On-device Integrated Re-ranking with Heterogeneous Behavior Modeling

Yunjia Xi
xiyunjia@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Weiwen Liu*
liuweiwen8@huawei.com
Huawei Noah's Ark Lab
Shenzhen, China

Yang Wang*
ywang@sei.ecnu.edu.cn
East China Normal University
Shanghai, China

Ruiming Tang
tangruiming@huawei.com
Huawei Noah's Ark Lab
Shenzhen, China

Weinan Zhang
wnzhang@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Yue Zhu
zhuyue9@huawei.com
Consumer Business Group, Huawei
Shenzhen, China

Rui Zhang
rayteam@yeah.net
ruizhang.info
Shenzhen, China

Yong Yu
yyu@apex.sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

ABSTRACT

As an emerging field driven by industrial applications, integrated re-ranking combines lists from upstream sources into a single list, and presents it to the user. The quality of integrated re-ranking is especially sensitive to real-time user behaviors and preferences. However, existing methods are all built on the cloud-to-edge framework, where mixed lists are generated by the cloud model and then sent to the devices. Despite its effectiveness, such a framework fails to capture users' real-time preferences due to the network bandwidth and latency. Hence, we propose to place the integrated re-ranking model on devices, allowing for the full exploitation of real-time behaviors. To achieve this, we need to address two key issues: first, how to extract users' preferences for different sources from heterogeneous and imbalanced user behaviors; second, how to explore the correlation between the extracted personalized preferences and the candidate items. In this work, we present the first on-Device Integrated Re-ranking framework, DIR, to avoid delays in processing real-time user behaviors. DIR includes a *multi-sequence behavior modeling* module to extract the user's source-level preferences, and a *preference-adaptive re-ranking* module to incorporate personalized source-level preferences into the re-ranking of candidate items. Besides, we design exposure loss and utility loss to jointly optimize exposure fairness and overall utility. Extensive experiments on three datasets show that DIR significantly outperforms the state-of-the-art baselines in utility-based and fairness-based metrics.

*Corresponding author.

KDD '23, August 6–10, 2023, Long Beach, CA, USA.

ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599878

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Recommender System, Edge Computing, Integrated Re-ranking

ACM Reference Format:
Yunjia Xi, Weiwen Liu, Yang Wang, Ruiming Tang, Weinan Zhang, Yue Zhu, Rui Zhang, and Yong Yu. 2023. On-device Integrated Re-ranking with Heterogeneous Behavior Modeling. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3580305.3599878

1 INTRODUCTION

Integrated re-ranking (also referred to as mixed re-ranking or multi-source re-ranking) is a rapidly emerging field driven by industrial applications. Different from traditional re-ranking that reorders lists from a single source, the integrated re-ranking system takes as input multiple ranking lists from different upstream sources (e.g., news, videos, photos, advertisements), and derives a mixed list of items. Items belonging to different sources usually lie in distinct feature spaces (i.e., heterogeneous), and are subject to different ranking models. The basic objective of integrated re-ranking is to optimize the overall listwise utility (e.g., the number of clicks), but there may also be other objectives, such as fairness. As the last stage of recommendation, the quality of the integrated re-ranking is especially sensitive to real-time user behaviors, which contain rich information about users' preferences and interests.

Existing integrated re-ranking work, however, might fall short of capturing such users' real-time preferences. They are all built on the *cloud-to-edge framework* [12, 22, 37, 40], where integrated re-ranking models serving on the cloud generate the mixed lists and then send the lists to the edge (i.e., the mobile devices). This architecture allows for the full utilization of cloud-side capabilities, whereas it fails to capture users' timely interest changes due to the network bandwidth and latency [14]. A short delay in time may

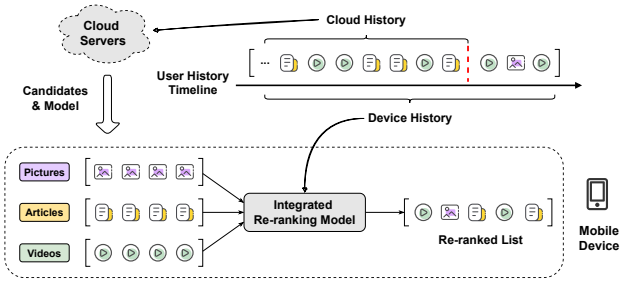

Figure 1: Integrated Re-ranking on the Edge.

cause great performance degradation for integrated re-ranking. For example, as displayed in the upper right of Figure 1, the historical behaviors on the cloud reveal that the user likes videos and articles. In contrast, the device, which has access to the latest user behaviors, can detect the user has recently clicked on pictures, possibly implying that the user has developed new interests in pictures.

With the sustained expansion of computational power and storage capacity on mobile devices, edge computing makes it possible to perform deep network inference or even training on edge devices. This leads to the emergence of on-device recommendation [7, 13, 14, 41, 42], where the model deployed on the device can access the full user behaviors, allowing it to track real-time changes in user interests and respond to users immediately. Therefore, we propose to put the integrated re-ranking model on the device for the full exploitation of real-time behaviors. The workflow is shown in Figure 1. The cloud server delivers the integrated re-ranking model and candidate items to the edge devices. The model on the device then performs the integrated re-ranking with the aid of real-time behaviors and displays the re-ranked list to the user. Although edge computing is appealing, it is non-trivial to deploy the integrated ranking on the device with the following two main challenges.

First of all, it is challenging to extract users' preferences for different sources from heterogeneous and imbalanced user behaviors. Users' preferences for different sources are generally two-fold: the high-level preferences toward different sources and the fine-grained preferences within each individual source [15]. For example, a user may prefer videos to articles. In the meantime, she may favor food in the video source while preferring social news in the article source. We refer to the two-fold preferences as **source-level preferences**. However, previous studies directly transform the heterogeneous behaviors into a single preference vector [22, 37, 40], which is incomplete and largely overlooks the fine-grained user preferences within each source. More importantly, previous work had not addressed the issue of imbalanced user behaviors. Users may have abundant behaviors in some sources (i.e., *warm source*) while having very sparse behaviors in others (i.e., *cold source*), making it especially difficult to learn users' actual preferences for the sparse sources. In fact, the absence of cold source items in the user history does not necessarily means that the user is uninterested in the source. It is probably because the user has not been recommended items from the cold source recently. Therefore, it is beneficial to exploit the correlation between different sources, which can help infer users' preferences for cold sources from warm sources.

The second challenge is how to effectively explore the correlation between the extracted personalized preferences and the candidate items for integrated re-ranking. For one thing, the candidate items come from different sources, and the user's preferences for each source contribute differently to the candidate items. For instance, users' preferences for the video source are more influential for video candidates, but are less useful for article candidates. Hence, it is essential for integrated re-ranking to conduct **source-level alignment** to align source-level preferences with the corresponding candidate items. Existing work [12, 22, 37], however, simply concatenates the candidate item with a vector that has mixed preferences from multiple sources, which may introduce distractions and thereby yield sub-optimal performances. For another, as the model is placed closer to the user in edge computing, the expectation for personalization on the device is much higher than that for the cloud. Yet the common practice in the industry, on-device inference framework, is to first train the model on the cloud and then send it to the edge devices for inference [13, 14]. Although the resource consumption for the device is reduced, it uses the same set of model parameters to serve different users, which limits the potential to provide tailored models for each user. As users' preferences contain rich personalized information, we propose to inject the preferences into the model parameters for candidate items in order to provide a customized model and real-time personalization for each user within the on-device inference framework.

Another important issue in integrated re-ranking is managing the balance between the total utility of the platform and the exposure fairness amongst different sources. Generally, the primary goal of a recommendation system is to maximize the overall utility (e.g., click-through rates, conversion rates, revenue) and to provide the user with a satisfactory experience. Yet, the system is accountable not only to the user, but also to the item sources in the integrated re-ranking. Each source is expected to receive fair exposure. The exposure determines the economic opportunities (revenue) for each item source, and the unfair distribution of exposure can harm underexposed sources and ultimately impair the long-term utility of the platform. In this work, we aim to boost the overall utility of the integrated re-ranking while considering exposure fairness.

