) to measure the offline recommendation accuracy. We do not use Logloss since baselines and our model have different losses. We further calculate the RelaImpr [35] to reflect the relative improvements over the base model (i.e., FM for WTS-1B). We conduct 3 runs for models on both WTS-1B and Article-333M.

5.4.2 Experimental Results . Table 2 shows the results on WTS-1B, and Table 3 shows the results on Article-333M. We can find that:

(1) AFE achieves significant improvements over all baselines on both list-wise and point-wise recommendation tasks, with the significance level of AFE's improvements $p < 0.01$. Note that the current AUC improvements, 2.8% on AFN and 0.4% on PFD (DDQN+) (an ablation version of AFE without GAN), are significant in billion-level industrial datasets. AFE has significant improvements over AFN and HRL-Rec, which indicates that the future information is beneficial in recommendation. It also implies that our adversarial future encoding is effective in modeling future information to polish the generator's training compared to PFD (DDQN+).

(2) The improvements of AFE mainly derive from the following aspects: (a) armed with the future information, the future-aware discriminator becomes more powerful. Trained under the minimax game, the generator is encouraged to capture more informative relations between behaviors throughout the past and the future. Hence, both short-term item correlations and long-term interest evolutions are successfully extracted, which helps to generate better recommendations. (b) The future-aware knowledge distillation further provides an additional lesson directly from the future-aware teacher (i.e., the discriminator) as a supplement to the adversarial future encoding. The direct learning also enables a stable and fast convergence of AFE. Sec. 5.6 gives detailed analyses on the effectiveness of different model components.

Table 2: Results of offline evaluation on WTS-1B. * indicates that models have the same list-wise neural model as AFE's. The improvements are significant (t-test with $p < 0.01$).

Model	AUC	RelaImpr
FM (Rendle 2010)	0.7004	0.00%
Wide&Deep (Cheng et al. 2016)	0.7101	4.84%
NFM (He and Chua 2017)	0.7065	3.04%
AFM (Xiao et al. 2017)	0.7077	3.64%
DeepFM (Guo et al. 2017)	0.7110	5.29%
AutoInt (Song et al. 2019)	0.7160	7.78%
AFN (Cheng et al. 2020)	0.7177	8.63%
DDQN+ (i.e., generator in AFE)*	0.7365	18.01%
HRL-Rec (Xie et al. 2021c)*	0.7393	19.41%
PFD (DDQN+)*	0.7425	21.01%
AFE (ours)*	**0.7461**	**22.80%**

(3) PFD (DDQN+) achieves the second-best performance but is still worse than AFE. It represents that the GAN-based training can further enhance the efficiency of future knowledge transfer in future encoding on the basis of the future-aware KD. We should highlight that: (a) the original PFD does not use the future information. We add the future-aware knowledge distillation loss in Eq. (12) and the direct learning loss in Eq. (11) to build the PFD (DDQN+). (b) For fair comparisons, we also enhance PFD with AFE's list-wise model in Sec. 3.5. PFD (DDQN+) can be viewed as an ablation version of AFE without the future-aware adversarial loss $L^{G,D}$ in Eq. (9).

(4) AFE achieves consistent improvements on both the point-wise AutoInt model and list-wise RL-based model, which demonstrates the flexibility and universality of our AFE framework. It is convenient to adopt AFE as a plug-and-play module on various types of ranking models. The flexible and universal designs of the feature-based GAN and the future information also make AFE be able to smoothly adapt to different practical scenarios.

Table 3: Results on Article-333M. We only display the representative results of competitive point-wise models in Table 2. The improvements are significant (t-test with $p < 0.01$).

Model	AUC	RelaImpr
AutoInt (Song et al. 2019)*	0.7842	0.00%
PFD (DDQN+)*	0.7894	1.83%
AFE (ours)*	**0.7933**	**3.20%**

5.5 Online A/B Tests (RQ2)

Besides the offline evaluation, we also evaluate AFE with an online A/B test on a well-known recommendation system to demonstrate its effectiveness in real-world scenarios.

