On the User Behavior Leakage from Recommender System Exposure

XIN XIN*, Shandong University, China
JIYUAN YANG*, Shandong University, China
HANBING WANG, Shandong University, China
JUN MA, Shandong University, China
PENGJIE REN, Shandong University, China
HENGLIANG LUO, Meituan Inc., China
XINLEI SHI, Meituan Inc., China
ZHUMIN CHEN, Shandong University, China
ZHAOCHUN REN†, Shandong University, China

Modern recommender systems are trained to predict users potential future interactions from users historical behavior data. During the interaction process, despite the data coming from the user side recommender systems also generate exposure data to provide users with personalized recommendation slates. Compared with the sparse user behavior data, the system exposure data is much larger in volume since only very few exposed items would be clicked by the user. Besides, the users historical behavior data is privacy sensitive and is commonly protected with careful access authorization. However, the large volume of recommender exposure data which is generated by the service provider itself usually receives less attention and could be accessed within a relatively larger scope of various information seekers or even potential adversaries.

In this paper, we investigate the problem of user behavior leakage in the field of recommender systems. We show that the privacy sensitive user past behavior data can be inferred through the modeling of system exposure. In other words, *one can infer which items the **user** have clicked just from the observation of current **system** exposure for this user.* Given the fact that system exposure data could be widely accessed from a relatively larger scope, we believe that the user past behavior privacy has a high risk of leakage in recommender systems. More precisely, we conduct an attack model whose input is the current recommended item slate (i.e., system exposure) for the user while the output is the user's historical behavior. Specifically, we exploit an encoder-decoder structure to construct the attack model and apply different encoding and decoding strategies to verify the attack performance. Experimental results on two real-world datasets indicate a great danger of user behavior leakage. To address the risk, we propose a two-stage privacy-protection mechanism which firstly selects a subset of items from the exposure slate and then replaces the selected items with uniform or popularity-based exposure. Experimental evaluation reveals a trade-off effect between the recommendation accuracy and the privacy disclosure risk, which is an interesting and important topic for privacy concerns in recommender systems.

*Both authors contributed equally to this research.
†Corresponding author.

Authors' addresses: Xin Xin, xinxin@sdu.edu.cn, Shandong University, Qingdao, China; Jiyuan Yang, jiyuan.yang@mail.sdu.edu.cn, Shandong University, Qingdao, China; Hanbing Wang, hanbing.wang@mail.sdu.edu.cn, Shandong University, Qingdao, China; Jun Ma, majun@sdu.edu.cn, Shandong University, Qingdao, China; Pengjie Ren, renpengjie@sdu.edu.cn, Shandong University, Qingdao, China; Hengliang Luo, luohengliang@meituan.com, Meituan Inc., Beijing, China; Xinlei Shi, shixinlei@meituan.com, Meituan Inc., Beijing, China; Zhumin Chen, chenzhumin@sdu.edu.cn, Shandong University, Qingdao, China; Zhaochun Ren, zhaochun.ren@sdu.edu.cn, Shandong University, Qingdao, China.

1046-8188/2022/10-ART $15.00
https://doi.org/10.1145/3568954

CCS Concepts: • **Information systems** → **Recommender systems**; **Information extraction**; • **Security and privacy** → **Privacy protections**.

Additional Key Words and Phrases: Recommender System, Privacy Leakage, Privacy Protection, Information Security

1 INTRODUCTION

Recommender systems have been widely applied in various online web services, e.g., online shopping [23], video or music platforms [63], news recommendation [42], etc., to provide users with the most interesting items. Generally speaking, a recommender system targets on predicting the ***user's*** future behavior based on the ***user's*** historical interactions. Then the recommendation list is generated by selecting the most relevant items according to the predicted user behavior. The user historical behavior data is the keystone to conduct effective recommendation. Meanwhile, it is also highly privacy sensitive since various user profile information such as gender, age, and even political orientation could be inferred from the user behavior data [62]. As a result, the user behavior data is protected with strict department-specific access authorization or regulations such as the General Data Protection Regulation (GDPR) [39].

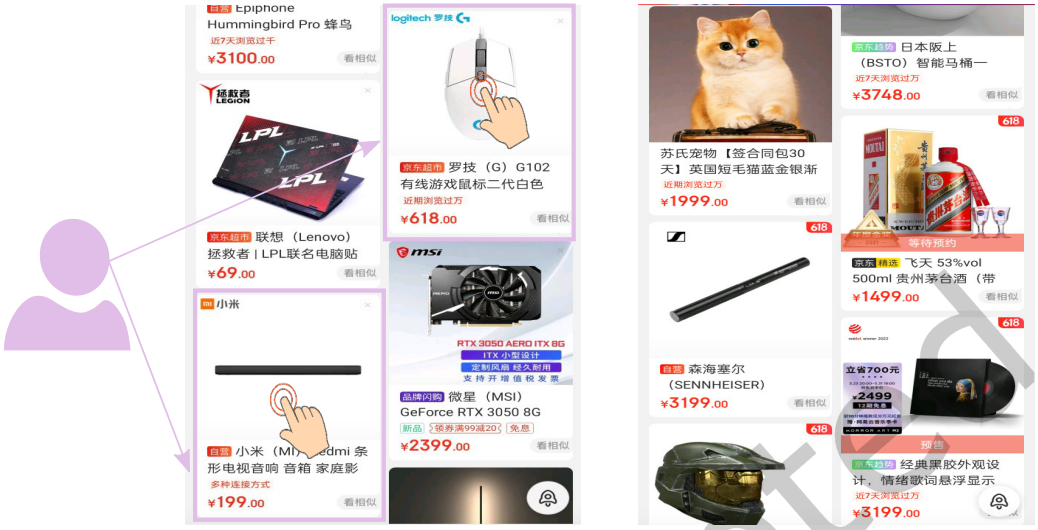
During the interaction process between users and recommenders, despite the data coming from the user feedback, the system also generates personalized slates of items which are pushed to the user as the recommendation service. Such a kind of ***system*** behavior data is also known as recommender system exposure. Fig. 1 gives illustrative examples of the exposure data. Compared with the sparse user behavior data, the system exposure data is much larger in volume since only very few exposed items would be clicked by the user. Besides, although service provides could adopt different strategies to infuse items or advertisements into the exposed list, the system exposure data still reflects users historical behavior patterns in a large degree. However, the large volume of system exposure receives much less attention compared with user behavior data, and could usually be accessed in a relatively larger scope which contains various information seekers or even adversaries [49]. For example, different departments may share the system behavior logs to perform cross-domain collaboration. [25] shows that there exists security concerns regarding the protection of recommender exposure data.

In this paper, we investigate the problem of user behavior leakage in the field of recommender systems. We aim to answer the following questions:

(1) Is there a risk that the ***user*** historical behavior privacy can be inferred from the ***system*** behavior (i.e., recommender exposure) data?
(2) If there is the risk, how to conduct the attack model?
(3) How to design a protection mechanism to downgrade the user behavior leakage risk?

There are plenty of previous works focusing on the privacy concern in recommender systems. For example, [14, 30] use cryptography to mask the user profile and behavior data. [14, 30] introduce differential privacy [13, 47, 66] to prevent user profile leakage. [34, 42, 53, 56, 61, 65] utilize federated learning to perform local computing on edge devices. [2, 9] utilize adversarial learning to promote recommendation models security. [64] investigates the membership inference attack against recommender systems. However, such existing works only focus on the protection of ***user*** data. While in this paper, we target on a new privacy preserving scenario, which is how the system behavior data exposes user privacy. Such a scenario poses a new attack and protection view from the ***recommender*** perspective.

More precisely, we propose a framework containing a black-box attack model [48] to infer user past behavior privacy from system exposure; and a protection mechanism to downgrade the privacy leakage risk. Here the black-box model means that we don't need to know or explicitly model the recommendation algorithm. For the attack model, we adopt an encoder-decoder architecture. The input of the attack model is the exposed items from the system over a period. Note that the input data just comes from the recommender perspective which means that we don't need to know which items in the exposed list are interacted by the user. Then the encoder



(a) (b)

Fig. 1. Two illustrative examples of recommender exposure data. (a) shows that the user interacts with two items in the exposed item slate while (b) shows that there is no observed interactions between the exposed items and the user.

is utilized to map the input system behavior data to a latent representation. Here we utilize three different encoding strategies including mean pooling, max pooling and self-attention [52] based encoding[1]. Based on the latent representation of system behavior, we propose two decoding methods to infer the privacy of user past behavior, namely point-wise decoding and sequence-wise decoding. Point-wise decoding treats the inference as a multi-label classification task with each past interaction as a label. Sequence-wise decoding further considers the sequential order of user past behavior. We utilized three different sequential models to conduct sequence-wise decoding, including Gated Recurrent Units (GRU), Long-Short Term Memory Network (LSTM), and attention-based transformer decoder. We conduct experiments on two real-world datasets and empirical results indicate a great danger of user behavior leakage. In other words, *potential adversaries can infer which items the user have clicked before just from the observation of current system exposure for this user.* To alleviate the leakage risk, we propose a two-stage protection mechanism which first selects a subset of items from the system exposure and then replaces the selected items with uniform or popularity-based exposure. Experimental evaluation reveals a trade-off effect between the recommendation accuracy and the privacy disclosure risk. We hope this work can raise more community concern regarding the protection of system data other than just focusing on the protection from the user perspective. To summarize, the main contributions of this work are:

- We point out a new privacy disclosure risk in recommender systems which is that the user historical behavior privacy could be inferred from the system exposure.
- We propose an attack model to perform user privacy inference. Experimental results on two real-world datasets indicate a great danger of privacy leakage.
- We propose a protection mechanism to alleviate the privacy risk. Empirical evaluation reveals a trade-off effect between the recommendation accuracy and the privacy leakage risk.

[1]In this paper, we utilize three simple encoding strategies to further demonstrate the risk of privacy leakage since the attack can be performed without trivial design and complex computation. We leave more advanced encoding methods for the attack model as one of our future works.

2 RELATED WORK

In this section, we provide a literature review of the research about recommendation, the research about exposure data, and the privacy concern in recommender systems.

2.1 Recommendation

A recommender system is an information filtering system, targeting on predicting user future preference based on historical user interactions with the system [64]. Collaborative filtering is one of the most successful recommendation methods, which is based on the similarity of users or items [32]. The key idea of collaborative filtering is that the user preference of an item can be predicted from similar user preference over similar items [32]. Typical collaborative filtering methods includes matrix factorization-based methods [27, 32, 38] or neighbourhood-based methods [46]. Besides, content-based recommendation [40] which utilizes the metadata of users and items (e.g., descriptions of item attributes and user profiles) to generate recommendation is also applied widely.

Recently, deep learning-based recommendation methods have become a hot research topic. [20] proposed neural collaborative filtering, which utilizes a multi-layer preceptron (MLP) to model high-order user-item interaction signals. Plenty of research has been proposed in this research line, such as NFM [18], CDAE[58] and Wide&Deep [7]. The key idea is to use deep learning to increase model expressiveness. Besides, graph neural networks also demonstrated their capability in recommendation since the user-item interactions can be naturally represented by interaction graphs. Plenty of graph recommendation models have emerged, such as HOP-Rec [60], NGCF [54] and LightGCN [19]. There are also other recommendation methods such as casual inference-based methods and reinforcement learning-based methods [6, 59, 67]. We don't elaborate them all since this work focuses on the attack and protection of user past behavior privacy, other than specific recommendation models.