To address the aforementioned issues, we propose an **on-Device Integrated Re-ranking (DIR)** framework, which achieves *real-time heterogeneous behavior modeling* and *preference-adaptive integrated re-ranking* without additional requests to the cloud servers. To begin with, we handle user heterogeneous behaviors as multiple sequences and propose a *multi-sequence behavior modeling* module to extract the user's source-level preferences. We first introduce intra-source interaction to acquire the user's preference for each source and then use representation sharing to infer the cold source preferences from the warm ones. Afterward, to learn the personalized correlation between source-level preferences and candidate items, we present the *preference-adaptive re-ranking* module, which includes a source-level fusion for source-level alignment and a preference-adaptive generator that converts the user's preferences into personalized parameters. We also design utility loss and exposure loss to jointly optimize the overall utility and exposure fairness. Our main contributions can be summarized as follows:

- We present the on-device integrated re-ranking framework DIR to overcome the intrinsic limitations of conventional cloud-based solutions—delays in processing real-time user behaviors and system feedback. To the best of our knowledge, this is the first deep integrated re-ranking model on devices.
- We model the heterogeneous and imbalanced history as multiple sequences and propose a multi-sequence behavior modeling module to extract users' source-level preferences and address the problem of cold sources.
- We address the drawback of on-device inference frameworks, the lack of personalization caused by the same model trained on the cloud, with personalized parameters. For this, we introduce preference-adaptive re-ranking that injects users' source-level preferences into personalized parameters to re-rank candidate items from different sources.

Additionally, we design a utility and exposure loss function to balance the tradeoff between utility and exposure fairness. Extensive experiments conducted on two benchmark datasets and an industrial dataset demonstrate the effectiveness of device-based models and that our proposed DIR outperforms the state-of-the-art algorithms in both utility-based and fairness-based metrics.

2 RELATED WORK

2.1 On-Device Recommendation

With the growth of computing power on edge devices, the on-device recommendation has been drawing increasing attention. We can classify it into on-device inference and on-device learning, according to whether the device undertakes the training task [32].

2.1.1 On-device inference. On-device inference trains the model on the cloud and performs the inference on the devices. This approach takes advantage of the real-time feedback on the device while avoiding too much resource consumption. Owing to the limited storage resource of devices, the models deployed on the devices are usually lightweight models or structures with special designs for the big embedding table. For instance, EdgeRec [14] splits the model into two parts, leaving the embedding on the cloud and the upper model on the device. SVR [13] designs a tiny model, which discards the big embedding table and chooses the most important features. Other approaches focus on model compression, especially embedding compression, such as TT-Rec [43] using tensor-train decomposition and RELU [6] utilizing a set of embedding blocks to design elastic embedding. There are also some methods that explore the incorporation of embedding compression and knowledge distillation, such as SKD [38], EODRec [39], and LLRec [35].

2.1.2 On-device learning. On-device learning involves training on edge devices in exchange for better personalization. An important example is Federated Learning [20], which keeps the individual data on devices and only communicates gradients or parameters via the cloud. There are plenty of works in Federated Learning, but they mainly focus on privacy protection, which is orthogonal to the topic of this work. Other studies are insensitive to privacy and concentrate on improving the efficiency of on-device learning with device-cloud collaboration [7, 26, 41, 42]. For example, Colla [26] allows the cloud and devices to learn collectively by preserving a tailored model for each device and aggregating the device-learned knowledge on the cloud. MPDA [41] conducts device-cloud collaboration under domain adaption, which retrieves similar data from the large-scale cloud's global pool to augment the user's local data.

On-device learning enables users to customize their own model, but it has a large among of time and resource consumption. In addition, it is more demanding for user devices and may merely cover limited users (i.e., high-quality edge TPU/CPU). Hence the main solutions implemented in industrial recommender systems adopt on-device inference [13, 14]. In this work, we also embrace the on-device inference framework, whilst overcoming its drawback of insufficient personalization by integrating personalized parameters.

2.2 Re-ranking

Re-ranking stage re-arranges the initial lists from the previous ranking stage and presents the refined lists to the users [25]. We group the re-ranking into single-source re-ranking and multi-source re-ranking depending on the number of the input initial lists.

2.2.1 Single-source re-ranking. Single-source re-ranking is a relatively well-studied area of re-ranking. It inputs a single list of homogeneous items, emphasizing the cross-item influence within the list. Various network structures, such as RNN [2, 4], Transformer [21, 29, 30, 36], and GNN [24], have been explored for modeling the cross-item influence. For example, DLCM [2] employs GRU to encode the top-ranked list into item representations. PRM [30] and SetRank [29] adopt self-attention to model the mutual influences between any pair of items. MIR [36] introduces multi-level interaction between the user behavior list and the candidate item set. IRGPR [24] exploits graph neural networks to aggregate information from neighborhoods and explicitly model item relationships.

Most single-source re-ranking methods cannot be applied directly to integrated re-ranking. This is because most of them require input lists with a good initial ranking order, whereas the input to integrated re-ranking is unsorted between multiple sources. A few methods insensitive to initial orderings, such as SetRank [29] and MIR [36], could be adapted in integrated re-ranking, but the lack of source-level information may lead to underperformance.

2.2.2 Multi-source re-ranking. Multi-source re-ranking, a.k.a. the integrated re-ranking, is an emerging field in recent years. Currently, the dominant approach in this field is the Reinforcement Learning (RL) method, and their settings are not entirely consistent. Some work in ad allocation requires the mixing of ads and organic items in feed under the order-preservation constraint [22, 37], which means only the sources need to be arranged. For instance, Cross-DQN [22] crosses the embeddings of items and models the crossed sequence by multichannel attention, to extract the arrangement signal. Other studies aim to integrate re-rank lists from more sources without order-preservation constraint [12, 40]. For example, HRL [40] is a hierarchical RL method, with a low-level agent as the channel selector and a high-level agent as the item recommender.

Besides, rank aggregation also aims to consolidate multiple rankings into a single ranking [3], sometimes with constraints like diversity and fairness [31]. This line of work differs from the setting of our integrated re-ranking in terms of its input. For rank aggregation, its input is different rankings generated by multiple base rankers applied to the same candidate set. In other words, each item

in the candidate set receives scores given by different base rankers. This is in contrast to integrated re-ranking that involves multiple base rankers producing rankings for heterogeneous candidate sets from diverse sources. For instance, video candidates are ranked by one ranker while articles are ranked by another. Its objective is to blend items from the two types of candidate sets. Though sharing some similarities, these two problems are orthogonal to each other.

These approaches do not adequately explore the critical role of source-level preferences embodied in user history for integrated re-ranking. Furthermore, these RL-based methods have a high time complexity in inference and cannot meet the requirement of real-time inference on the device, so they are unsuitable for on-device integrated re-ranking. Different from the above work, our focus is to design a model that can fit on mobile devices and exploit source-level preferences for integrated re-ranking.

3 PROBLEM FORMULATION

In this section, we formalize the definition of the on-device integrated re-ranking task. In integrated re-ranking, the model takes as input the initial ranking lists of different sources generated by different rankers, and outputs a recommendation list with items from different sources intermixed. However, existing work has placed the integrated re-ranking model on the cloud, which may not acquire user behavior in real time due to network bandwidth and latency. On-device integrated re-ranking, furthermore, deploys the well-trained model on the device and mixes the candidate items from the cloud based on the user's real-time behaviorse on the device.

Formally, for a user u, the cloud server delivers m candidate lists $S_1, S_2, \ldots, S_m$ from different sources to the device side, where S_i, $i = 1, \ldots, m$, denotes the candidate list of the i-th source and m is the number of sources. The device side receives the candidate list set and collects the user's real-time behavior history $H_1, H_2, \ldots, H_m$ where $H_i, i = 1, \ldots, m$, denotes the history item list recently clicked by user u in the i-th source. The problem of on-device integrated re-ranking refers to the device side employing the real-time user history to mix the m candidate lists $S_1, S_2, \ldots, S_m$ and generate a single integrated list R of K items.

The basic objective of integrated re-ranking is to optimize the overall utility by considering the user's real-time preferences and listwise context. Here, the utility is defined by the expected sum (or weighted sum) of the click[1] probability for each item in the mixed list R, e.g., the total number of clicks or the total revenue. Additionally, in most cases for industrial applications, there is a requirement for exposure fairness in the integrated re-ranking [22, 37]. Let $\mathbf{w} = [w_1, w_2, \ldots, w_m] \in \mathbb{R}^m$ be the given target exposure distribution of m sources, where $w_i \geq 0, i = 1, \ldots, m$, denotes the desired exposure ratio for the i-th source and $\sum_{i=1}^{m} w_i = 1$. We expect the number of exposure for each source in the final recommended list to be proportional to the target exposure distribution $\mathbf{w}$. With this setting, integrated re-ranking is required to generate a mixed list R that balances the overall utility and exposure fairness.

4 THE PROPOSED FRAMEWORK

In this section, we present the details of our proposed DIR. As illustrated in Figure 2, we introduce three components: heterogeneous encoding module, multi-sequence behavior modeling module, and preference-adaptive re-ranking module, accordingly.