5.5.1 Evaluation Protocols. Following Sec. 4, we deploy the generator of AFE on the ranking module of WeChat Top Stories with other modules unchanged. The compared online list-wise recommendation baseline is an enhanced version of HRL-Rec [32], which mainly replaces the AutoInt feature interaction layer [20] with the AFN layer [4], and is also armed with the knowledge distillation objective. This enhanced HRL-Rec model has been verified online for a long time, which is a competitive baseline. In online evaluation, we focus on two evaluation metrics, including (1) the average click number per capita (ACN) [32], and (2) the average dwell time per capita (DT) [36], to model recommendation accuracy and user satisfaction. We conduct the online A/B test for 24 days, affecting nearly 6 million users. We show the improvement percentages of AFE over the strong base model in online evaluation.

Table 4: Online A/B tests on WeChat Top Stories.

Metrics	Average click number	Average dwell time
AFE	+1.63%	+0.47%

5.5.2 Experimental Results. From Table 4 we can find that:

(1) AFE achieves significant improvements on both ACN and DT with the significance level $\alpha = 0.01$. We should highlight that the 1.63% ACN improvement is encouraging in real-world recommendation with complicated models and feature engineering. Note that the main differences between AFE and the online base model only exist in the usage of future information and the GAN framework for future encoding. It reconfirms the importance of considering future information in model training, which is beneficial to learn a better generator that can even confuse the recommendation prophet. It also implies that our AFE framework is capable of modeling future information to improve models in real-world scenarios.

(2) The average click number (ACN) is one of the most essential metrics to evaluate online models. It reflects both recommendation accuracy and user activeness in online systems [32]. A higher ACN indicates that users will browse and click more items, which usually implies that users are more satisfied with the online system.

(3) The improvements on the average dwell time (DT) further demonstrate that the recommended lists are accurate and high-quality. Differing from CTR that is directly calculated by user clicks, dwell time is measured by the time users spend on items, which can alleviate the noises from clickbaits and thus reflects users' real satisfaction. Other online metrics of AFE such as CTR and diversity are comparable or not significantly improved.

5.6 Ablation Tests (RQ3)

We further conduct several ablation tests to verify that different components are indispensable in AFE. Table 5 displays the results of 7 ablation versions of AFE on WTS-1B. We can observe that:

(1) We first remove all future features in the discriminator to disable the future information in both GAN and KD modules (note that the GAN and KD modules still remain in this ablation setting). The substantial reduction in AUC verifies the effectiveness of future features. It also indicates that the future information can be indirectly transferred into the generator via GAN and KD.

Table 5: Ablation tests on WTS-1B.

Ablation settings	AUC	RelaImpr
AFE	**0.7461**	**22.80%**
– future information	0.7377	18.61%
– GAN loss $L^{G,D}$	0.7425	21.01%
– KD losses L^{KD}	0.7385	19.01%
– Direct learning loss L^{MSE}	0.6995	-0.45%
– GRU	0.7351	17.32%
– AFN	0.7404	19.96%
– future rewards ($\gamma = 0$)	0.7384	18.96%

(2) In the second block, we evaluate the importances of three losses in the multi-factor optimization. We remove the GAN framework to test $L^{G,D}$ (which is equal to the PFD (DDQN+) model in Table 2), and remove the future-aware KD to test L^{KD}. The results show that both GAN and KD are beneficial in future encoding and model training. These two methods also enable a fast and stable model convergence in AFE training with complicated list-wise RL models. We have observed the convergence curves of the generator and discriminator, and find that both of them are optimized smoothly. We also find that the direct learning L^{MSE} is indispensable in AFE (which is a basic loss in ranking). It is natural since the generator should be directly optimized by real feedbacks. All three losses are combined to get the best performances.