Another closely related sub-field is sequential recommendation, which aims to predict the user's next interesting item from previous interacted items (in an interaction session). Early works for sequential recommendation are based on Markov Chain [43] and factorization methods [22]. Recently, plenty of deep learning-based recommendation models also emerged, including recurrent neural network (RNN)-based methods [21], convolutional neural network (CNN)-based methods [63], and the renowned attention-based methods [28, 50]. The main difference between our attack task and sequential recommendation lies in the following folds:

- The input of our attack model is the *current system* exposure data while the input for sequential recommendation is the *previous user* interactions.
- The output of our attack model is the *past user* behavior while the output of sequential recommendation is the predicted *user future* behaviours.
- The system usually exposes a slate of items simultaneously which means that the input for our attack model could have no strong sequential signals.

2.2 Recommender Exposure

Compared with the sparse user feedback data, the large volume of exposure data generated by the recommender receives relatively less research attention. The exposed items are selected by the recommender and thus can be regarded as a kind of system behavior data. Existing research regarding the exposure data mainly focuses on negative sampling from exposed items or exposure debiasing [5].

To perform effective item ranking, negative training instances are necessary to provide comparison signals. To this end, the recommender exposure data contains rich information about the negative preference of users since only very few exposed items would be interacted by the user. [10] proposed to use the exposed but not interacted items as negative examples. [12, 55] proposed reinforced negative samplers which can generate exposure-alike negative instances instead of directly choosing from exposure data. Generally speaking, the exposed but not interacted items can be seen as a kind of hard negative examples which can help to boost the ranking performance.

However, [11, 33] verified that both easy negative examples and hard negative examples are important for the model training. Negative sampling based on only exposure data would also downgrade the model performance. The mixture of exposed items and uniformly sampled items could be a good negative sampling strategy.

Besides, exposure bias also known as "previous model bias" [35] since the exposure data generation is affected by previous recommendation policies, is one of the biggest sources of bias in recommendation [35]. Exposure bias happens since only parts of specific items are exposed by the system so non-exposed and non-interacted items do not always represent user negative preference. More specifically, an unobserved interaction could only be attributed to the user unawareness of the item because the item is not exposed to the user. Therefore, regarding non-interacted items as negative samples could lead to misunderstanding of user true preference and sub-optimal performance. There are works focusing on alleviate the effect of exposure bias. [44] proposed to use the missing-not-at-random assumption to perform debiasing. [26] proposed an exposure-based propensity matrix factorization framework to counteract the exposure bias. [4] constructed a general debiasing framework and proposed an automatic debiasing method for recommendation system based on meta learning. How to model the exposure data distribution and conduct exposure debiasing has become an emerging hot topic.

To conclude, existing research regarding recommender exposure mainly focuses on negative sampling and debiasing. However, this work focuses on a new perspective which is how the system exposure data leaks user behavior privacy.

2.3 Privacy Security in Recommender Systems

Recent research shows that various user sensitive information such as age, gender, occupation, and even political orientation can be inferred from user-item interactions [62]. As a result, plenty of laws and regulations regarding the privacy protection has been established for online web services, especially for recommender systems [39]. Privacy security has obtained rapidly growing research interests in both academia and industry.

Existing privacy preserving recommendation methods can be categorized into cryptography-based methods, differential privacy-based methods, and federated learning-based methods. The cryptography methods attempt to mask the user data, for example through fully homomorphic encryption [30]. [14] presented a privacy recommendation algorithm in which the data on the server are encrypted and embedded to reduce the readability of the data. Differential privacy-based methods aim to introduce random perturbations as the noisy signal to the user data [13, 47]. For example, [47] developed matrix factorization algorithms under local differential privacy. [66] proposed differential private graph convolutional networks to protect users' sensitive data against the attribute inference attack and provide high-quality recommendation at the same time. [9] utilized gender obfuscation for user profiles to protect user privacy. [2] conducted privacy-aware recommendation with adversarial training which aims to both protect users against attribute inference attacks and preserve the quality of recommendation. In recent years, federated learning-based methods have become a hot research topic with the prevalence of more and more edge devices with computation capacity. The key idea of federated learning-based methods is the distributed model training [61]. The original data is not transferred between the server and the device to avoid leakage. The transferred communication message is based on model gradients [1, 61]. However, the model gradients can also be utilized to infer the original data. As a result, [3, 34, 36] proposed to combine federated learning with cryptography methods or differential privacy. There is also research focusing on improving the efficiency of federated learning-based recommender systems [29, 37]. Besides, federated learning has also been applied to domain-specific recommendation tasks, such as news recommendation [42], graph-based recommendation [56] and sequential recommendation [15].

Furthermore, [64] conducted the research focusing on the membership inference attack against recommender systems. Their attack targets on determining whether a user's behavior data is used by the target recommender and then they proposed a simple yet effective protection mechanism to address the privacy concern.

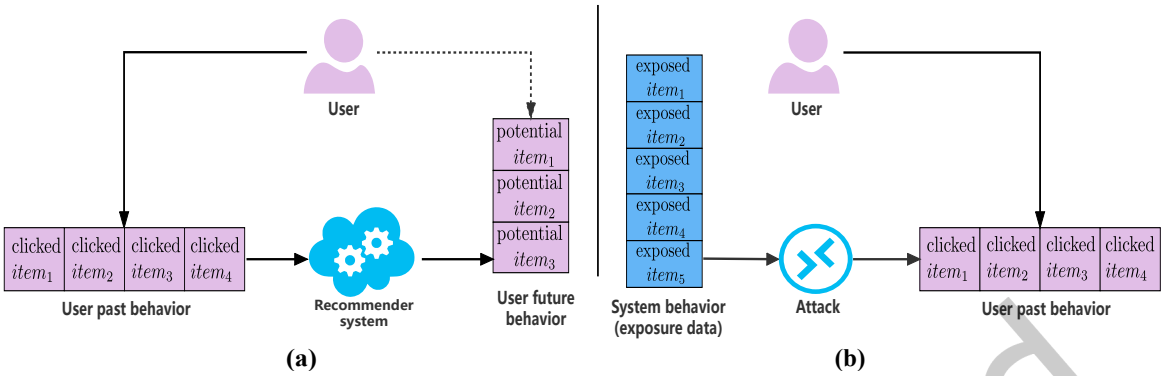


Fig. 2. A recommender system (a) aims to predict the user's future behavior from the historical user interactions. In this paper, the attack scenario (b) focuses on inferring the privacy of user's past behavior from the system behavior data.

To conclude, the above existing privacy preserving methods in recommender systems almost focus on the protection of user behavior data. However, the system behavior data (i.e., the system exposure data) is less explored for the privacy concern. Due to the fact that the interactions between users and recommenders naturally form a closed loop, we argue that the system behavior data should also be investigated to protect the user privacy, especially when there is data sharing between different recommender systems. To the best of our knowledge, our work is the first attempt which raises the privacy concern from the system perspective other than the commonly investigated user perspective.

3 THE ATTACK MODEL

In this section, we first present notations and the task formulation of the user behavior privacy attack in this work. Then we describe the encoding and decoding detail of the attack model. The task of user behavior privacy attack is to infer user historical behavior given system exposure data. In this paper, we exploit a simple encoder-decoder architecture to further demonstrate the risk of privacy leakage since the attack can be performed without trivial design and complex computation.

3.1 Notations and the Attack Task Formulation

Let $\mathcal{U}$ and $\mathcal{I}$ denote the user set and the item set, respectively. During the service of a recommender system, the user u produces a series of user behaviors (e.g., view, clicks, or purchases) based on the items exposed by the system. We use B^u=$\{B_1^u, B_2^u, ..., B_{|B^u|}^u\}$ denotes the set of user behavior sequences of user u, where B_i^u=$(b_1, b_2, ..., b_M)$ with $b_j \in \mathcal{I}$ is a specific user behavior sequence. This is regarded as the user privacy in this work. The set of recommended item slates for user u is represented by E^u=$\{E_1^u, E_2^u, ..., E_{|E^u|}^u\}$, where E_i^u=$(e_1, e_2, ..., e_N)$ with $e_j \in \mathcal{I}$ denotes a specific exposed item slate. This is the input data for the attack model of this work. As shown in Fig. 2(b), the task of the attack model is to infer B_i^u from the observation of E_i^u, which can be formulated as the estimation of

$$p(b_1, b_2, ..., b_M | e_1, e_2, ..., e_N). \tag{1}$$

The common sequential recommendation task which aims to predict the next interesting item from previous user interactions can be formulated as the estimation of $p(b_{t+1}|b_1, b_2, ..., b_t)$ as shown in Fig. 2(a). We can see that there are substantial differences between our attack task and the sequential recommendation task. The input and output of our attack task are from different information sources, which are systems and users, respectively. While

Table 1. The glossary table.

Notations	Description
$\mathcal{U}, \mathcal{I}$	the user and item set
B^u	the user behavior sequence set of user u, B^u=$\{B_1^u, B_2^u, ..., B_{\|B^u\|}^u\}$
B_i^u	a specific user behavior sequence of user u, B_i^u=$(b_1, b_2, ..., b_M)$ with $b_j \in \mathcal{I}$
E^u	the set of exposed item slates for user u, E_u=$\{E_1^u, E_2^u, ..., E_{\|E^u\|}^u\}$
E_i^u	a specific exposed item slate for user u, E_i^u=$(e_1, e_2, ..., e_N)$ with $e_j \in \mathcal{I}$
M	the length of user behavior sequence
N	the size of exposed item slate
d	the item embedding size

for sequential recommendation, the input and output only focus on the user perspective. Besides, as the input of our attack task, the system exposure data E_i^u just comes from the system side, which means that the attack model doesn't know which items in E_i^u were clicked by the user. Generally speaking, only very few of items in E_i^u could be interacted by the user. Finally, the recommender usually exposes a slate of items simultaneously which means that the items in E_i^u could have no strong sequential orders.

Table 1 summarizes the important notations used in this paper.

3.2 Privacy Attack

To perform the attack task, we exploit an encoder-decoder architecture. The encoder aims to convert the input of current system exposure E_i^u into a latent representation. Then the decoder attempt to infer the past user behavior privacy B_i^u from the encoded representation. Fig. 3 illustrates the overall architecture of the attack model.

3.2.1 ***Encoder***. Due to the fact that the recommender system usually exposes a slate of items simultaneously, so we don't consider the sequential order of items in E_i^u. In this work, we utilize three simple encoding strategies including mean pooling, max pooling and self-attention based encoding to further demonstrate the risk of privacy leakage since the attack can be conducted without trivial design and complex computation. We leave more advanced encoding methods for the attack model as one of our future works.