To begin with, the heterogeneous encoding module transforms heterogeneous items from different sources to the same feature space. Next, the multi-sequence behavior modeling module extracts users' source-level preferences from their historical behaviors in various sources. It first applies intra-source interaction to acquire users' preferences for different sources separately, and then transfers the preferences from warm sources to cold sources through linear and high-order representation sharing. Lastly, the preference-adaptive re-ranking module will fuse the extracted source-level preferences into the reranking of candidate items. It begins with a listwise context modeling of self-attention, and subsequently, the source-level fusion combines the candidate items and the user's preferences for the corresponding source. In addition, a preference-adaptive generator will produce a set of personalized parameters for the final MLP, enhancing the personalization of the model.

4.1 Heterogeneous Encoding Module

The inputs to DIR are the m candidate lists $S_1, \ldots, S_m$ from different sources and m real-time user behavior lists $H_1, H_2, \ldots, H_m$ of user u. As the features of items from various sources may be heterogeneous and lie in distinctive feature spaces (e.g., videos and articles), we apply different encoding layers to transform these heterogeneous item features into the same feature space. The implementation of the encoder layer is flexible and can be varied depending on the feature types of different sources. For example, suppose the features of m sources are all categorical. In that case, we could maintain m separate embedding layers to obtain the low-dimensional dense embedding vectors and m MLPs to map the embedding vectors to the same feature space. After the encoding, we derive the representation for each item in candidate lists $S_1, \ldots, S_m$ and user behavior history $H_1, \ldots, H_m$. Specifically, let $\mathbf{x}_{i,j} \in \mathbb{R}^{d_x}$ and $\mathbf{h}_{i,j} \in \mathbb{R}^{d_h}$ be the representation of the j-th item in the candidate list S_i of the i-th source and the representation of the j-th item in the user history H_i of the i-th source, respectively. Here, d_x and d_h denote the dimension of representation for candidate items and history items. Then, we obtain the candidate representation matrix $\mathbf{X} \in \mathbb{R}^{mn \times d_x}$ and history representation matrices for m source $\mathbf{H}_1, \ldots, \mathbf{H}_m$ with the dimension of $\mathbf{H}_i$, $i = 1, \ldots, m$, being $t \times d_h$. Here, m, n, and t are the number of sources, the maximum number of items in each candidate list for each source, and the maximum length of user history for each source, respectively.

4.2 Multi-sequence Behavior Modeling

Users' history behaviors contain rich information for inferring their personal preferences and tastes [21, 36]. The intention of on-device integrated re-ranking is also to leverage real-time user behaviors to deduce users' current interests more accurately. Thus, how to model user behaviors in real time becomes a critical issue. In integrated re-ranking, users' historical behaviors from various sources naturally constitute a multi-sequence history. Yet previous studies usually re-assemble the multi-sequence history into a single sequence.

Applying one single sequence for behavior modeling may suffer from the following two defects. First, the user has source-level preferences, where the user not only has preferences for different

[1]The click could be replaced by other utility metrics like conversions, purchases, etc.

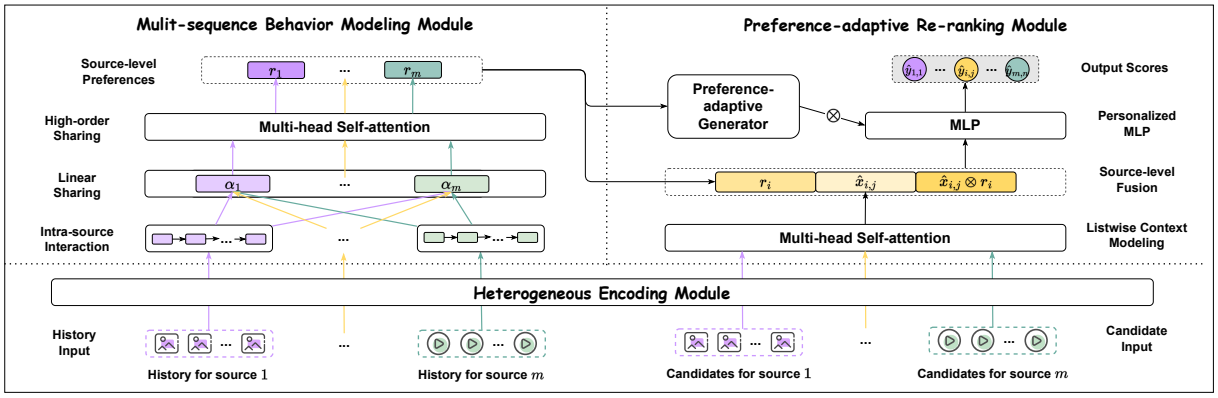

Figure 2: The overall framework of DIR.

sources, but also has fine-grained preferences within sources that may be interrelated or mutually exclusive. The single sequence may overlook or confound these fine-grained preferences. Second, historical behaviors from various sources are often imbalanced. Cold sources may have extremely sparse behaviors, and are overwhelmed by other warm sources. Prior work usually ignores this issue and fails to uncover users' preferences for cold sources. In this work, we adopt a more spontaneous approach, **multi-sequence behavior modeling**, with *intra-source interaction* to extract users' fine-grained preferences for each source and *representation sharing* to mitigate the problem of the imbalanced history.

4.2.1 Intra-source Interaction. This part is designed to capture users' fine-grained preferences within each source. Its input is the user's history lists in various sources, which carry the temporal pattern of the user's interests and preferences. To better leverage the temporal pattern, we employ GRU [8], a commonly used lightweight network for sequence data. User behavior can vary with sources, so we use separate GRUs for different sources for modeling users' evolving interests, and we have $< \widehat{\mathbf{H}}_i, \mathbf{p}_i >= \text{GRU}_i(\mathbf{H}_i),\ i = 1, \ldots, m$, where $\mathbf{H}_i = \{\mathbf{h}_{i,1}, \ldots, \mathbf{h}_{i,t}\}$ denotes the representation of items in user history from the i-th source, $\widehat{\mathbf{H}}_i = \{\widehat{\mathbf{h}}_{i,1}, \ldots, \widehat{\mathbf{h}}_{i,t}\}$ denotes the output sequence of GRU encoding, and $\mathbf{p}_i \in \mathbb{R}^{d_p}$ of size d_p is the output final state of GRU. Here, we consider $\mathbf{p_i}, i = 1, \ldots, m$, as the user's fine-grained preferences for the i-th sources since the final state contains the information of the whole sequence. Note that GRU could be replaced by other networks for sequential data, such as LSTM [16] and self-attention with positional embedding [34].

4.2.2 Representation Sharing. Handling user history as multiple sequences makes it straightforward to obtain fine-grained preferences of users for individual sources, but does not intuitively solve the problem of imbalanced history. In particular, if the user has no history in a specific source, the corresponding preference representation obtained in intra-source interaction would be empty, which prevents us from extracting information about the source. To this end, we propose the **representation sharing** module that allows preference representations to be shared by means of linear and higher-order interactions. In this way, user behavioral information in the warm sources can then be transferred to the cold sources for extrapolating the user's preferences in the cold sources.

First, we conduct representation sharing through linear combination motivated by Cross-Stitch [27], which can automatically learn an optimal combination of shared and task-specific representations. Specifically, this linear combination is computed as follows

$$\mathbf{V} = \begin{bmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_m \end{bmatrix} = \begin{bmatrix} \alpha_{1,1} & \ldots & \alpha_{1,m} \\ \vdots & \ddots & \vdots \\ \alpha_{m,1} & \ldots & \alpha_{m,m} \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_m \end{bmatrix}, \tag{1}$$

where $\mathbf{p}_i \in \mathbb{R}^{d_p}$ and $\mathbf{v}_i \in \mathbb{R}^{d_p}, i = 1, \ldots, m$, are the preference representations obtained in intra-source interaction and after the linear sharing. The sharing coefficient $\alpha_{i,j} \in \mathbb{R}, i = 1, \ldots, m, j = 1, \ldots, m$, indicates the extent to which $\mathbf{p}_i$ contributes to $\mathbf{v}_j$. In this way, each preference is expressed as a combination of other preferences, subsequently ensuring that they interact with each other and preferences in the cold sources are not empty.