(3) For the list-wise implementation of AFE, We also evaluate the effectiveness of different components in the model architecture. We replace the GRU and AFN layers with average pooling layers respectively. The results confirm that GRU and AFN are beneficial. Moreover, we also set the discount factor $\gamma = 0$, which indicates that AFE does not consider any future reward in list-wise scenarios. The result reflects the effectiveness of modeling future rewards with RL-based models in list-wise recommendation.

6 CONCLUSION AND FUTURE WORK

In this work, we propose a novel adversarial future encoding to make full use of future information in general recommendation under a GAN framework. The future-aware discriminator works as a recommendation prophet that "peeps into the future", which encourages the generator to learn better. We have successfully adopted AFE on both a list-wise RL-based model and a point-wise ranking model. Both AFE models achieve significant improvements in two large-scale industrial datasets. Currently, we have deployed AFE on a real-world system, affecting millions of users.

We are convinced that future information is an essential source of user preferences. In the future, we will explore more sophisticated RL models and neural architectures for a better combination of common features and future features. We will also try to jointly model future information via adversarial training and pre-training tasks, and verify our AFE on other models, domains, and tasks.

REFERENCES

[1] Pedro G Campos, Fernando Díez, and Iván Cantador. 2014. Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. *User Modeling and User-Adapted Interaction* (2014).

[2] Xusong Chen, Dong Liu, Chenyi Lei, Rui Li, Zheng-Jun Zha, and Zhiwei Xiong. 2019. BERT4SessRec: Content-Based Video Relevance Prediction with Bidirectional Encoder Representations from Transformer. In *Proceedings of MM.*

[3] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the DLRS workshop.*

[4] Weiyu Cheng, Yanyan Shen, and Linpeng Huang. 2020. Adaptive Factorization Network: Learning Adaptive-Order Feature Interactions. In *Proceedings of AAAI.*

[5] Konstantina Christakopoulou and Arindam Banerjee. 2019. Adversarial attacks on an oblivious recommender. In *Proceedings of RecSys.*

[6] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. In *Proceedings of ICLR.*

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL.*

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of NIPS.*

[9] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. In *Proceedings of IJCAI.*

[10] Xiaobo Hao, Yudan Liu, Ruobing Xie, Kaikai Ge, Linyao Tang, Xu Zhang, and Leyu Lin. 2021. Adversarial Feature Translation for Multi-domain Recommendation. In *Proceedings of KDD.*

[11] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of SIGIR.*

[12] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In *Proceedings of ICLR.*

[13] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2014. Distilling the knowledge in a neural network. In *Proceedings of NIPS.*

[14] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR.*

[15] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* (2009).

[16] Bin Liu, Chenxu Zhu, Guilin Li, Weinan Zhang, Jincai Lai, Ruiming Tang, Xiuqiang He, Zhenguo Li, and Yong Yu. 2020. AutoFIS: Automatic Feature Interaction Selection in Factorization Models for Click-Through Rate Prediction. In *Proceedings of KDD.*

[17] Yujie Lu, Shengyu Zhang, Yingxuan Huang, Luyao Wang, Xinyao Yu, Zhou Zhao, and Fei Wu. 2020. Future-Aware Diverse Trends Framework for Recommendation. *arXiv preprint arXiv:2011.00422* (2020).

[18] Steffen Rendle. 2010. Factorization machines. In *Proceedings of ICDM.*

[19] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2015. Fitnets: Hints for thin deep nets. In *Proceedings of ICLR.*

[20] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. 2019. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of CIKM.*

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

[22] Zhu Sun, Di Yu, Hui Fang, Jie Yang, Xinghua Qu, Jie Zhang, and Cong Geng. 2020. Are we evaluating rigorously? benchmarking recommendation for reproducible evaluation and fair comparison. In *Proceedings of RecSys.*

[23] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of AAAI.*

[24] 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 SIGIR.*

[25] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* (1992).