Mean pooling-based encoding. The pooling technique has been widely used in the field of computer vision to produce a summary statistic of the input and reduce the spatial dimension. Given the input system exposure $E_i^u = (e_1, e_2, ...e_N)$, we first embed each item e_j into a dense representation $\mathbf{e}_j \in \mathbb{R}^d$ where d denotes the embedding size. This can be done through a simple embedding table lookup operation. Then we utilize mean pooling to seize the mean effect of the recommender exposed item feature vectors as the slate-level representation:

$$\mathbf{c}_{mean} = \frac{1}{N} \sum_{j=1}^{N} \mathbf{e}_j \in \mathbb{R}^d. \tag{2}$$

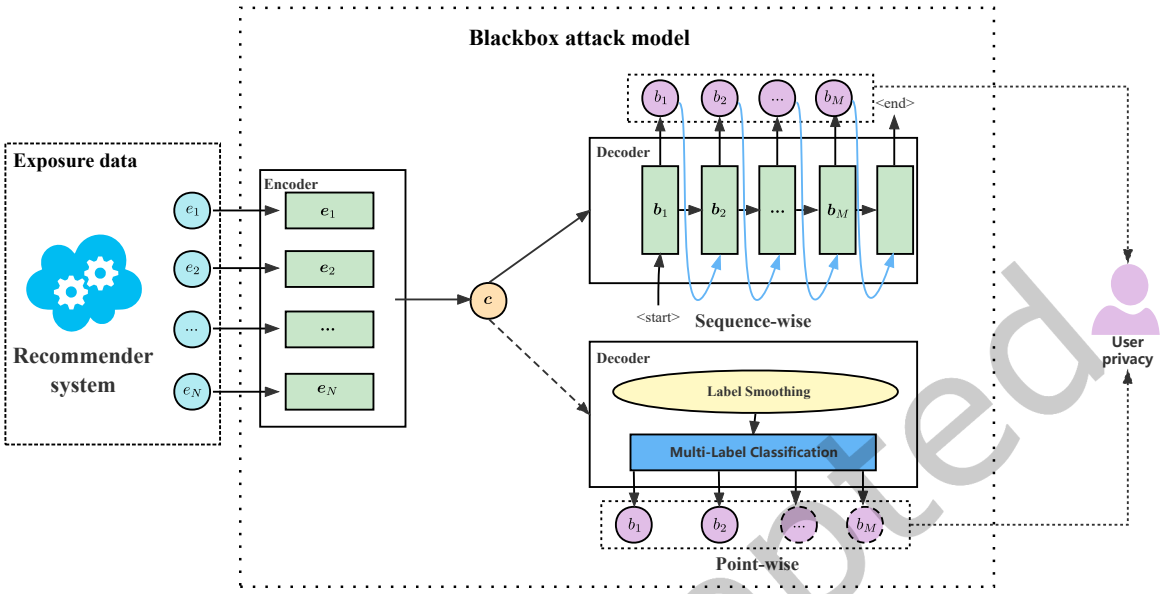


Fig. 3. The overall attack model structure. The encoder aims to map the input of system exposure E_i^u to a latent representation $\mathbf{c}$. Then the user privacy B_i^u could be inferred through point-wise decoding or sequence-wise decoding.

Max pooling-based encoding. While the mean pooling-based encoding attempts to encode the mean effect of system behavior, the max pooling-based encoding forces the encoder to retain only the most useful exposed item features. We select the highest neuron activation value across the whole embedding space:

$$c_{max}(i) = \max_{j=1,...,N} e_j(i), \qquad i = 1, ..., d, \tag{3}$$

where $c_{max}(i)$ denotes the i-th element of the max pooling encoding representation $\mathbf{c}_{max} \in \mathbb{R}^d$. $e_j(i)$ is the i-th element of the item embedding $\mathbf{e}_j$.

Self-attention based encoding. While the mean pooling and max pooling are rather simple encoding methods, we also attempt to exploit the successful transformer encoder [52] to learn the latent representation of the system exposure data. The transformer encoder is based on the self-attention mechanism, which is highly efficient and capable of uncovering semantic patterns of the input data. We illustrate the structure of the self-attention based encoding in Fig. 4. As discussed before, the recommender often exposes a slate of items simultaneously, so we don't introduce the position embeddings in our attack encoder.

The self-attention based encoder contains *multi-head attention* and *feed-forward network*. The multi-head attention adopts scaled dot-product attention at each head to learn the importance of input items. The dot-product based attention is formulated as

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}})\mathbf{V}, \tag{4}$$

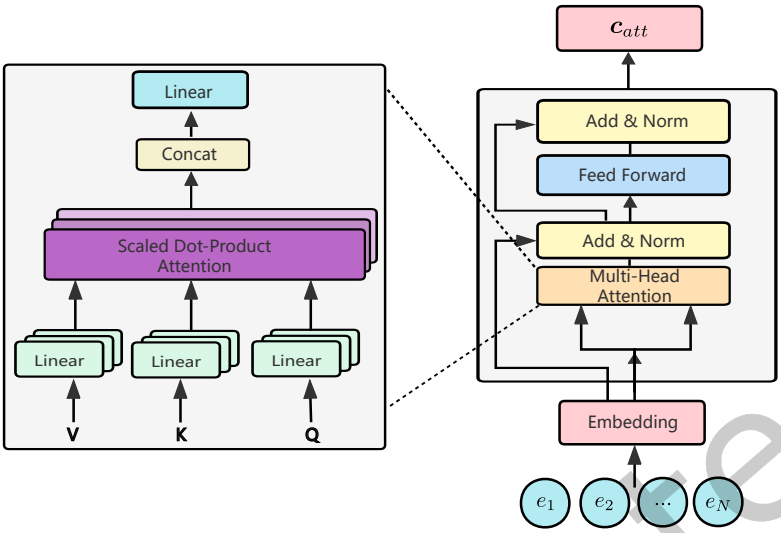


Fig. 4. Self-attention based encoding.

where $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ denote the queries, keys and values, respectively. The attention operation computes a weighted sum of values according to the weights calculated from the correlations between the query and the key. The scale factor $\sqrt{d}$ is used to normalize the computed correlations to avoid overly large inner product. Self-attention uses the same objects as queries, keys, and values. Then the multi-head attention on the input item embeddings is formulated as:

$$\begin{aligned} MHA(\mathbf{E}) &= concat\{head_1, head_2, ..., head_h\}\mathbf{W}^O, \text{ where} \\ head_i &= Attention(\mathbf{W}_i^Q\mathbf{E}, \mathbf{W}_i^K\mathbf{E}, \mathbf{W}_i^V\mathbf{E}). \end{aligned} \tag{5}$$

$\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, ...\mathbf{e}_N] \in \mathbb{R}^{N\times d}$ is the stacked embedding matrix of the exposed items. $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^Q$, and $\mathbf{W}^O$ are trainable parameters. h is the number of heads.

To avoid overfitting and enable a more stable learning without vanishing or exploding gradient issues, we also introduce dropout layers, residual connection, and layer normalization. The representation after multi-head attention is formulated as

$$\tilde{\mathbf{E}} = \mathbf{E} + dropout(MHA(LayerNorm(\mathbf{E}))) \in \mathbb{R}^{N\times d}. \tag{6}$$

After that, a two layer of feed-forward network (FFN) is utilized to increase the encoder capacity. Then, the latent representation after self-attention based encoding is formulated as

$$\mathbf{C}_{att} = \tilde{\mathbf{E}} + dropout(FFN(LayerNorm(\tilde{\mathbf{E}}))) \in \mathbb{R}^{N\times d}. \tag{7}$$

Besides, we insert a CLS token into the original input of $(e_1, e_2, ..e_N)$ and then the corresponding vector of the CLS token in $\mathbf{C}_{att}$ can also be regarded as the final encoded vector representation of system exposure.

3.2.2 ***Decoder***. Based on the encoded latent representation of system behavior, we propose two decoding strategies to infer the privacy of user past behavior, namely point-wise decoding and sequence-wise decoding.

More specifically, for sequence-wise decoding, we utilize three notable sequential models including GRU, LSTM, and the attention-based transformer decoder.

Point-wise decoding. Point-wise decoding treats the inference as a multi-label classification task with each past interaction as a label. The decoder is composed of a fully-connected layer and the classification probability can be defined as

$$\mathbf{q} = [q_1, q_2, ..q_{|\mathcal{I}|}] = softmax(\sigma(\mathbf{W}_d\mathbf{c} + \mathbf{b})) \in \mathbb{R}^{|\mathcal{I}|}, \tag{8}$$

where $\mathbf{q}$ denotes the classification probability, σ is the activation function, $\mathbf{c}$ is $\mathbf{c}_{mean}$, $\mathbf{c}_{max}$ or the CLS token vector in $\mathbf{C}_{att}$ according to different encoding methods. $\mathbf{W}_d \in \mathbb{R}^{d\times|\mathcal{I}|}$ and $\mathbf{b} \in \mathbb{R}^{|\mathcal{I}|}$ are trainable parameters.

Then we use *label smoothing regularization*[17, 51] to generate the ground-truth label for the multi-label classification task. Label smoothing prevents the network from becoming over-confident. It encourages the fully-connected layer to make a finite output and generalize better. Assume the ground-truth user behavior sequence is $B_i^u = (b_1, b_2, ..., b_M)$, the ground-truth probability after label smoothing is defined as

$$y_j = \begin{cases} (1-\varepsilon)/M & \text{if } j \in B_i^u \\ \varepsilon/(|\mathcal{I}| - M) & \text{otherwise,} \end{cases} \tag{9}$$

where ε is a small constant.

Finally, the training loss function for point-wise decoding is formulated as

$$\mathcal{L}(\mathbf{y}, \mathbf{q}) = -\sum_{j=1}^{|\mathcal{I}|} y_j \log q_j. \tag{10}$$

Empirically, we found that label smoothing dramatically downgrade the risk of model overfitting.

Sequence-wise decoding. Point-wise decoding doesn't consider the sequential order of user past behaviors. In this subsection, we describe the detail to use sequence-wise decoding to infer the user behavior privacy in a reversed order (i.e., from the most recently b_M to the earliest b_1). According to subsection 3.2.1, we have obtained a latent representation for the system exposure data. We then use three different sequential models to perform the inference, including GRU [8],LSTM [45], and the transformer decoder [52].

More precisely, during the training process, we process the user behavior sequence $B_i^u = (b_1, b_2, ..., b_M)$ reversely and complement B_i^u with two special tokens as $(< start >, b_M, ..., b_2, b_1, < end >)$. Then we put $B_i^{'u} = (< start >, b_M, ..., b_2, b_1)$ as the input of the decoder and the output is shifted to $(b_M, ..., b_2, b_1, < end >)$. We use $\mathbf{Q} \in \mathbb{R}^{(M+1)\times|\mathcal{I}|}$ to denote the output classification probability of the decoder as

$$\mathbf{Q} = SeqDecoder(\mathbf{C}, B_i^{'u}), \tag{11}$$

where *SeqDecoder* represents the three different sequential models. $\mathbf{C} \in \mathbb{R}^{N\times d}$ is the stacked $\mathbf{c}_{mean}$, $\mathbf{c}_{max}$ or $\mathbf{C}_{att}$ according to different encoding methods. In the following, we briefly introduce the three sequential decoding models.

(1) LSTM-based decoder. LSTM is a famous sequence modeling neural network. We utilize the most common LSTM unit which is composed of a cell, an input gate, an output gate and a forget gate. The cell can remember values over arbitrary time intervals and the above three gates regulate the flow of information into and out of the cell[45]. More precisely, for each time step $t(1 \le t \le M+1)$, the output classification probability $\mathbf{q}_t \in \mathbb{R}^{|\mathcal{I}|}$ from

LSTM-based decoder is formulated as

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_i[\mathbf{h}_{t-1}; \mathbf{b}_{t-1}] + \mathbf{p}_i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_f[\mathbf{h}_{t-1}; \mathbf{b}_{t-1}] + \mathbf{p}_f) \\
\tilde{\mathbf{g}}_t &= tanh(\mathbf{W}_g[\mathbf{h}_{t-1}; \mathbf{b}_{t-1}] + \mathbf{p}_g) \\
\mathbf{g}_t &= \mathbf{i}_t \odot \tilde{\mathbf{g}}_t + \mathbf{f}_t \odot \mathbf{g}_{t-1} \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o[\mathbf{h}_{t-1}; \mathbf{b}_{t-1}] + \mathbf{p}_o) \\
\mathbf{h}_t &= \mathbf{o}_t \odot tanh(\mathbf{g}_t) \\
\mathbf{q}_t &= softmax(FFN(\mathbf{h}_t)),
\end{aligned} \tag{12}$$

where $\mathbf{b}_{t-1}$ is the embedding of (t-1)-th item (i.e., b_{t-1}) in the decoder. $\mathbf{h}_{t-1}$ denote the output hidden state of (t-1)-th step. The $\mathbf{i_t}, \mathbf{f_t}, \mathbf{o_t}$ represent the three gates' activation vector, respectively. $\mathbf{g}_t$ and $\tilde{\mathbf{g}}_t$ are cell state vectors. $\mathbf{W}_i, \mathbf{W}_f, \mathbf{W}_g, \mathbf{W}_o, \mathbf{p}_i, \mathbf{p}_f, \mathbf{p}_g$, and $\mathbf{p}_o$ are trainable parameters. Specially, at the first training time step, the input $\mathbf{b}_0$ is the embedding of the special token $< start >$.