However, the linear sharing described above is low-dimensional and might be insufficient to extract interactions between preferences, so we introduce self-attention to achieve high-order representation interaction [33]. First, we concatenate the preferences before and after linear sharing to get the input of high-order sharing, that is $\hat{\mathbf{v}}_i = [\mathbf{p}_i \oplus \mathbf{v}_i],\ i = 1, \ldots, m$, where $\oplus$ is the concatenation operation. Then, we stack all the $\hat{\mathbf{v}}_i \in \mathbb{R}^{2d_p}, i = 1, \ldots, m$, and obtain a matrix $\widehat{\mathbf{V}} \in \mathbb{R}^{m \times 2d_p}$ as the input of self-attention, so we get

$$\mathbf{R} = \text{Attention}(\widehat{\mathbf{V}}) = \text{softmax}\left(\frac{(\widehat{\mathbf{V}}\mathbf{W}_Q)(\widehat{\mathbf{V}}\mathbf{W}_K)^\top}{\sqrt{d_a}}\right)(\widehat{\mathbf{V}}\mathbf{W}_V), \tag{2}$$

where $\mathbf{W}_Q \in \mathbb{R}^{2d_p \times d_a}$, $\mathbf{W}_K \in \mathbb{R}^{2d_p \times d_a}$, and $\mathbf{W}_V \in \mathbb{R}^{2d_p \times d_a}$ denotes the parameter matrices of projection. The output $\mathbf{R} \in \mathbb{R}^{m \times d_a}$ is the attended matrix and $\sqrt{d_a}$ is used to stabilize gradients during training. In terms of implementation, we apply the multi-head mechanism to increase the stability of the self-attention network. Finally, We get the user's source-level preferences $\mathbf{r}_i \in \mathbb{R}^{d_a}$ in the i-th source, that is the i-th row of attended matrix $\mathbf{R}$. It is worth noting that both low-level and high-level representation sharing are essential to ensure that cold sources can gain useful information about the user's preferences from warm sources. Supposing only the self-attention mechanism is used and yet some sources have

no historical behavior at all, the extracted preferences will still be empty, and representation sharing cannot be accomplished.

4.3 Preference-adaptive Re-ranking

Having extracted the user's source-level preferences from the real-time behaviors, we now need to blend the multiple candidate lists of different sources into one mixed re-ranked list, while considering the personalized source-level preferences. The user's feedback on an item in the mixed list depends not only on the current item but also on the other items in the list. As such, the listwise context of the whole list should also be factored in. In addition, the expectation for personalization at the edge devices is a lot higher than that for the cloud. Previous on-device inference methods [13, 14] fail to achieve good personalization, since they use the same set of model parameters for different users. Given that the user's preferences contain rich personalized information in real time, we inject it into the model parameters for candidate items in order to provide a customized model that adapts to real-time and instant user interests. Thus, we introduce a preference-adaptive re-ranking module, with self-attention for the listwise context and a preference-adaptive generator to generate personalized parameters.

To begin with, we draw on techniques for listwise context modeling from conventional re-ranking [29, 30]. We adopt the self-attention mechanism, which can model mutual influences between any two items directly and is proved to be permutation-equivalent [29]. Taking the candidate representation matrix $\mathbf{X} \in \mathbb{R}^{mn \times d_x}$ obtained in the encoding module, we can obtain the attended matrix $\widehat{\mathbf{X}} = \text{Attention}(\mathbf{X})$ through the similar process in Eq. (2). Then, we reshape the attended matrix $\widehat{\mathbf{X}} \in \mathbb{R}^{mn \times d_a}$ into $\widehat{\mathbf{X}} \in \mathbb{R}^{m \times n \times d_a}$ and unstack the matrix to obtain $\hat{\mathbf{x}}_{i,j}$, $i = 1, \ldots, m$, $j = 1, \ldots, n$, the interacted representation of the j-th item in the i-th source.

Aside from the interactions within the candidate set, it is also necessary to align the candidate item with the user's preferences to avoid confusion of the preferences among different sources. Accordingly, we devise source-level alignment to unite each item $\hat{\mathbf{x}}_{i,j}$ and the user's preference $\mathbf{r}_i$ for the associated source i

$$\mathbf{o}_{i,j} = [\hat{\mathbf{x}}_{i,j} \oplus \mathbf{r}_i \oplus (\hat{\mathbf{x}}_{i,j} \odot \mathbf{r}_i)] , \tag{3}$$

where $\oplus$ and $\odot$ denote the concatenation and element-wise product operation. The element-wise product introduces the second-order interaction between items and preferences.

Lastly, multi-layer perception (MLP) is integrated into the output layer. We feed each item embedding $\mathbf{o}_{i,j}$ into a shared MLP to obtain the final re-ranking score

$$\hat{y}_{i,j} = \text{MLP}(\mathbf{o}_{i,j}) , \tag{4}$$

where $\hat{y}_{i,j} \in \mathbb{R}$ is the score for the j-th item in the i-th source.

So far, we have not covered the personalization of the model parameters. Motivated by [9], we utilize a preference-adaptive generator $g(\mathbf{R})$ to generate a set of personalized parameters with preferences embodied in historical behaviors. Here, $\mathbf{R} = \{\mathbf{r}_i, \ldots, \mathbf{r}_m\}$ is the user's source-level preferences for m sources that we obtained in Section 4.2. The design of the preference-adaptive generator is flexible. Here we adopt a straightforward way of concatenating and feeding the preferences for m sources into an MLP to produce customized model parameters, i.e.,

$$g(\mathbf{R}) = \text{MLP}([\mathbf{r}_1 \oplus \mathbf{r}_2 \oplus \cdots \oplus \mathbf{r}_m]) . \tag{5}$$

The dimension of $g(\mathbf{R})$ depends on the modules to which it applies. Taking the fully connected (FC) layer as an example, its personalized version is formulated as

$$\hat{\mathbf{z}} = g(\mathbf{R}) \odot \sigma(\mathbf{W}\mathbf{z} + b) , \tag{6}$$

where $\mathbf{W}$ and b denote the weight parameters, $\mathbf{z}$ and $\hat{\mathbf{z}}$ are the input and output of the FC layer, and $\sigma(\cdot)$ is the activation function. In this case, $g(\mathbf{R})$ has the same dimension as $\hat{\mathbf{z}}$ and acts as gating weights to mask the corresponding parameters, which is personalized and adapts to the user's preferences. Naturally, the personalized version of MLP is formed by stacking the personalized FC layers, with each layer applying part of the parameters in $g(\mathbf{R})$. In practice, however, we only employ the personalized FC in the last layer of the MLP in Eq.(4). The benefits of personalization via preference-adaptive generator are notable. Unlike meta-learning methods, which manually split training data into support/query sets and probably require multiple training rounds, the parameter generator follows an end-to-end training procedure. Moreover, its computational overheads are negligible. Only an additional element-wise multiplication is involved, which is computationally efficient.

4.4 Optimization

As mentioned in Section 3, there are two objectives in integrated re-ranking. One is to guarantee the accuracy of the recommendation and optimize the overall utility, and the other is exposure fairness that the actual exposure in each source is expected to approximate the desired exposure distribution. Consequently, we design utility loss and exposure loss to achieve the two objectives.

Utility loss. For utility loss, we adopt the common cross-entropy loss. Assuming that the label $y_{i,j}$ represents whether the j-th candidate item from the i-th source is clicked by user u, we can obtain

$$\mathcal{L}_{util} = -\frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \sum_{i}^{m} \sum_{j}^{n} y_{i,j} \log \hat{y}_{i,j} + (1 - y_{i,j}) \log(1 - \hat{y}_{i,j}) \tag{7}$$

where $\hat{y}_{i,j}$ is the predicted score for the j-th candidate item from the i-th source for the user u, and $\mathcal{U}$ is the user set.

Exposure loss. As for exposure loss, we assume that the first K positions of the re-ranked list will be exposed. Therefore, we first sort the items in the candidate lists by the prediction score $\hat{y}_{i,j}$, $i = 1, \ldots, m$, $j = 1, \ldots, n$ and then calculate the distribution of items from m sources at the top K positions of all the re-ranked lists, which is the actual accumulated exposure distribution for each source. We denote it as $\mathbf{w}^p = [w_1^p, \ldots, w_m^p]$, where $w_i^p \in \mathbb{R}^*$, $i = 1, \ldots, m$ represents the exposure ratio of the i-th source. The actual exposure distribution is expected to achieve the target exposure distribution $\mathbf{w} = [w_1, \ldots, w_m] \in \mathbb{R}^m$. Commonly used measures of similarity between probability distributions are Kullback–Leibler (KL) divergence [19] and Jensen–Shannon (JS) divergence [23]. Because JS divergence is symmetric and more appropriate than the asymmetric KL divergence, we utilize JS divergence $\mathcal{D}_{JS}$, as presented in Eq.(8)

$$\mathcal{L}_{JS} = \mathcal{D}_{JS}(\mathbf{w} || \mathbf{w}^p) , \tag{8}$$

Combining Eq.(7) and Eq.(8), the loss for DIR is:

$$\mathcal{L} = \beta \mathcal{L}_{util} + (1 - \beta)\mathcal{L}_{JS}, \tag{9}$$

where β denotes the hyper-parameter that controls the tradeoff between the utility and the exposure loss. In the implementation, the calculation of $\mathbf{w}^p$ entails sorting the candidate lists, which is non-differentiable. Thus, we use a simple reparameterization trick [17], a common technique to pass derivatives when dealing with non-differentiable operations, to approximate the sorting operation.