[26] Chaojun Xiao, Ruobing Xie, Yuan Yao, Zhiyuan Liu, Maosong Sun, Xu Zhang, and Leyu Lin. 2021. UPRec: User-Aware Pre-training for Recommender Systems. *arXiv preprint arXiv:2102.10989* (2021).

[27] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional factorization machines: Learning the weight of feature interactions via attention networks. In *Proceedings of IJCAI.*

[28] Ruobing Xie, Cheng Ling, Yalong Wang, Rui Wang, Feng Xia, and Leyu Lin. 2020. Deep Feedback Network for Recommendation. In *Proceedings of IJCAI.*

[29] Ruobing Xie, Yanlei Liu, Shaoliang Zhang, Rui Wang, Feng Xia, and Leyu Lin. 2021. Personalized Approximate Pareto-Efficient Recommendation. In *Proceedings of WWW.*

[30] Ruobing Xie, Zhijie Qiu, Jun Rao, Yi Liu, Bo Zhang, and Leyu Lin. 2020. Internal and Contextual Attention Network for Cold-start Multi-channel Matching in Recommendation.. In *Proceedings of IJCAI.*

[31] Ruobing Xie, Shaoliang Zhang, Rui Wang, Feng Xia, and Leyu Lin. 2021. Explore, Filter and Distill: Distilled Reinforcement Learning in Recommendation. In *Proceedings of CIKM.*

[32] Ruobing Xie, Shaoliang Zhang, Rui Wang, Feng Xia, and Leyu Lin. 2021. Hierarchical Reinforcement Learning for Integrated Recommendation. In *Proceedings of AAAI.*

[33] Chen Xu, Quan Li, Junfeng Ge, Jinyang Gao, Xiaoyong Yang, Changhua Pei, Fei Sun, Jian Wu, Hanxiao Sun, and Wenwu Ou. 2020. Privileged Features Distillation at Taobao Recommendations. In *Proceedings of KDD.*

[34] Feng Xue, Xiangnan He, Xiang Wang, Jiandong Xu, Kai Liu, and Richang Hong. 2019. Deep item-based collaborative filtering for top-n recommendation. *TOIS* (2019).

[35] Ling Yan, Wu-Jun Li, Gui-Rong Xue, and Dingyi Han. 2014. Coupled group lasso for web-scale ctr prediction in display advertising. In *Proceedings of ICML.*

[36] Xing Yi, Liangjie Hong, Erheng Zhong, Nanthan Nan Liu, and Suju Rajan. 2014. Beyond clicks: dwell time for personalization. In *Proceedings of RecSys.*

[37] Junliang Yu, Hongzhi Yin, Jundong Li, Min Gao, Zi Huang, and Lizhen Cui. 2020. Enhance Social Recommendation with Adversarial Graph Convolutional Networks. *TKDE* (2020).

[38] Fajie Yuan, Xiangnan He, Haochuan Jiang, Guibing Guo, Jian Xiong, Zhezhao Xu, and Yilin Xiong. 2020. Future Data Helps Training: Modeling Future Contexts for Session-based Recommendation. In *Proceedings of WWW.*

[39] Fajie Yuan, Xiangnan He, Alexandros Karatzoglou, and Liguang Zhang. 2020. Parameter-efficient transfer from sequential behaviors for user modeling and recommendation. In *Proceedings of SIGIR.*

[40] Zheni Zeng, Chaojun Xiao, Yuan Yao, Ruobing Xie, Zhiyuan Liu, Fen Lin, Leyu Lin, and Maosong Sun. 2021. Knowledge transfer via pre-training for recommendation: A review and prospect. *Frontiers in big Data* (2021).

[41] Dongyang Zhao, Liang Zhang, Bo Zhang, Lizhou Zheng, Yongjun Bao, and Weipeng Yan. 2020. MaHRL: Multi-goals Abstraction Based Deep Hierarchical Reinforcement Learning for Recommendations. In *Proceedings of SIGIR.*

[42] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. 2018. Deep reinforcement learning for page-wise recommendations. In *Proceedings of RecSys.*