(2) GRU-based decoder. GRU is also a notable gating-based recurrent neural network. Compared with LSTM, it's relatively simpler with fewer parameters. For each time step $t (1 \leq t \leq M+1)$, the output classification probability $\mathbf{q}_t \in \mathbb{R}^{|\mathcal{I}|}$ from GRU-based decoder is formulated as

$$\begin{aligned}
\mathbf{z}_t &= \sigma(\mathbf{W}_z[\mathbf{h}_{t-1}; \mathbf{b}_{t-1}] + \mathbf{p}_z) \\
\mathbf{r}_t &= \sigma(\mathbf{W}_r[\mathbf{h}_{t-1}; \mathbf{b}_{t-1}] + \mathbf{p}_r) \\
\tilde{\mathbf{h}}_t &= tanh(\mathbf{W}_h[\mathbf{r}_t \odot \mathbf{h}_{t-1}; \mathbf{b}_{t-1}] + \mathbf{p}_h) \\
\mathbf{h}_t &= (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t \\
\mathbf{q}_t &= softmax(FFN(\mathbf{h}_t)),
\end{aligned} \tag{13}$$

where $\mathbf{b}_{t-1}$ and $\mathbf{h}_{t-1}$ are similar to LSTM-based decoder. $\mathbf{z}_t$ and $\mathbf{r}_t$ represent the "update gate" vector and the "reset gate" vector, respectively. $\mathbf{W}_z, \mathbf{W}_r, \mathbf{W}_h, \mathbf{p}_z, \mathbf{p}_r$, and $\mathbf{p}_h$ are learnable parameters. Similar with the LSTM-based decoder, the input $\mathbf{b}_0$ is the embedding of the special token $< start >$ at the first training time step.

(3) Attention-based transformer decoder. We don't elaborate the mathematical detail again since the attention-based transformer decoder shares similar calculation with the attention-based encoding method. However, we still need to claim the following differences

- Unlike the encoder, position embeddings are introduced in the decoder to distinguish the order of user past behaviors and introduce the sequential signals.
- Due to the nature of attacking user past behavior sequences, the model should consider only the last j items when inferring the (j-1)-th item. As a result, a casual mask is introduced in the decoder to modify the attention computation as masked multi-head attention.

Different with the point-wise decoding, in sequence-wise decoding we define a cross-entropy loss for each output position to perform the classification. The final loss function for sequence-wise decoding can be formulated as

$$\mathcal{L}(\mathbf{Q}, \mathbf{Y}) = -\sum_{m=1}^{M} \sum_{j=1}^{|\mathcal{I}|} y_{mj} \log q_{mj}, \tag{14}$$

where y_{mj} and q_{mj} are ground-truth probability and the (m, j)-th entry of the computed probability matrix $\mathbf{Q}$ (i.e., the stack of $\mathbf{q}_t$). Note that in the inference stage, we cannot obtain the ground-truth user behaviors like the training process. As a result, we use the inference results of previous steps as the input for the next inference step.

4 PRIVACY PROTECTION

In the previous section, we describe the attack model to infer user past behavior privacy from system exposure data. Here we describe a two-stage protection mechanism which targets on alleviating the privacy risk. The reason that user behavior can be inferred from system exposure data lies in the fact that the items exposed by the system have inherent relationships with the previous interacted items of the user (e.g., similarity or sequential connections). So a simple solution is to infuse random items as the noisy signal into the system exposure to obfuscate the relationships. The key idea is similar with differential privacy [49]. However the difference is that existing differential privacy-based methods add noise to the user data while our protection targeting on introducing noise into the system exposure.

The proposed protection mechanism consists of two stages namely position selection and item replacement. At the first stage, we decide which exposure positions would be replaced using random-based or similarity-based position selection method. After that, we utilize uniform-based or popularity-based replacement strategy to sample items to replace the exposed items on corresponding selected positions. Fig.5 shows the illustration of the protection mechanism.

4.1 Position Selection

At this stage, we design two methods: the random-based method and the similarity-based method to choose which positions in the exposed item slates would be replaced, as shown in the left part of Fig. 5.

Random-based position selection. In this method, we randomly select positions of the exposed item slate according to a uniform distribution. Given the exposed item slate E_i^u=(e_1, e_2, ..., e_N) and a replacement proportion L. We randomly select $m = \lceil N * L \rceil$ positions as p=$\{p_1, p_2, ..., p_m\}$, where $p_i \leq N$.

Similarity-based position selection. The above random-based position selection could lead to a situation that potential future interacted items are switched out and thus downgrades the recommendation accuracy. In similarity-based position selection, we select positions according to the similarity between a specific exposed item and the overall slate representation (i.e., the representation center of all exposed items within a slate). Given the exposed item slate E_i^u, we utilize the item embeddings obtained from the recommendation model(i.e., SASRec[28]) pre-trained using user historical behavior data and perform mean pooling to seize the mean effect of the user historical behavior. The user behavior representation of user u is defined as

$$\mathbf{b}_u = \frac{1}{|B^u|} \sum_{j=1}^{|B^u|} \mathbf{b}_j^u \in \mathbb{R}^d. \tag{15}$$

Then we calculate the softmax cosine similarity between the above user behavior representation $\mathbf{b}_u$ and each exposed item $\mathbf{e}_i$ as

$$s(u,i) = softmax(\frac{\mathbf{e}_i \cdot \mathbf{b}_u}{max(\|\mathbf{e}_i\|_2 \cdot \|\mathbf{b}_u\|_2, \omega)}), \tag{16}$$

where $s(u,i)$ represents the probability similarity between $\mathbf{e}_i$ and the user behavior, and ω is a small constant.

Since $\mathbf{b}_u$ denotes the overall user historical behavior representation, which can be further regarded as the user preference, we thus believe that replacing dissimilar positions could help us to maintain a good recommendation performance. As a result, we assign the probability of selecting a position to be replaced according to the value of $s(u,i)$ and positions with smaller $s(u,i)$ are more likely to be replaced. Finally, we select $m = \lceil N * L \rceil$ positions as p=$\{p_1, p_2, ..., p_m\}$ according to the calculated similarity.

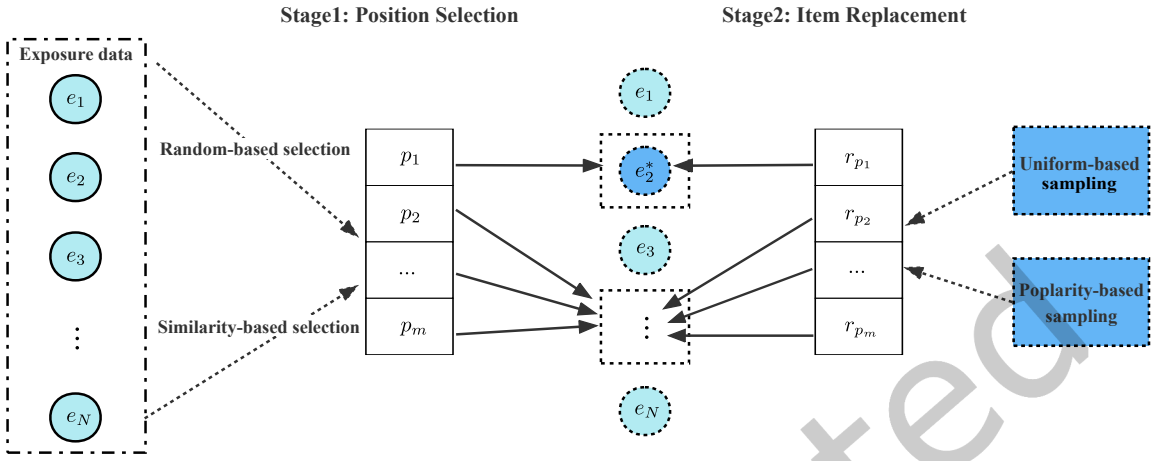


Fig. 5. Privacy protection includes two stages: position selection and item replacement. Position selection decides which exposure positions would be replaced using random-based or similarity-based strategies. Item replacement replaces the exposed items on previously selected positions with uniform or popularity sampled items.

4.2 Item Replacement

After position selection, we investigate two item replacement strategies based on uniform exposure and popularity exposure, as shown in the right part of Fig.5.

Uniform replacement. In this strategy, we randomly sample items from the whole item set according to a uniform distribution to replace the exposed items of the system. For the selected positions p=$\{p_1, p_2, ..., p_m\}$ discussed in section 4.1, we uniformly sample a replacement item set $\{r_{p_1}, r_{p_2}, ..., r_{p_m}\}$. After that, the system exposure data can be updated as

$$E_i^{*u} = (e_1^*, .., e_j^*, .., e_N^*), e_j^* = \begin{cases} r_{p_f} \sim Unif(\mathcal{I}) & \text{if } j = p_f, \\ e_j & \text{otherwise,} \end{cases} \tag{17}$$

where $f \leq m$. We take E_i^{*u} as the input data for the trained attack model to perform a new attack inference to verify how the replacement affect the attack model performance. Besides, the new recommendation accuracy can be defined as

$$acc_u^* = \frac{|E^{*u} \cap B^u|}{|E^{*u}|}. \tag{18}$$

Obviously, the uniform replacement would also affect the recommendation accuracy.

Popularity replacement. In this strategy, we sample the replacement item set according to the item popularity. Same as discussed before, given the exposed item slate E_i^u and the replacement proportion L, we select the positions as p=$\{p_1, p_2, ..., p_m\}$ as described in section 4.1. Then we sample a replacement item set $\{r_{p_1}, r_{p_2}, ..., r_{p_m}\}$ according to the item popularity. Here we adopt two kinds of popularity, the overall popularity and the in-batch popularity. The overall popularity means that the popularity distribution is computed over all items while the in-batch popularity means that the popularity is computed over the items in the current inference batch. Then,

Table 2. Dataset statistics. The impressions can be seen as system behavior data while the clicks can be regarded as the user behavior privacy.

Dataset	Zhihu	MIND
#users	7,963	94,057
#items	64,573	34,376
#impressions	1,000,026	8,584,442
#clicks	271,725	347,727

the system exposure data can be updated as

$$e_j^* = \begin{cases} r_{p_f} \sim Pop(\mathcal{I}) \text{ or } Pop(\mathcal{I}_b) & \text{if } j = p_f, \\ e_j & \text{otherwise,} \end{cases} \tag{19}$$

where $f \leq m$, and $\mathcal{I}_b$ denotes the item set in the batch. Similarly, we take E_i^{*u} as the input data for the trained attack model to perform a new attack inference to verify how the popularity-based replacement affect the attack model performance and the recommendation accuracy.