5 EXPERIMENT

This section first compares DIR with state-of-the-art baselines. Then, we investigate the impact of several components and hyper-parameters, followed by a case study to evaluate the performance of DIR in addressing cold source and personalization issues.

5.1 Experimental Setup

5.1.1 Datasets. Our experiments are conducted on two public datasets, Taobao Cloud Theme Click Dataset[2] and Taobao Open MCC Dataset[3], and a proprietary dataset, Celia Suggestions.

- **Taobao Open MCC Dataset** [42] (**MCC** for short) is collected from Mobile Taobao App, containing 13,081,990 click records of randomly sampled 7,995,248 users. Each record includes not only the features of the current item, but also the user's real-time click behaviors. The items in this dataset can be classified into three categories, so we use the category as the source following [12].
- **Taobao Cloud Theme Click Dataset** [10] (**CTC** for short) records the click data of Cloud Theme, an important recommendation procedure in Taobao App. The dataset includes 1,423,835 click records under different themes during a 6-day promotion season, users' purchase history in one month before the promotion, and the embedding of 720,210 users and 1,361,672 items. Here, we select three most frequently interacted themes as sources.
- **Celia Suggestions Dataset** contains 50 million ranking lists along with associated real-time APP/Service click history during two days from Celia Suggestions of Huawei. The candidates come from two different sources, installed Apps (e.g., TikTok, Instagram) and Services (atomic function, e.g., News browsing service, QR code scanning service). Different sources have distinctive display formats and sizes.

More description about history and candidate generation as well as reproducibility can be found in Appendix A.

5.1.2 Baselines. Since there is no previous work in the field of on-device integrated re-ranking, we compare DIR with the state-of-the-art models in single-source re-ranking and integrated re-ranking based on the cloud, as well as on-device re-ranking or recommendation. We employ three single-source re-ranking baselines. **DIN** [44] is a click-through rate prediction model that captures users' interests to target items from historical behaviors. **SetRank** [29] re-orders items by learning their permutation-equivariant representations via self-attention. **MIR** [36] re-ranks items with multi-level interaction between user history list and candidate item set. In integrated re-ranking, we adopt **HRL** [40], which proposes a hierarchical RL framework to recommend channels and items sequentially. As for on-device recommendation, we use **EdgeRec** [14] which makes the first attempt to implement the recommender system on the device and **SVR** [13] that deploys an on-device context-aware ranking model for short video recommendations.

Except for the on-device recommendation models that have already been put on the device (EdgeRec and SVR) and SetRank where no history behavior is utilized, we implement both the device and cloud version of the baselines for comparison. For example, **DIN-D** and **DIN-C** denote DIN models that use device and cloud history in our experiment, respectively.

5.1.3 Metrics. As the goal of the integrated re-ranking is to maximize the utility and balance the exposure fairness, we evaluate all the models in terms of utility-based and fairness-based metrics.

- **Utility-based metric**: Following [13, 36, 40], several widely used ranking metrics, *AUC* [11], *NDCG@K* [18], and *MAP@K*, are adopted. Moreover, we employ *utility@K*, the average number of clicks on the top-K recommended items, following [37]. The metrics are computed with clicks in log data.
- **Fairness-based metric**: Following [28], we apply $\mathcal{JS}@K$ to indicate how close the exposure distribution on sources is to the ideal distribution. Formally, $JS@K = \mathcal{D}_{JS}(\hat{\pi}_K || \pi)$, where $\hat{\pi}_K$ denotes the cumulative exposure distribution on all the top-K recommended items, π is the ideal distribution, and $\mathcal{D}_{JS}$ is JS divergence. The lower value of $JS@K$ means the better performance of a model on exposure fairness.

On the three datasets, we assume that the top-K items will be exposed and set K to 5. As for the ideal distribution π in the fairness metric, we set it roughly to the original proportion of the number of items that each source owns to ensure a fair comparison. Specifically, π is set to $(0.5, 0.4, 0.1)$, $(0.33, 0.33, 0.34)$, and $(0.2, 0.8)$ on the MCC, CTC, and Celia Suggestions dataset, respectively.

5.2 Overall Performance

The overall performance of our proposed DIR and baselines on the MCC, CTC, and Celia Suggestions datasets is reported in Table 1, from which we have the following observations.

Firstly, our proposed DIR significantly and consistently outperforms the state-of-the-art approaches on the three datasets. As presented in Table 1, DIR obtains the best performance with respect to utility-based metrics, *AUC*, *NDCG*, *MAP*, and *utility*, as well as fairness-based metric $\mathcal{JS}$. For instance, DIR achieves 4.53%, 9.03%, 7.01%, and 3.23 % improvement over baseline MIR-D in *AUC*, *MAP*, *NDCG*, and *utility* on MCC dataset. On the CTC dataset, DIR also surpasses the strongest baseline SVR by 4.48% in *AUC*, 5.27% in *MAP*, 4.03% in *NDCG*, and 2.22% in *utility* on CTC dataset. This illustrates the huge potential of bringing edge computing to integrated re-ranking and capturing users' real-time interests via real-time historical behaviors. We also attribute this improvement to DIR successfully extracting the user's source-level preference and personalizing it with candidate items from each source. With regard to the fairness-based metric $\mathcal{JS}$, DIR turns out to be even more effective, with improvements of 66.7%, 81.0%, and 58.8% gained on MCC, CTC, Celia Suggestions dataset over the strongest baselines. This

[2]https://tianchi.aliyun.com/dataset/9716
[3]https://tianchi.aliyun.com/dataset/109858

Table 1: Overall performance on three datasets.

Model	Taobao Open MCC (MCC)					Taobao Cloud Theme Click (CTC)					Celia Suggestions				
	AUC	JS	MAP	NDCG	utility	AUC	JS	MAP	NDCG	utility	AUC	JS	MAP	NDCG	utility
SetRank	0.6723	0.0028	0.3809	0.4659	0.7251	0.6112	0.0042	0.3525	0.4226	0.6367	0.8713	0.00062	0.8441	0.8203	2.2736
DIN-C	0.6943	0.0032	0.4141	0.5010	0.7657	0.5831	0.0073	0.3193	0.3890	0.6042	0.8574	0.00035	0.8431	0.8131	2.2418
DIN-D	0.7219	0.0021	0.4658	0.5502	0.8060	0.6300	0.0053	0.3734	0.4452	0.6641	0.8586	0.00038	0.8449	0.8150	2.2444
MIR-C	0.7134	0.0026	0.4291	0.5131	0.7685	0.5919	0.0038	0.3230	0.3917	0.6029	0.8837	0.00024	0.8611	0.8354	2.2952
MIR-D	0.7790	0.0027	0.5108	0.5955	0.8513	0.6135	0.0038	0.3501	0.4201	0.6341	0.8972	0.00026	0.8746	0.8515	2.3447
HRL-C	0.7232	0.0022	0.4798	0.5530	0.7744	0.6029	0.0023	0.3545	0.4207	0.6230	0.8646	0.00017	0.8520	0.8179	2.2557
HRL-D	0.7500	0.0025	0.5091	0.5834	0.8086	0.6334	0.0021	0.3704	0.4411	0.6562	0.8773	0.00017	0.8601	0.8303	2.2892
EdgeRec	0.7232	0.0050	0.4346	0.5227	0.7906	0.6195	0.0047	0.3554	0.4325	0.6680	0.8579	0.00018	0.8427	0.8140	2.2460
SVR	0.6907	0.0028	0.4075	0.4934	0.7551	0.6402	0.0033	0.3737	0.4564	0.7083	0.8558	0.00027	0.8424	0.8115	2.2360
DIR	**0.8143***	**0.0007**	**0.5569***	**0.6372***	**0.8788***	**0.6689***	**0.0004**	**0.3934***	**0.4748***	**0.7240***	**0.9135***	**0.00007**	**0.8911***	**0.8723***	**2.3925***

$*$ denotes statistically significant improvement (t-test with p-value < 0.05) over the best baseline. Note that this is not available for $\mathcal{JS}$ metric computed in a cumulative way.

Table 2: Comparison of DIR and its variants on three datasets.

Variant	Taobao Open MCC (MCC)					Taobao Cloud Theme Click (CTC)					Celia Suggestions				
	AUC	JS	MAP	NDCG	utility	AUC	JS	MAP	NDCG	utility	AUC	JS	MAP	NDCG	utility
DIR-ISI	0.7690	0.0007	0.5126	0.5972	0.8526	0.6101	0.0005	0.3369	0.4096	0.6315	0.9060	0.00004	0.8850	0.8639	2.3705
DIR-LRS	0.7901	**0.0006**	0.5420	0.6230	0.8672	0.6201	0.0004	0.3467	0.4252	0.6654	0.9093	0.00005	0.8860	0.8671	2.3829
DIR-HRS	0.7700	0.0011	0.5197	0.6026	0.8532	0.6431	**0.0003**	0.3537	0.4316	0.6706	0.9095	0.00004	0.8866	0.8674	2.3825
DIR-LCM	0.7936	0.0009	0.5310	0.6125	0.8581	0.6226	0.0004	0.3676	0.4366	0.6471	0.8841	0.00009	0.8649	0.8469	2.3377
DIR-PG	0.7887	**0.0006**	0.5444	0.6249	0.8676	0.6488	0.0004	0.3660	0.4494	0.7044	0.9095	**0.00003**	0.8852	0.8666	2.3831
DIR-JS	**0.8144**	0.0024	**0.5582**	**0.6393**	**0.8833**	**0.6699**	0.0021	0.3859	0.4686	0.7227	**0.9137**	0.00019	**0.8925**	**0.8727**	2.3906
DIR	0.8143	0.0007	0.5569	0.6372	0.8788	0.6689	0.0004	**0.3934**	**0.4748**	**0.7240**	0.9135	0.00007	0.8911	0.8723	**2.3925**

demonstrates the effectiveness of exposure loss and shows that DIR makes a better tradeoff between the overall utility and exposure fairness, achieving better utility and fairness simultaneously.

Secondly, the utilization of real-time behaviors enhances the performance of integrated re-ranking, and the more sufficiently the history is modeled, the greater the enhancement. Even for the same method, the version accessing to the device history with the latest behaviors outperforms the version with the cloud history. For example, DIN-D is superior to DIN-C, and MIR-D is superior to MIR-C. This illustrates the critical role of real-time user behaviors for integrated re-ranking and that it is reasonable to deploy integrated re-ranking on edge devices. In addition, the better the modeling of the history, the greater the boost is likely to be. For instance, MIR, with multi-layer interactions between candidates and history, is usually more effective than DIN with simple attention. DIR that extracts source-level preferences from histories achieves even better results than MIR that does not factor in the source information. This also reveals the importance of the source-level preferences in integrated re-ranking. Thus, EdgeRec and SVR, which only employ single-source modeling, also do not perform very well.

Lastly, approaches designed for integrated re-ranking commonly yield better results on the fairness-based metric. As depicted in Table 1, integrated re-ranking methods HRL and DIR outperform the other non-integrated re-ranking baselines regarding $\mathcal{JS}$. These integrated re-ranking methods usually consider source-level information and are more likely to generate balanced recommendations. Nevertheless, the source-level information is inadequate for the model to achieve the target distribution, so we design exposure loss for exposure fairness in DIR. The huge improvement of DIR over baselines in $\mathcal{JS}$ also demonstrates the effectiveness of our design.

5.3 In-depth Analysis

To investigate the effectiveness of the components in DIR, we design several variants of DIR, which are listed as follows. **DIR-ISI** replaces the GRU used in intra-source interaction with the average pooling. **DIR-LRS** and **DIR-HRS** remove the linear and high-order representation sharing from the multi-sequence behavior modeling module. **DIR-LCM** and **DIR-PG** remove the listwise context modeling of candidate items and the preference-adaptive generator from the preference-adaptive re-ranking module. **DIR-JS** removes the exposure loss. The comparison of the above variants and original DIR on three datasets is shown in Table 2, from which we have the following observations.

Firstly, compared to the original DIR, the performance of all variants declines to some extent, illustrating the effectiveness of each component. DIR-ISI generally suffers the greatest decline on utility-based metrics, indicating that source-level preferences are sensitive to the temporal order. DIR-JS performs poorly on the fairness-based metric, but improves on the utility-based metrics, suggesting that exposure loss can enhance exposure fairness and that there is a tradeoff between fairness and utility. Furthermore, the removals of different components yield different performances on different datasets, probably due to some inherent characteristics of the datasets. As an example, on the Celia Suggestions dataset, the DIR-LCM performs the worst, suggesting that the context of the candidates is more critical for integrated re-ranking than the user history. This is probably because user click behavior is more influenced by listwise context than historical preferences in this dataset, whereas the opposite is the case in the other two datasets.

In addition, more experiments about hyper-parameters and case study can be found in Appendix B and C.

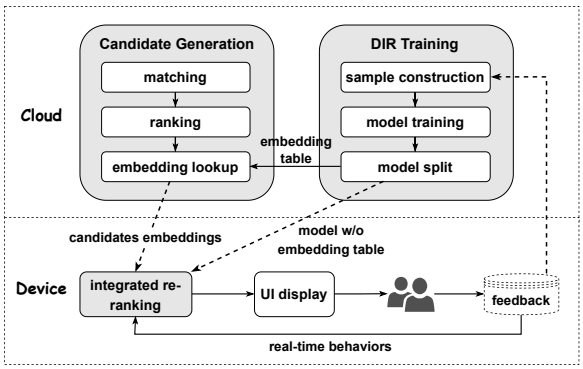

Figure 3: The system implementation of DIR.

6 DEPLOYMENT FEASIBILITY

We adopt the on-device inference approach for implementing DIR, following the practices of EdgeRec [14] and SVR [13], which have been serving the main traffic in Taobao and Kuaishou. As presented in Figure 3, the cloud side is mainly responsible for two tasks: candidate generation and the training of DIR. The cloud constructs training samples with the corresponding user features, item features, and context features, and trains the integrated re-ranking model. After training, the model is split into two parts. The learned embedding table (which usually accounts for more than 95% of the total parameters [5]) remains in the cloud, while the model without embedding table is dispatched to the devices for inference. As such, DIR can take up less storage space of the device without losing model accuracy, as is usually the case with compression methods [6, 43]. Whenever a recommendation is triggered, the cloud side will generate candidate items through two stages: matching and ranking, and then transfer the embedding of the corresponding candidate items found in the embedding table, together with the user and context features, to the devices. Once the device receives embedding and features, it will utilize the user's real-time behaviors and the previously received model to conduct integrated re-ranking, and present the results to the user. As discussed in EdgeRec [14], the split-deployment approach we use above can meet the requirements for resource consumption and system load on devices.

As for the efficiency feasibility, we compare the time complexity of DIR with EdgeRec and SVR, which are currently deployed in Kuaishou and Taobao. Assuming that h and n denote the number of history items and candidate items from all sources, we can obtain that the time complexity of DIR is $O(h + n^2)$. And the time complexity of EdgeRec and SVR are $O(h+hn)$ and $O(khn+kn^2)$, where k is the parameter of beam search in SVR. Considering the number of historical items is usually at the same scale as that of candidate items, the time complexities of the three methods are similar.

Furthermore, we analyze the actual inference time of DIR, SVR, and EdgeRec. Table 3 presents the average inference time of three models which is calculated by dividing the inference time of the entire testset by the total number of samples (lists). As illustrated in Table 3, the inference time of DIR is between EdgeRec and SVR, indicating that the inference speed of DIR is efficient and comparable to that of the methods serving online. The difference in inference speed on the two datasets is probably due to the number of features. Unlike the MCC dataset with only several categorical features, the CTC dataset also includes an additional 128-dimensional dense feature for each item, which may slow down the inference speed.

Table 3: The comparison of inference time (ms).

	CTC	MCC
SVR	3.0742	0.2790
EdgeRec	2.5469	0.1835
DIR	2.6786	0.2154

Additionally, We analyze the improvement in the system efficiency of running the model on edge compared to the cloud from the following two aspects. (i) Less system response time: Due to the computing overhead of serving hundreds of millions of users, the average response time for a cloud model is about 800ms. In contrast, the model running on the user's own device only needs to process the user's own requests, reducing the cost of centralized computing and network communication overhead, thus achieving a response time within 100ms. (ii) Efficiency in leveraging user behaviors: Owing to the limitations of network bandwidth and latency, cloud models usually have a delay of several minutes in collecting user behaviors. Such delay will be even longer to reduce the devices' power consumption as the request of transmitting behavior data to the cloud requires large power consumption. While on the edge side, the user's historical behavior is directly queried from the database on the device, which only requires a delay of milliseconds. Moreover, the power consumption of making a request to the edge is considerably lower than that of the cloud, which is only 1/20th of the power consumption of a single transmission with the cloud. Thus, running the model on edge allows for collecting and leveraging the user's real-time behaviors at a much lower cost. To sum up, on-device inference can more efficiently utilize the user's real-time behaviors.