5 EXPERIMENTS

In this section, we conduct experiments to verify the user behavior privacy leakage risk in recommender systems[2]. We aim to answer the following experimental questions:

RQ1 How does the attack model perform? Whether there is a substantial user behavior leakage risk in recommender systems?
RQ2 How does the number of exposed items affect the attack performance? What's the proper setting to conduct the attack?
RQ3 How does the protection mechanism affect the attack performance and recommendation accuracy?

5.1 Dataset Description

The experiments were conducted on two real-world datasets Zhihu[3] [16] and MIND[4] [57]. Both of the two datasets contain user behavior data (e.g., clicks) and system behavior data (e.g., exposed impressions). Table 2 summarizes the statistics of the two datasets.

Zhihu. This dataset is collected from a large-scale knowledge-acquisition platform. The original data contains question information, answer information and the user profile. Here we focus on the answer recommendation scenario. In the serving period of the recommender, a slate of answers (i.e., the items in our setting) are exposed to the user. This can be seen as the system behavior data. The user may click some of the answers for more detail information, which can be seen as the user behavior data. The dataset contains the show time and click time of all answers (0 for non-click answers). More precisely, given a timestamp t and a user u, the M answer clicks just before t are seen as the past user behavior privacy. The N system exposure impressions just following the timestamp t are regarded as the input for the attack model. Finally, the dataset contains 1,000,026 system

[2]We release our code at https://github.com/nancheng58/On-the-User-Behavior-Leakage-from-Recommender-System-Exposure
[3]https://github.com/THUIR/Zhihu-Dataset
[4]https://msnews.github.io/

impressions and 271,725 user clicks over 64,573 items of 7,963 users. With the default setting of $M = 5$ and $N = 10$, we can get a total of 213,172 pairs of $[B_i^u|E_i^u]$.

MIND. This dataset focuses on the news recommendation scenario, which is collected from the system logs of a news website. The data we use contains 8,584,442 recommended impressions for 94,057 users over 34,376 news with 347,727 total user clicks. Each impression log contains a slate of recommended news and historical user click behaviors of this user before the impression. We sort the historical user clicks according to the timestamp. Then the historical clicked news are regarded as the user behavior privacy and the news in the recommended impressions are seen as the system behavior data. Finally, we can get a total of 291,595 pairs of $[B_i^u|E_i^u]$ with the default setting of $M = 5$ and $N = 10$.

5.2 Evaluation Protocols

5.2.1 Attack evaluation. We adopt cross-validation to evaluate the performance of the attack model. The ratio of training, validation, and test set is 8:1:1. We randomly sample the data of 80% users as the training set. Then the data of 10% users is used for validation and the rest 10% users are regarded as the test users. Such user-based data splits can effectively avoid the potential information shortcuts between training and test. For validation and test, the evaluation is done by providing the attack model with item slates exposed by the system and then checking the rank of the M ground-truth items in the inference results. We adopt Recall to evaluate the attack performance. Let $\tilde{B}_i^u@k$ denote the top-k inference output of the attack model[5]. Recall@k measures how many ground-truth user behaviors are included in $\tilde{B}_i^u@k$, which is formulated as

$$\text{Recall@}k = \frac{|\tilde{B}_i^u@k \cap B_i^u|}{M}. \tag{20}$$

We then report the average Recall@k across the whole test user set as the final results. Besides, we also report normalized discounted cumulative gain (NDCG) and mean reciprocal rank (MRR), which are weighted versions of Recall assigning higher weights to the top-ranked positions of the inference result lists.

5.2.2 Protection evaluation. To verify the effect of the proposed protection mechanism, we first compute E_i^{*u} for user u in test users according to Eq.(17) or Eq.(19). Then we feed E_i^{*u} to the pre-trained attack model to calculate the new attack metrics (i.e., Recall, NDCG, and MRR). Besides, we also compute the new recommendation accuracy for user u according to Eq.(18). Then we report the overall recommendation accuracy as the average of all users in the test set.

5.3 Hyperparameter Settings

We conduct our experiments with a batch size of 400 pairs of system exposure data and user behavior data (i.e.,$[B_i^u|E_i^u]$). The sizes of B_i^u and E_i^u are set as $M = 5$ and $N = 10$, respectively without special mention. The item embedding size is set as $d = 128$. We train all models with the Adam optimizer[31]. The learning rate is set as 0.001. The dropout rate is tuned to 0.1. For attention-based encoding and transformer-based decoder, the hyperparameters of the multi-head self-attention are set as 2 heads with a total of 128 hidden neurons. The hidden size of the FFN is also set as 128. For LSTM-based decoder and GRU-based decoder, the depth of recurrent layer is set to 1. We utilize the weight sharing technique [24, 41] to tie the weights of the encoder item embedding and softmax layer item embedding in the decoder. For label smoothing, the ε is set to the $1/|\mathcal{I}|$. Each experiment is conducted 3 times and the average result is reported.

[5] $\tilde{B}_i^u@k$ contains $k \times M$ items. For point-wise decoding, $\tilde{B}_i^u@k$ is the top-$(k \times M)$ items of the inference results since point-wise decoding doesn't consider the sequential order of user behavior. For sequence-wise decoding, $\tilde{B}_i^u@k$ is composed of the top-k items of all M inference positions.

Table 3. Attack performance with Point-wise decoding. Boldface denotes the highest score. Rec is short for Recall. "Mean", "Max", and "Att" denote three encoding strategies of mean-pooling, max-pooing and self-attention based encoding, respectively.

Datasets	Encoder	Rec@5	NDCG@5	MRR@5	Rec@10	NDCG@10	MRR@10	Rec@20	NDCG@20	MRR@20
Zhihu	Mean	0.0673	0.0403	0.0315	0.1282	0.0597	0.0394	0.2372	0.0870	0.0467
	Max	0.0652	0.0393	0.0309	0.1236	0.0580	0.0385	0.2318	0.0850	0.0458
	Att	**0.0737**	**0.0439**	**0.0342**	**0.1364**	**0.0640**	**0.0424**	**0.2493**	**0.0922**	**0.0500**
MIND	Mean	**0.3998**	**0.2416**	**0.1898**	**0.6684**	**0.3282**	**0.2255**	**0.8828**	**0.3829**	**0.2408**
	Max	0.3984	0.2406	0.1889	0.6631	0.3259	0.2241	0.8762	0.3804	0.2394
	Att	0.3977	0.2399	0.1883	0.6683	0.3271	0.2242	0.8818	0.3817	0.2395

Table 4. Attack performance with LSTM-based decoders. Boldface denotes the highest score. Rec is short for Recall. "Mean", "Max", and "Att" denote three encoding strategies of mean-pooling, max-pooing and self-attention based encoding, respectively.

Datasets	Encoder	Rec@5	NDCG@5	MRR@5	Rec@10	NDCG@10	MRR@10	Rec@20	NDCG@20	MRR@20
Zhihu	Mean	0.0916	0.0579	0.0470	0.1634	0.0809	0.0563	0.2868	0.1118	0.0647
	Max	0.1102	0.0690	0.0556	0.1930	0.0955	0.0664	0.3301	0.1299	0.0757
	Att	**0.1198**	**0.0744**	**0.0596**	**0.2063**	**0.1021**	**0.0709**	**0.3495**	**0.1380**	**0.0806**
MIND	Mean	**0.5861**	**0.3591**	**0.2852**	**0.8772**	**0.4541**	**0.3249**	0.9809	**0.4811**	**0.3327**
	Max	0.5657	0.3435	0.2711	0.8719	0.4432	0.3127	**0.9837**	0.4724	0.3212
	Att	0.5335	0.3353	0.2707	0.8158	0.4269	0.3087	0.9606	0.4644	0.3194

Table 5. Attack performance with GRU-based decoders. Boldface denotes the highest score. Rec is short for Recall. "Mean", "Max", and "Att" denote three encoding strategies of mean-pooling, max-pooing and self-attention based encoding, respectively.

Datasets	Encoder	Rec@5	NDCG@5	MRR@5	Rec@10	NDCG@10	MRR@10	Rec@20	NDCG@20	MRR@20
Zhihu	Mean	0.0875	0.0542	0.0434	0.1578	0.0767	0.0526	0.2841	0.1084	0.0612
	Max	**0.1009**	**0.0628**	**0.0504**	**0.1792**	**0.0878**	**0.0606**	**0.3114**	**0.1210**	**0.0695**
	Att	0.0833	0.0512	0.0408	0.1523	0.0733	0.0497	0.2744	0.1038	0.0580
MIND	Mean	0.5384	0.3265	0.2575	0.8429	0.4255	0.2987	0.9739	0.4596	0.3086
	Max	**0.5586**	**0.3471**	**0.2782**	**0.8506**	**0.4422**	**0.3178**	**0.9748**	**0.4745**	**0.3271**
	Att	0.5414	0.3415	0.2763	0.8110	0.4291	0.3126	0.9561	0.4665	0.3233

5.4 Attack Performance (RQ1)

Table 3 show the attack performance with point-wise decoding on the two datasets. We can see that for point-wise decoding, the self-attention based encoding achieves much better attack performance than mean-pooling and max-pooling based encoding on the Zhihu dataset. While for the MIND dataset, the three different encoding methods achieve similar attack performance.

The attack performance of the three sequence-wise decoding methods are shown in Table 4, Table 5, and Table 6. We can see from Table 4 that for LSTM-based decoder, the self-attention based encoding achieves the

Table 6. Attack performance with transformer decoders. Boldface denotes the highest score. Rec is short for Recall. "Mean", "Max", and "Att" denote three encoding strategies of mean-pooling, max-pooing and self-attention based encoding, respectively.

Datasets	Encoder	Rec@5	NDCG@5	MRR@5	Rec@10	NDCG@10	MRR@10	Rec@20	NDCG@20	MRR@20
Zhihu	Mean	**0.1453**	**0.0946**	**0780**	**0.2359**	**0.1235**	**0.0898**	**0.3748**	**0.1585**	**0.0992**
	Max	0.1426	0.0920	0.0755	0.2339	0.1213	0.0874	0.3725	0.1561	0.0968
	Att	0.1344	0.0850	0.0699	0.2293	0.1161	0.0822	0.3721	0.1520	0.0919
MIND	Mean	0.5742	0.3714	0.3051	**0.8232**	0.4524	0.3389	**0.9549**	0.4864	0.3485
	Max	**0.5758**	**0.3729**	**0.3066**	0.8208	**0.4526**	**0.3398**	0.9519	**0.4864**	**0.3494**
	Att	0.5647	0.3639	0.2983	0.8119	0.4443	0.3316	0.9498	0.4798	0.3417

highest attack performance on the Zhihu dataset. While on the MIND dataset, the mean-pooling based encoding achieves much better performance than max-pooling and self-attention based encoding. For GRU-based decoder performance shown in Table 5, we can see that the max-pooling encoding performance is better than the other two encoding methods on both the two datasets. The reason could be that the max-pooling encoding can select the most important features for GRU-based decoder. Table 6 shows the attack results of the transformer decoder. We can see that for the transformer decoder, the mean-pooling encoding achieves the best result on the Zhihu dataset. On the MIND dataset, the max-pooling encoding achieves the highest attack performance.