7 CONCLUSION

In this work, we address the intrinsic limitations of the cloud-based integrated re-ranking – delayed processing of real-time user behaviors, and propose the first on-device integrated re-ranking framework, DIR. We design a multi-sequence behavior modeling module to extract users' source-level preferences and introduce preference-adaptive re-ranking to incorporate the source-level preference into the re-ranking of candidates from different sources. Both the multi-sequence behavior modeling and the personalization for on-device inference can be easily transferred to other domains involving multiple sequences and personalization. Extensive experiments show that DIR significantly outperforms state-of-the-art baselines. In the future, we plan to explore better collaboration between edge and cloud computing. For instance, we can investigate on-device training strategies and adapt the model to various devices with different computational capacities and memory limitations.

ACKNOWLEDGEMENT

The team is supported by Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102), National Natural Science Foundation of China (62177033), and Shanghai Artificial Intelligence Innovation and Development Fund grant 2020-RGZN-02026. The work is also sponsored by Huawei Innovation Research Program. We thank MindSpore [1] for the partial support of this work.

REFERENCES

[1] 2020. MindSpore. https://www.mindspore.cn/

[2] Qingyao Ai, Keping Bi, Jiafeng Guo, and W Bruce Croft. 2018. Learning a deep listwise context model for ranking refinement. In *The 41st international ACM SIGIR conference on research & development in information retrieval*. 135–144.

[3] Javier Alcaraz, Mercedes Landete, and Juan F. Monge. 2022. *Rank Aggregation: Models and Algorithms*. 153–178.

[4] Irwan Bello, Sayali Kulkarni, Sagar Jain, Craig Boutilier, Ed Chi, Elad Eban, Xiyang Luo, Alan Mackey, and Ofer Meshi. 2018. Seq2slate: Re-ranking and slate optimization with rnns. *arXiv preprint arXiv:1810.02019* (2018).

[5] Ting Chen, Lala Li, and Yizhou Sun. 2020. Differentiable product quantization for end-to-end embedding compression. In *International Conference on Machine Learning*. PMLR, 1617–1626.

[6] Tong Chen, Hongzhi Yin, Yujia Zheng, Zi Huang, Yang Wang, and Meng Wang. 2021. Learning Elastic Embeddings for Customizing On-Device Recommenders. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 138–147.

[7] Zeyuan Chen, Jiangchao Yao, Feng Wang, Kunyang Jia, Bo Han, Wei Zhang, and Hongxia Yang. 2021. MC^2-SF: Slow-Fast Learning for Mobile-Cloud Collaborative Recommendation. https://doi.org/10.48550/ARXIV.2109.12314

[8] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. 103–111.

[9] Shangfeng Dai, Haobin Lin, Zhichen Zhao, Jianying Lin, Honghuan Wu, Zhe Wang, Sen Yang, and Ji Liu. 2021. POSO: Personalized Cold Start Modules for Large-scale Recommender Systems. https://doi.org/10.48550/ARXIV.2108.04690

[10] Zhengxiao Du, Xiaowei Wang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Sequential Scenario-Specific Meta Learner for Online Recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2895–2904.

[11] Tom Fawcett. 2006. An introduction to ROC analysis. *Pattern Recognition Letters* 27, 8 (2006), 861–874.

[12] Mingsheng Fu, Anubha Agrawal, Athirai A. Irissappane, Jie Zhang, Liwei Huang, and Hong Qu. 2022. Deep Reinforcement Learning Framework for Category-Based Item Recommendation. *IEEE Transactions on Cybernetics* 52, 11 (2022), 12028–12041. https://doi.org/10.1109/TCYB.2021.3089941

[13] Xudong Gong, Qinlin Feng, Yuan Zhang, Jiangling Qin, Weijie Ding, Biao Li, Peng Jiang, and Kun Gai. 2022. Real-Time Short Video Recommendation on Mobile Devices. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 3103–3112.

[14] Yu Gong, Ziwen Jiang, Yufei Feng, Binbin Hu, Kaiqi Zhao, Qingwen Liu, and Wenwu Ou. 2020. EdgeRec: Recommender System on Edge in Mobile Taobao. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2477–2484.

[15] Wei Guo, Can Zhang, Zhicheng He, Jiarui Qin, Huifeng Guo, Bo Chen, Ruiming Tang, Xiuqiang He, and Rui Zhang. 2022. MISS: Multi-Interest Self-Supervised Learning Framework for Click-Through Rate Prediction. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. 727–740. https://doi.org/10.1109/ICDE53745.2022.00059

[16] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9 (1997), 1735–1780.

[17] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical Reparameterization with Gumbel-Softmax. https://arxiv.org/abs/1611.01144

[18] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-Based Evaluation of IR Techniques. *ACM Trans. Inf. Syst.* (2002), 422–446.

[19] James M. Joyce. 2011. *Kullback-Leibler Divergence*. Springer Berlin Heidelberg, Berlin, Heidelberg, 720–722. https://doi.org/10.1007/978-3-642-04898-2_327

[20] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtarik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated Learning: Strategies for Improving Communication Efficiency. In *NIPS*.

[21] Yi Li, Jieming Zhu, Weiwen Liu, Liangcai Su, Guohao Cai, Qi Zhang, Ruiming Tang, Xi Xiao, and Xiuqiang He. 2022. PEAR: Personalized Re-ranking with Contextualized Transformer for Recommendation. *arXiv preprint arXiv:2203.12267* (2022).

[22] Guogang Liao, Ze Wang, Xiaoxu Wu, Xiaowen Shi, Chuheng Zhang, Yongkang Wang, Xingxing Wang, and Dong Wang. 2022. Cross DQN: Cross Deep Q Network for Ads Allocation in Feed. In *WWW*. 401–409.

[23] J. Lin. 1991. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory* 37, 1 (1991), 145–151.

[24] Weiwen Liu, Qing Liu, Ruiming Tang, Junyang Chen, Xiuqiang He, and Pheng Ann Heng. 2020. Personalized Re-ranking with Item Relationships for E-commerce. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 925–934.

[25] Weiwen Liu, Yunjia Xi, Jiarui Qin, Fei Sun, Bo Chen, Weinan Zhang, Rui Zhang, and Ruiming Tang. 2022. Neural Re-ranking in Multi-stage Recommender Systems: A Review. *arXiv preprint arXiv:2202.06602* (2022).

[26] Yan Lu, Yuanchao Shu, Xu Tan, Yunxin Liu, Mengyu Zhou, Qi Chen, and Dan Pei. 2019. Collaborative Learning between Cloud and End Devices: An Empirical Study on Location Prediction. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*. 139–151.

[27] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert. 2016. Cross-Stitch Networks for Multi-task Learning. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 3994–4003.

[28] Natwar Modani, Deepali Jain, Ujjawal Soni, Gaurav Kumar Gupta, and Palak Agarwal. 2017. Fairness Aware Recommendations on Behance. In *Advances in Knowledge Discovery and Data Mining*, Jinho Kim, Kyuseok Shim, Longbing Cao, Jae-Gil Lee, Xuemin Lin, and Yang-Sae Moon (Eds.). 144–155.

[29] Liang Pang, Jun Xu, Qingyao Ai, Yanyan Lan, Xueqi Cheng, and Jirong Wen. 2020. Setrank: Learning a permutation-invariant ranking model for information retrieval. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 499–508.

[30] Changhua Pei, Yi Zhang, Yongfeng Zhang, Fei Sun, Xiao Lin, Hanxiao Sun, Jian Wu, Peng Jiang, Junfeng Ge, Wenwu Ou, et al. 2019. Personalized re-ranking for recommendation. In *Proceedings of the 13th ACM conference on recommender systems*. 3–11.

[31] Evaggelia Pitoura, Kostas Stefanidis, and Georgia Koutrika. 2021. Fairness in Rankings and Recommendations: An Overview. *The VLDB Journal* 31 (oct 2021), 431–458.

[32] Divyasheel Sharma and Santonu Sarkar. 2022. *Enabling Inference and Training of Deep Learning Models for AI Applications on IoT Edge Devices*. Springer International Publishing, Cham, 267–283.

[33] 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 *CIKM*. 1161–1170.

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPS*, Vol. 30.

[35] Qinyong Wang, Hongzhi Yin, Tong Chen, Zi Huang, Hao Wang, Yanchang Zhao, and Nguyen Quoc Viet Hung. 2020. Next Point-of-Interest Recommendation on Resource-Constrained Mobile Devices. In *WWW*. 906–916.

[36] Yunjia Xi, Weiwen Liu, Jieming Zhu, Xilong Zhao, Xinyi Dai, Ruiming Tang, Weinan Zhang, Rui Zhang, and Yong Yu. 2022. Multi-Level Interaction Reranking with User Behavior History. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[37] Wei Xia, Weiwen Liu, Yifan Liu, and Ruiming Tang. 2022. Balancing Utility and Exposure Fairness for Integrated Ranking with Reinforcement Learning. In *CIKM*. 4590–4594.

[38] Xin Xia, Hongzhi Yin, Junliang Yu, Qinyong Wang, Guandong Xu, and Quoc Viet Hung Nguyen. 2022. On-Device Next-Item Recommendation with Self-Supervised Knowledge Distillation. In *SIGIR*. 546–555.

[39] Xin Xia, Junliang Yu, Qinyong Wang, Chaoqun Yang, Nguyen Quoc Viet Hung, and Hongzhi Yin. 2023. Efficient On-Device Session-Based Recommendation. *ACM Trans. Inf. Syst.* (jan 2023).

[40] Ruobing Xie, Shaoliang Zhang, Rui Wang, Feng Xia, and Leyu Lin. 2021. Hierarchical Reinforcement Learning for Integrated Recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 4521–4528.

[41] Yikai Yan, Chaoyue Niu, Renjie Gu, Fan Wu, Shaojie Tang, Lifeng Hua, Chengfei Lyu, and Guihai Chen. 2022. On-Device Learning for Model Personalization with Large-Scale Cloud-Coordinated Domain Adaption. In *SIGIR*. 2180–2190.

[42] Jiangchao Yao, Feng Wang, Kunyang Jia, Bo Han, Jingren Zhou, and Hongxia Yang. 2021. Device-Cloud Collaborative Learning for Recommendation. In *SIGIR*. 3865–3874.

[43] Chunxing Yin, Bilge Acun, Xing Liu, and Carole-Jean Wu. 2021. TT-Rec: Tensor Train Compression for Deep Learning Recommendation Models. In *Proceedings of the 4th MLSys Conference*.

[44] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep Interest Network for Click-Through Rate Prediction. In *CIKM*. 1059–1068.

A IMPLEMENTATION DETAILS

A.1 History and Candidates Generation

Since the three datasets do not partition the cloud and device history behaviors, we follow [7], with the edge device accessing all the histories and the cloud side missing the three most recently clicked items. In other words, we sort the history items from all sources in chronological order to obtain $[h_1, h_2, \ldots, h_l]$, where h_i, $i = 1, \ldots, l$ denotes the i-th most recent item clicked by the user, and l is the maximum length of the history. We treat the whole sequence $[h_1, h_2, \ldots, h_l]$ as device history and $[h_4, \ldots, h_l]$ as cloud history.

Unlike the Celia Suggestions dataset, the MCC and CTC dataset do not provide ranking lists or candidate sets from different sources, so we have to construct the candidate set of each source on these two datasets. As such, we sample 1 positive (clicked) and 9 negative (non-clicked) items from all sources, and then categorized all the items into different candidate sets based on their sources.

A.2 Reproducibility

The implementation of our proposed DIR is publicly available [4]. We use the last 20 items clicked by the user on each source as history (device history). The maximum length of initial lists of each source is set to 10. The learning rate is selected from $\{5\times10^{-4}, 6\times10^{-4}, 1\times10^{-3}, 2\times10^{-3}, 3\times10^{-3}\}$, and the parameter of L2-Regularization from $\{1\times10^{-5}, 5\times10^{-5}, 1\times10^{-4}, 1\times10^{-3}, 2\times10^{-3}\}$. The embedding size of the categorical feature is set to 16. The batch size and hidden size are set to 128 and 32. The hyper-parameter β that controls the tradeoff between utility and fairness loss is set to 0.5, and target distribution $\mathbf{w}$ in exposure loss is set the same with π in metric $\mathcal{JS}$. For a fair comparison, we also fine-tune all baselines to achieve their best performances.

B HYPER-PARAMETER STUDY

Since we design two losses for the utility and exposure fairness tasks, some hyper-parameters, such as the tradeoff parameter β and the target distribution $\mathbf{w}$ of exposure loss, can substantially impact the final results. As such, we conduct several experiments to get a comprehensive understanding of how the two hyper-parameters affect the performance of DIR.

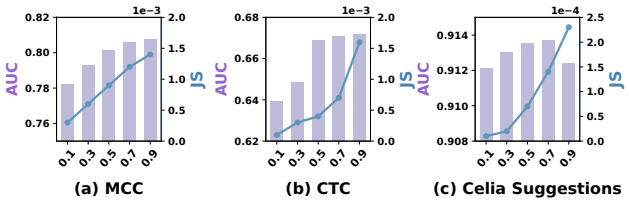

Figure 4: The impact of the tradeoff parameter β.

First, we fix all the other hyper-parameters and tune the tradeoff parameter β. Then, we visualize the variation of a utility-based metric *AUC* and a fairness-based metric $\mathcal{JS}$ on the three datasets in Figure 4. From the figure, we can notice that the tendency of $\mathcal{JS}$ and *AUC* are reversed. When β is smaller, the model performs worse on *AUC*, while the value of $\mathcal{JS}$ is smaller, which means that the model obtains better $\mathcal{JS}$. Conversely, when the value of β is larger, the model yields a better *AUC* and a worse $\mathcal{JS}$. This suggests that there is a tradeoff between utility and exposure fairness controlled by β. Next, we investigate the effect of target distribution $\mathbf{w}$ on DIR's

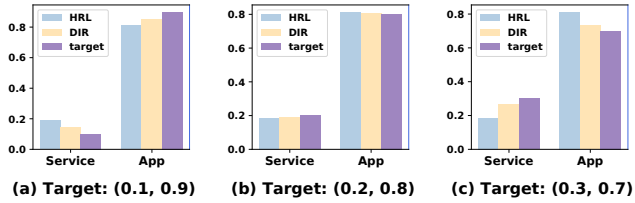

Figure 5: The performance of DIR and HRL under different target distributions. The *AUC* of HRL is 0.8773 and *AUC*s of DIR are (a) 0.9082, (b) 0.9135, and (c) 0.9040.

performance. We select three different target distributions with the ratio of Service versus App being 1: 9, 2: 8, and 3: 7, respectively, of which 2: 8 is the closest one to that in the original Celia Suggestions dataset. As $\mathcal{JS}$ is not very intuitive, we visualize the ratio of exposed Service and App for the results of DIR and HRL, the strongest baseline on $\mathcal{JS}$, in Figure 5. We can observe that the target distribution does not affect HRL, but can drive the distribution of DIR close to the target distribution. Certainly, the deviation of the target distribution from the original one also leads to a decrease in the utility-based metrics of DIR, such as *AUC*, but its result is still better than HRL, indicating that DIR makes a good tradeoff between utility and exposure fairness.

C CASE STUDY

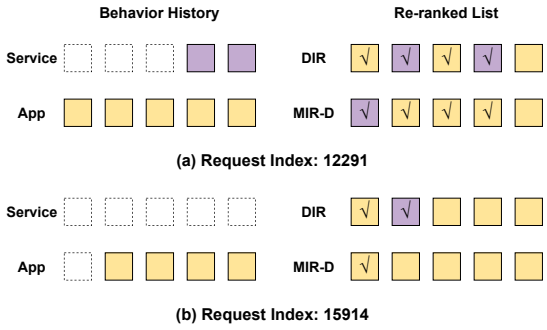

Figure 6: The re-ranked results of DIR and MIR-D. The purple square denotes App and the yellow one denotes Service.

To study how DIR mixes items from various sources, we pick two cases from Celia Suggestions dataset and compare the re-ranked results of DIR with the strongest baseline MIR-D. In Figure 6, the left part presents users' historical behaviors in Service and App sources, while the right part shows the top-5 recommendations generated by DIR and MIR-D. As Celia Suggestions is an imbalanced dataset with few Service occurrences, we adopt the dashed boxes for the absence of historical items. On the right-hand side, we use a check mark to

[4]The MindSpore implementation is available at: https://gitee.com/mindspore/models/tree/master/research/recommend/DIR

indicate that the item was clicked, thus enabling the measurement of the re-ranking performance.

Based on Figure 6, we can observe that DIR is capable of achieving exposure fairness in a personalized way. For instance, when there are many Service items historically clicked by the user in Request 12291, DIR will also recommend more Service items; whereas when faced with Request 15914 where the user has no historical Service items, DIR will be more cautious in recommending Service items. This indicates that DIR can capture the user's personalized preferences and adjust its recommendations dynamically to meet the constraint of exposure fairness. Apart from that, in Request 15914 without the historical behavior in the Service source, DIR successfully recommends the clicked Service item, indicating that DIR is also able to infer interests in Service source from the user's behavior in App source and overcome the problem of cold history.