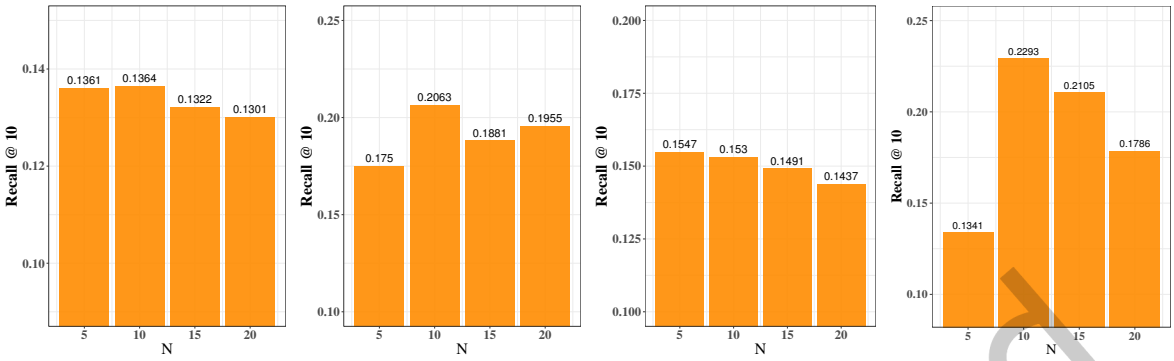
Through the comparison between point-wise decoding (i.e., Table 3) and sequence-wise decoding (i.e., Table 4, Table 5, and Table 6), we can see the sequence-wise decoding achieves much better performance than point-wise decoding with the combination of all three encoders. Such results indicates that considering the sequential order of user behaviours could further improve the attack model performance.

Regarding the encoding methods, we can see that on the Zhihu dataset, self-attention based encoding achieves the best performance when using point-wise decoding and the LSTM-based decoder. While for the MIND dataset, the mean-pooling encoding achieves the highest attack scores with point-wise decoding and the LSTM-based decoder. It demonstrates that for different datasets, we should choose different encoders to conduct the attack. Besides, we can see the compared with the powerful self-attention based encoding, the much simpler mean-pooling and max-pooling can still achieve comparable and even better attack performance. It further demonstrate the risk of privacy leakage since the attack can be performed without trivial and complex encoding methods.

Besides, we can see that the attack model performs much better on MIND other than Zhihu. The reason could be that the Zhihu dataset comes from an answer recommendation scenario. Each answer has an underlying latent question. As a result, the attack model could encounter difficulty to infer the change of underlying questions, leading to relatively lower attack model performance than the MIND dataset.

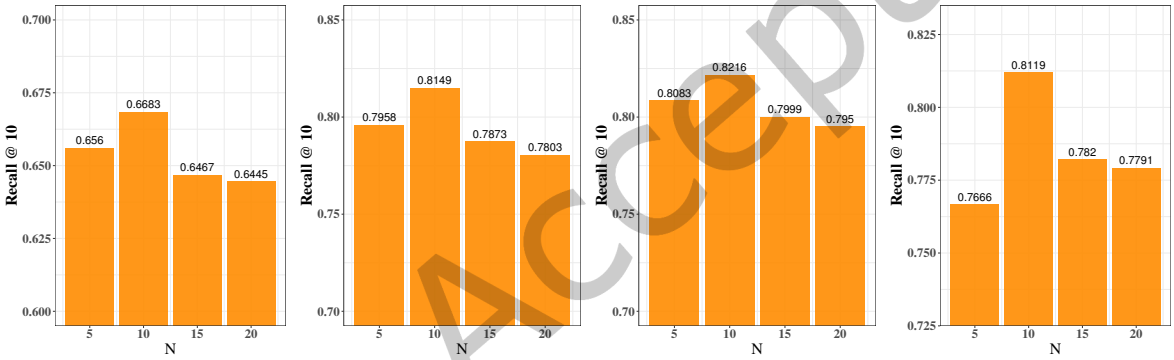
Taking an overall look at the results of Table 3, Table 4, Table 5, and Table 6, we can see that on the Zhihu dataset, the attack model can achieve the highest 23.59% recall in the top-10 list of the inference results. It indicates a great danger that more than 20% of user privacy can be exactly inferred from top-10 attack results. Regarding the MIND dataset, we can see that the privacy risk leakage further increases to the highest 87.72% recall@10 and 98.09% recall@20.

To validate this common risk in the recommendation, we don't focus on the trivial model design and even though utilizing the simple encoder-decoder architecture also can achieve great attack performance. The reason is that exposure data includes enough information to infer the user's historical behavior, even the mean-pooling encoder can learn the representation of exposure data easily.



(a) Point-wise decoding (b) LSTM-based decoder (c) GRU-based decoder (d) Transformer decoder

Fig. 6. Attack performance with different N on Zhihu.



(a) Point-wise decoding (b) LSTM-based decoder (c) GRU-based decoder (d) Transfrmer decoder

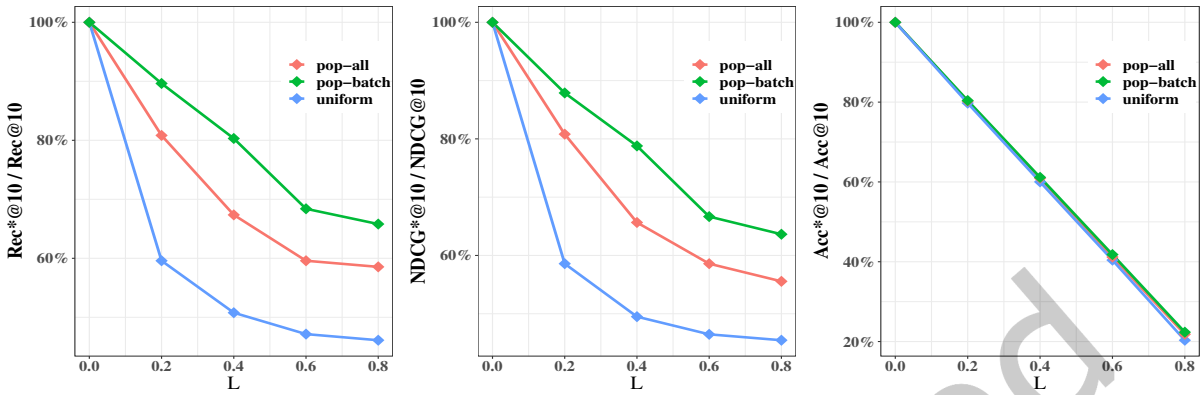
Fig. 7. Attack performance with different N on MIND.

To conclude, we can see that there is substantial user privacy leakage in recommender systems. A large amount of user past behavior privacy can be inferred from system exposure data through the simple encoder-decoder based attack model.

5.5 Hyperparameter Study (RQ2)

In this subsection, we conduct experiments to see how the number of exposed items affect the attack model performance. We illustrate the results when using the self-attention based encoding method. The results of the other two encoding methods show the similar trend.

Fig. 6a, Fig. 6b, Fig. 6c, and Fig. 6d show the attack performance (Recall@10) of point-wise decoding, LSTM-based decoders, GRU-based decoders, and transformer decoders on Zhihu, respectively. We can see that when using point-wise decoding, different exposed item numbers lead to similar attack performance. However, when using sequence-wised decoding, the attack performance is much better and varies much more with different N. The reason could be that the recommendation scenario of Zhihu is answer recommendation. Each answer



(a) Effect on Recall of the attack model (b) Effect on NDCG of the attack model (c) Effect on recommendation accuracy

Fig. 8. Effect of the protection mechanism with random-based position selection on Zhihu. L is the proportion of replacement. Rec*, NDCG* and Acc* denote the new attack recall, new attack NDCG and new recommendation accuracy, respectively.

belongs to a latent question. Due to the fact that point-wise decoding doesn't model the sequential order of user behavior, so point-wise decoding cannot capture the change of underlying questions of the answers, leading to a more flat and lower attack performance. However, the sequence-wise decoding performs the inference based on the previous inference results; such a method could have some capability to learn the change of latent underlying questions from the previous inference results. As a result, the sequence-wise decoding achieves a much higher attack performance with the proper setting of N.

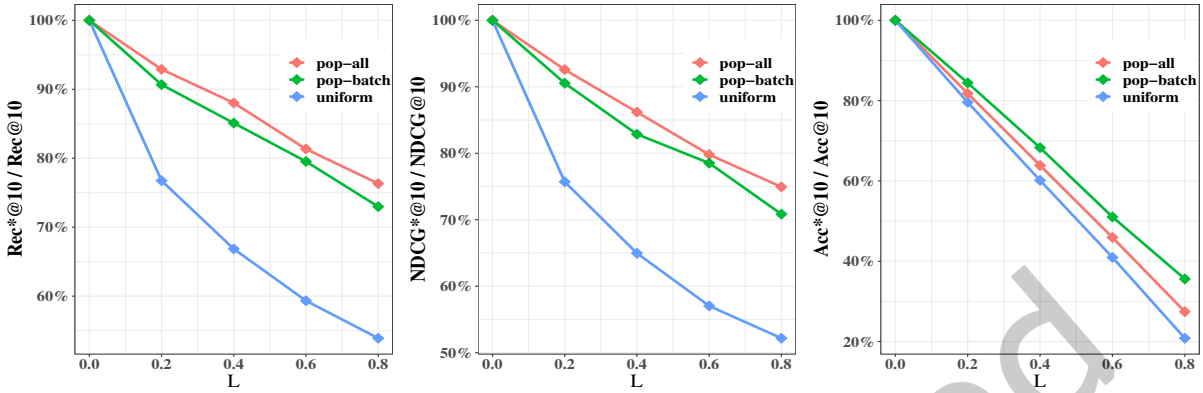
Fig. 7a, Fig. 7b, Fig. 7c, and Fig. 7d show the attack performance (Recall@10) of point-wise decoding and three different sequence-wise decoding methods on the MIND dataset, correspondingly. We can see that on the news recommendation dataset, both the point-wise decoding and sequence-wise decoding methods have more various attack performance with the different setting of N.

Besides, we can see that on both datasets, the best attack performance is achieved when $N = 10$. This observation indicates that with fewer exposed items, the attack model may not learn strong signals to infer user behavior privacy. However, a too large number of exposed items could also introduce extra noise, which further confuses the model and downgrades the attack performance. $N = 10$ could be a proper setting for the attack.

5.6 Effect of Protection (RQ3)

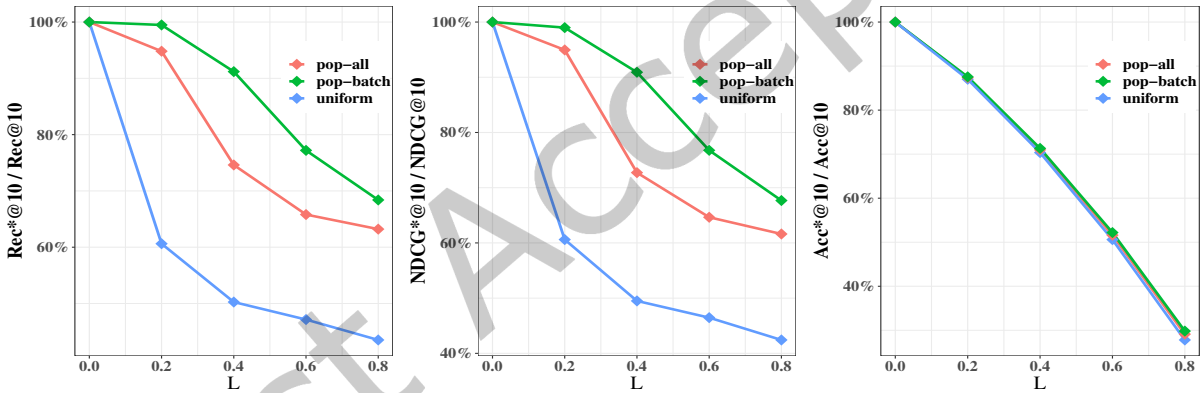
In this subsection, we conduct experiments to verify the effect of the protection mechanism. We show the attack performance regarding recall and NDCG[6]; we also report the new recommendation accuracy. Fig. 8 and Fig. 9 illustrate the results with different item replacement ratios using random-based position selection at the first stage. The results using similarity-based position selection with different item replacement ratios are shown in Fig. 10, and Fig. 11. We can see that with the increase of the replacement proportion L, the attack performance drops dramatically. Such observation demonstrates that random exposure can help to alleviate the user privacy leakage risk. However, we can also see from Fig. 8c, Fig. 9c, Fig. 10c and Fig. 11c that the recommendation accuracy decreases. It indicates a trade-off effect between the recommendation accuracy and the leakage risk. We can see from Fig. 8 that random-based position selection method with uniform-based replacement mechanism alleviate

[6]Results of MRR show the same trend with NDCG.



(a) Effect on Recall of the attack model (b) Effect on NDCG of the attack model (c) Effect on recommendation accuracy

Fig. 9. Effect of the protection mechanism with random-based position selection on MIND. L is the proportion of replacement. Rec*, NDCG* and Acc* denote the new attack recall, new attack NDCG and new recommendation accuracy, respectively.

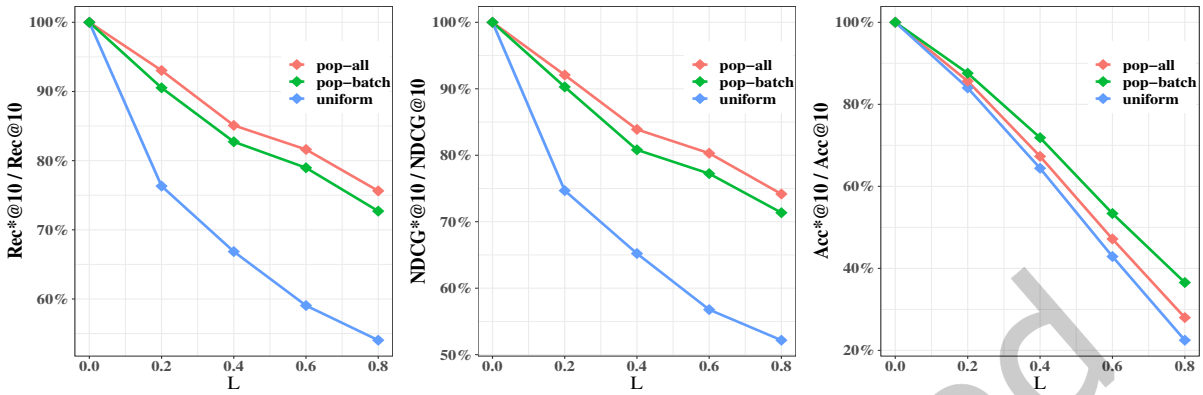


(a) Effect on Recall of the attack model (b) Effect on NDCG of the attack model (c) Effect on recommendation accuracy

Fig. 10. Effect of the protection mechanism with similarity-based position selection on Zhihu. L is the proportion of replacement. Rec*, NDCG* and Acc* denote the new attack recall, new attack NDCG and new recommendation accuracy, respectively.

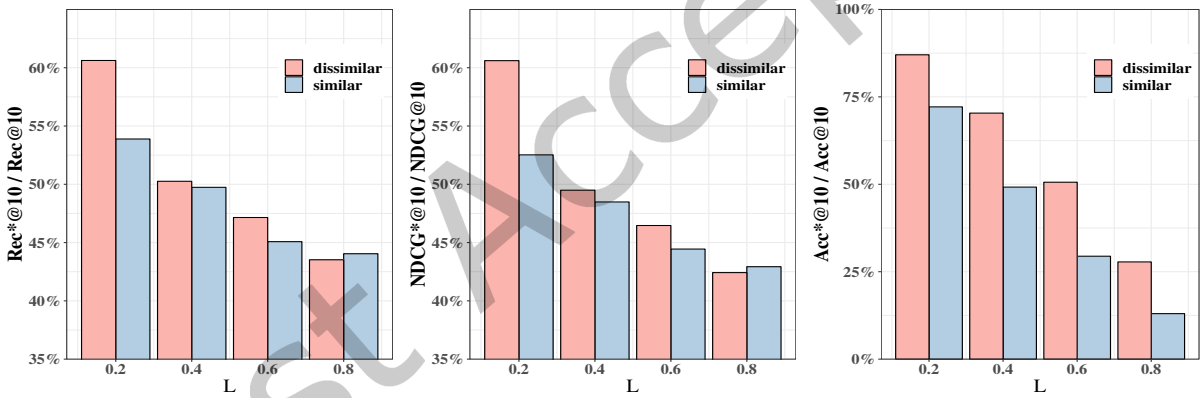
about 40% leakage risk while maintain about 80% recommendation accuracy when the setting of the replacement ration L to 0.2 in the Zhihu dataset.

Given the fact that users would only click very few items in the exposed list, the designed similarity-based position selection method, which can keep the items that the user would like to click and then replace the other items with random or more diverse items, achieve better recommendation accuracy. We can see from the comparison between Fig. 8c and Fig. 10c (also (Fig. 9c and Fig. 11c)) that compared with random-based position selection method, the similarity-based selection method maintain higher recommendation accuracy with the increase of the replacement ratio L, when using uniform-based item replacement.



(a) Effect on Recall of the attack model (b) Effect on NDCG of the attack model (c) Effect on recommendation accuracy

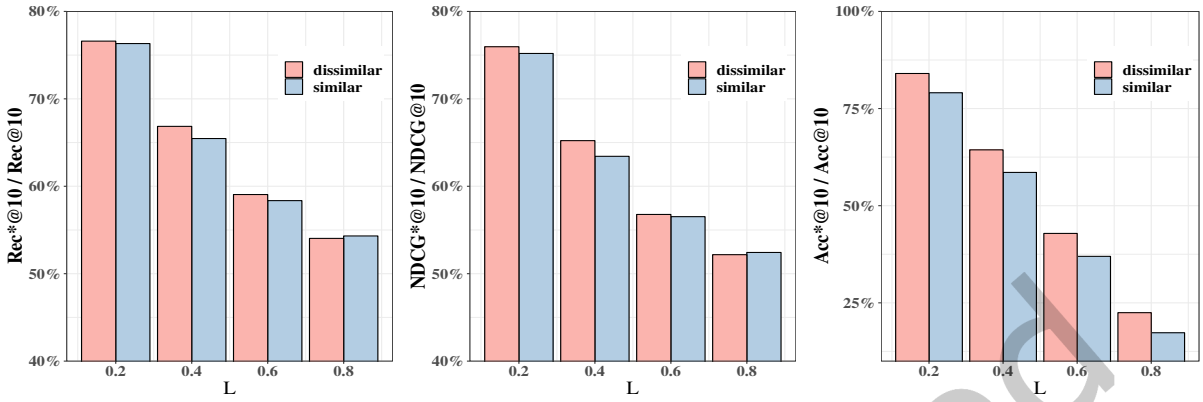
Fig. 11. Effect of the protection mechanism with similarity-based position selection on MIND. L is the proportion of replacement. Rec*, NDCG* and Acc* denote the new attack recall, new attack NDCG and new recommendation accuracy, respectively.



(a) Effect on Recall of the attack model (b) Effect on NDCG of the attack model (c) Effect on recommendation accuracy

Fig. 12. Effect of the similarity-based position selection protection mechanism with uniform-based item replacement on Zhihu. L is the proportion of replacement. Rec*, NDCG* and Acc* denote the new attack recall, new attack NDCG and new recommendation accuracy, respectively.

Besides, Fig. 12 and Fig.13 show the results of using similarity-based position selection at the first stage and uniform-based replacement at the second stage on two datasets. Different from the method of replacing the dissimilar position items (i.e., similarity-based position selection method with uniform-based replacement in Fig. 10 and Fig. 11), the similar position selection method selects positions with items matching well with user preference. The results show that it can alleviate user privacy leakage risk at the cost of recommendation accuracy degradation compared with replacing the dissimilar positions items. The reason could be that more similar positions items reveal much more user behavior information.



(a) Effect on Recall of the attack model (b) Effect on NDCG of the attack model (c) Effect on recommendation accuracy

Fig. 13. Effect of the similarity-based position selection protection mechanism with uniform-based item replacement on MIND. L is the proportion of replacement. Rec*, NDCG* and Acc* denote the new attack recall, new attack NDCG and new recommendation accuracy, respectively.

Regarding the item replacement strategy, we can see that uniform-based replacement is the most effective one to downgrade the attack performance while in-bath popularity replacement and overall popularity replacement have the similar effect on attack performance. For the recommendation accuracy, in-batch popularity replacement achieves the minimum accuracy sacrifice.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we have investigated the risk of user behavior privacy leakage in the field of recommender systems. We have focused on answering the question of whether the user past behavior privacy can be inferred from system behavior data. We have conducted an attack model through an encoder-decoder architecture. We have proposed to utilize three different encoding methods to encode the system exposure data. Thereafter, we have presented the point-wise decoding and three sequence-wise decoders to infer user past behaviors from the encoded representation. Experimental results on two real-world datasets have verified a great danger of privacy leakage in recommender systems. To alleviate the risk, we have proposed a two-stage protection mechanism based on infusing random items into the exposed item sets. We first select exposure positions based on random or item similarity at the first stage, and then replace exposed items on the corresponding positions with uniform or popularity sampled items. Experimental results have demonstrated a trade-off effect between the recommendation accuracy and the privacy leakage risk.

We hope that this work could raise more community concerns regarding the protection of recommender system behavior data other than just focusing on user perspectives. Compared with the sparse user historical behavior, the large volume of system exposure data receives relatively less research attention. This work have proposed a new perspective regarding the attack and protection of the system behavior data.

Future work includes investigating more advanced encoding and decoding methods to conduct the attack. Besides, the exposure data is highly affected by various kinds of biases (e.g., the popularity bias) so how to conduct debiased attack model would also be a research direction. More importantly, one of the promising future directions is the design of more advanced protection methods with little sacrifice of the recommendation

accuracy. Finally, exploring the relationship across the recommendation accuracy, the privacy leakage risk, and the recommendation diversity or novelty would also be an interesting direction.

ACKNOWLEDGEMENTS

This work was supported by the Natural Science Foundation of China (62272274, 62202271, 61972234, 61902219, 62072279, 62102234), the Key Scientific and Technological Innovation Program of Shandong Province (2019JZZY010129), the Fundamental Research Funds of Shandong University, Meituan, the Tencent WeChat Rhino-Bird Focused Research Program (JR-WXG-2021411). All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] Muhammad Ammad-ud din, Elena Ivannikova, A. Suleiman Khan, Were Oyomno, Qiang Fu, Eeik Kuan Tan, and Adrian Flanagan. 2019. Federated Collaborative Filtering for Privacy-Preserving Personalized Recommendation System. *arXiv: Information Retrieval* (2019).

[2] Ghazaleh Beigi, Ahmadreza Mosallanezhad, Ruocheng Guo, Hamidreza Alvari, Alexander Nou, and Huan Liu. 2020. Privacy-Aware Recommendation with Private-Attribute Protection using Adversarial Learning. *WSDM '20: The Thirteenth ACM International Conference on Web Search and Data Mining Houston TX USA February, 2020* (2020), 34–42.

[3] Di Chai, Leye Wang, Kai Chen, and Qiang Yang. 2021. Secure Federated Matrix Factorization. *IEEE Intell. Syst.* 36, 5 (2021), 11–20.

[4] Jiawei Chen, Hande Dong, Yang Qiu, Xiangnan He, Xin Xin, Liang Chen, Guli Lin, and Keping Yang. 2021. AutoDebias: Learning to Debias for Recommendation. In *SIGIR*. ACM, 21–30.

[5] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. 2020. Bias and Debias in Recommender System: A Survey and Future Directions. *CoRR* abs/2010.03240 (2020).

[6] Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H Chi. 2019. Top-k off-policy correction for a REINFORCE recommender system. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. ACM, 456–464.

[7] 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 1st workshop on deep learning for recommender systems*. 7–10.

[8] 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 *EMNLP*. ACL, 1724–1734.

[9] Yashar Deldjoo, Tommaso Di Noia, and Felice Antonio Merra. 2021. A survey on Adversarial Recommender Systems: from Attack/Defense strategies to Generative Adversarial Networks. *Comput. Surveys* (2021). https://doi.org/10.1145/3439729

[10] Jingtao Ding, Fuli Feng, Xiangnan He, Guanghui Yu, Yong Li, and Depeng Jin. 2018. An improved sampler for bayesian personalized ranking by leveraging view data. In *Companion Proceedings of the The Web Conference 2018*. 13–14.

[11] Jingtao Ding, Fuli Feng, Xiangnan He, Guanghui Yu, Yong Li, and Depeng Jin. 2018. An Improved Sampler for Bayesian Personalized Ranking by Leveraging View Data. In *Companion Proceedings of the The Web Conference 2018* (Lyon, France) *(WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 13–14. https://doi.org/10.1145/3184558.3186905

[12] Jingtao Ding, Yuhan Quan, Xiangnan He, Yong Li, and Depeng Jin. 2019. Reinforced Negative Sampling for Recommendation with Exposure Data. In *IJCAI*. ijcai.org, 2230–2236.

[13] Chen Gao, Chao Huang, Dongsheng Lin, Depeng Jin, and Yong Li. 2020. DPLCF: Differentially Private Local Collaborative Filtering. In *SIGIR*. ACM, 961–970.

[14] Zhengqiang Ge, Xinyu Liu, Qiang Li, Yu Li, and Dong Guo. 2021. PrivItem2Vec: A privacy-preserving algorithm for top-N recommendation. *International Journal of Distributed Sensor Networks* 17, 12 (2021). https://doi.org/10.1177/15501477211061250

[15] Jialiang Han, Yun Ma, Qiaozhu Mei, and Xuanzhe Liu. 2021. DeepRec: On-device Deep Learning for Privacy-Preserving Sequential Recommendation in Mobile Commerce. In *Proceedings of the Web Conference 2021*. 900–911.

[16] Bin Hao, Min Zhang, Weizhi Ma, Shaoyun Shi, Xinxing Yu, Houzhi Shan, Yiqun Liu, and Shaoping Ma. 2021. A Large-Scale Rich Context Query and Recommendation Dataset in Online Knowledge-Sharing. arXiv:2106.06467 [cs.IR]

[17] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. 2019. Bag of Tricks for Image Classification with Convolutional Neural Networks. In *CVPR*. Computer Vision Foundation / IEEE, 558–567.

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

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

[20] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

[21] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *ICLR (Poster)*.

[22] Balázs Hidasi and Domonkos Tikk. 2016. General factorization framework for context-aware recommendations. *Data Mining and Knowledge Discovery* 30, 2 (2016), 342–371.

[23] Yujing Hu, Qing Da, Anxiang Zeng, Yang Yu, and Yinghui Xu. 2018. Reinforcement learning to rank in e-commerce search engine: Formalization, analysis, and application. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 368–377.

[24] Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying Word Vectors and Word Classifiers: A Loss Framework for Language Modeling. In *ICLR (Poster)*. OpenReview.net.

[25] Arjan JP Jeckmans, Michael Beye, Zekeriya Erkin, Pieter Hartel, Reginald L Lagendijk, and Qiang Tang. 2013. Privacy in recommender systems. In *Social media retrieval*. Springer, 263–281.
[26] Ray Jiang, Silvia Chiappa, Tor Lattimore, András György, and Pushmeet Kohli. 2019. Degenerate Feedback Loops in Recommender Systems. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society* (Honolulu, HI, USA) *(AIES '19)*. Association for Computing Machinery, New York, NY, USA, 383–390. https://doi.org/10.1145/3306618.3314288
[27] Santosh Kabbur, Xia Ning, and George Karypis. 2013. Fism: factored item similarity models for top-n recommender systems. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 659–667.
[28] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 197–206.
[29] Farwa K Khan, Adrian Flanagan, Kuan Eeik Tan, Zareen Alamgir, and Muhammad Ammad-Ud-Din. 2021. A Payload Optimization Method for Federated Recommender Systems. In *Recsys*. 432–442.
[30] Jinsu Kim, Dongyoung Koo, Yuna Kim, Hyunsoo Yoon, Junbum Shin, and Sungwook Kim. 2018. Efficient Privacy-Preserving Matrix Factorization for Recommendation via Fully Homomorphic Encryption. *ACM Trans. Priv. Secur.* 21, 4 (2018), 17:1–17:30.
[31] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR (Poster)*.
[32] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.
[33] Jianxun Lian, Fuzheng Zhang, Min Hou, Hongwei Wang, Xing Xie, and Guangzhong Sun. 2017. Practical Lessons for Job Recommendations in the Cold-Start Scenario. In *Proceedings of the Recommender Systems Challenge 2017* (Como, Italy) *(RecSys Challenge '17)*. Association for Computing Machinery, New York, NY, USA, Article 4, 6 pages. https://doi.org/10.1145/3124791.3124794
[34] Guanyu Lin, Feng Liang, Weike Pan, and Zhong Ming. 2020. Fedrec: Federated recommendation with explicit feedback. *IEEE Intelligent Systems* (2020).
[35] Dugang Liu, Pengxiang Cheng, Zhenhua Dong, Xiuqiang He, Weike Pan, and Zhong Ming. 2020. A General Knowledge Distillation Framework for Counterfactual Recommendation via Uniform Data. In *SIGIR*. ACM, 831–840.
[36] Lorenzo Minto, Moritz Haller, Benjamin Livshits, and Hamed Haddadi. 2021. Stronger Privacy for Federated Collaborative Filtering With Implicit Feedback. In *RecSys*. ACM, 342–350.
[37] Khalil Muhammad, Qinqin Wang, Diarmuid O'Reilly-Morgan, Elias Tragos, Barry Smyth, Neil Hurley, James Geraci, and Aonghus Lawlor. 2020. Fedfast: Going beyond average for faster training of federated recommender systems. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1234–1242.
[38] Xia Ning and George Karypis. 2011. Slim: Sparse linear methods for top-n recommender systems. In *2011 IEEE 11th International Conference on Data Mining*. IEEE, 497–506.
[39] Marta Otto. 2018. Regulation (EU) 2016/679 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data (General Data Protection Regulation–GDPR). In *International and European Labour Law*. Nomos Verlagsgesellschaft mbH & Co. KG, 958–981.
[40] Michael J. Pazzani and Daniel Billsus. 2007. Content-Based Recommendation Systems. In *The Adaptive Web (Lecture Notes in Computer Science, Vol. 4321)*. Springer, 325–341.
[41] Ofir Press and Lior Wolf. 2017. Using the Output Embedding to Improve Language Models. In *EACL (2)*. Association for Computational Linguistics, 157–163.
[42] Tao Qi, Fangzhao Wu, Chuhan Wu, Yongfeng Huang, and Xing Xie. 2020. Privacy-Preserving News Recommendation Model Learning. In *EMNLP (Findings) (Findings of ACL, Vol. EMNLP 2020)*. Association for Computational Linguistics, 1423–1432.
[43] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized Markov chains for next-basket recommendation. In *WWW*. ACM, 811–820.
[44] Yuta Saito, Suguru Yaginuma, Yuta Nishino, Hayato Sakata, and Kazuhide Nakata. 2020. Unbiased Recommender Learning from Missing-Not-At-Random Implicit Feedback. In *WSDM*. ACM, 501–509.
[45] Hasim Sak, Andrew W. Senior, and Françoise Beaufays. 2014. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *INTERSPEECH*. ISCA, 338–342.
[46] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. 285–295.
[47] Hyejin Shin, Sungwook Kim, Junbum Shin, and Xiaokui Xiao. 2018. Privacy Enhanced Matrix Factorization for Recommendation with Local Differential Privacy. *IEEE Trans. Knowl. Data Eng.* 30, 9 (2018), 1770–1782.
[48] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 3–18.
[49] Lisa J Sotto, Bridget C Treacy, and Melinda L McLellan. 2010. Privacy and Data Security Risks in Cloud Computing. *World Communications Regulation Report* 5, 2 (2010), 38.
[50] 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 the 28th ACM international conference on information and*

knowledge management. 1441–1450.
[51] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2016. Rethinking the Inception Architecture for Computer Vision. In *CVPR*. IEEE Computer Society, 2818–2826.
[52] 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 *NIPS*. 5998–6008.
[53] Qinyong Wang, Hongzhi Yin, Tong Chen, Junliang Yu, Alexander Zhou, and Xiangliang Zhang. 2021. Fast-adapting and privacy-preserving federated recommender system. *The VLDB Journal* (2021), 1–20.
[54] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 165–174.
[55] Xiang Wang, Yaokun Xu, Xiangnan He, Yixin Cao, Meng Wang, and Tat-Seng Chua. 2020. Reinforced negative sampling over knowledge graph for recommendation. In *Proceedings of The Web Conference 2020*. 99–109.
[56] Chuhan Wu, Fangzhao Wu, Yang Cao, Yongfeng Huang, and Xing Xie. 2021. Fedgnn: Federated graph neural network for privacy-preserving recommendation. *arXiv preprint arXiv:2102.04925* (2021).
[57] Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, and Ming Zhou. 2020. MIND: A Large-scale Dataset for News Recommendation. In *ACL*. Association for Computational Linguistics, 3597–3606.
[58] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. 153–162.
[59] Xin Xin, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M Jose. 2020. Self-supervised reinforcement learning for recommender systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 931–940.
[60] Jheng-Hong Yang, Chih-Ming Chen, Chuan-Ju Wang, and Ming-Feng Tsai. 2018. HOP-rec: high-order proximity for implicit recommendation. In *Recsys*. 140–144.
[61] Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu. 2019. Federated learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 13, 3 (2019), 1–207.
[62] Fajie Yuan, Xiangnan He, Alexandros Karatzoglou, and Liguang Zhang. 2020. Parameter-efficient transfer from sequential behaviors for user modeling and recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1469–1478.
[63] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. 2019. A Simple Convolutional Generative Network for Next Item Recommendation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. ACM, 582–590.
[64] Minxing Zhang, Zhaochun Ren, Zihan Wang, Pengjie Ren, Zhumin Chen, Pengfei Hu, and Yang Zhang. 2021. Membership Inference Attacks Against Recommender Systems. In *CCS*. ACM, 864–879.
[65] Shijie Zhang and Hongzhi Yin. 2022. Comprehensive Privacy Analysis on Federated Recommender System against Attribute Inference Attacks. *CoRR* abs/2205.11857 (2022).
[66] Shijie Zhang, Hongzhi Yin, Tong Chen, Zi Huang, Lizhen Cui, and Xiangliang Zhang. 2021. Graph Embedding for Recommendation against Attribute Inference Attacks. In *WWW*. ACM / IW3C2, 3002–3014.
[67] Yang Zhang, Fuli Feng, Xiangnan He, Tianxin Wei, Chonggang Song, Guohui Ling, and Yongdong Zhang. 2021. Causal Intervention for Leveraging Popularity Bias in Recommendation. In *SIGIR*. ACM, 11–20.