Controllable Multi-Objective Re-ranking with Policy Hypernetworks

Sirui Chen[*]
School of Information,
Renmin University of China
chensr16@gmail.com

Yuan Wang[*][†]
Alibaba Group
wy175696@taobao.com

Zijing Wen
Alibaba Group
wzj267727@taobao.com

Zhiyu Li
Alibaba Group
tuanyu.lzy@taobao.com

Changshuo Zhang
Gaoling School of AI (GSAI)
Renmin University of China
lyingcs@foxmail.com

Xiao Zhang[†]
Gaoling School of AI (GSAI)
Renmin University of China
zhangx89@ruc.edu.cn

Quan Lin
Alibaba Group
tieyi.lq@taobao.com

Cheng Zhu
Alibaba Group
xize.zc@taobao.com

Jun Xu
Gaoling School of AI (GSAI)
Renmin University of China
junxu@ruc.edu.cn

ABSTRACT

Multi-stage ranking pipelines have become widely used strategies in modern recommender systems, where the final stage aims to return a ranked list of items that balances a number of requirements such as accuracy, diversity etc. Typically, linear scalarization is used to merge these requirements into a single optimization objective by summing them with preference weights. However, existing final-stage ranking methods often rely on static models where preference weights are determined during offline training and remain unchanged during online serving. Adjusting these weights requires retraining the models, which can be time-consuming and resource-intensive. Moreover, the most suitable weights may vary significantly for different user groups or at different time periods, such as during holiday promotions. In this paper, we propose a framework called controllable multi-objective re-ranking (CMR) which incorporates a hypernetwork to generate parameters for a re-ranking model according to different preference weights. In this way, CMR is enabled to adapt the preference weights according to the environment changes in an online manner, without any retraining. Moreover, we classify practical business-oriented tasks into four main categories and seamlessly incorporate them in a new proposed re-ranking model based on an Actor-Evaluator framework, which serves as a reliable real-world testbed for CMR. Offline experiments based on the dataset collected from Taobao App showed that CMR improved several popular re-ranking models by using

them as underlying models. Online A/B tests also demonstrated the effectiveness and trustworthiness of CMR.

CCS CONCEPTS

• **Computing methodologies → Multi-task learning**; **Reinforcement learning**; • **Information systems → Recommender systems**.

KEYWORDS

multi-objective learning, re-ranking, reinforcement learning

1 INTRODUCTION

Multi-stage strategies are prevalent in today's industrial recommender systems, featuring high scalability and low response latency, retrieving relevant items from billion candidates within tens of milliseconds. Early recommender systems [3, 41, 46] estimate the relevance of each item to a user and rank items according to their relevance scores. In recent years, recommender systems have placed more emphasis on other important objectives of the results, such as diversity [36], fairness [27], serendipity [15], and unbiasedness [43, 44] etc. It requires the system to be able to capture the relationship between items and balance multiple objectives, where re-ranking models excel. Therefore, the final-stage re-ranking model [1, 34] is gaining its popularity as a solid supplement of the traditional systems.

Linear scalarization [2] is the most widely used technique to merge multiple optimization objectives into a single one, by weighted summing the objectives. Existing multi-objective re-ranking models conduct the linear scalarization in a static way: the trade-off

---

[*]These authors contributed equally to this work.

[†]First corresponding author: Xiao Zhang. Second corresponding author: Yuan Wang

between tasks, or the set of preference weights, is predefined and fixed during the training and testing phases, resulting in a solution corresponding to a single preference combination. To obtain the performances of different preference weights, in principle, one needs to train the model multiple times, each for a single combination.

In real-world applications, however, dynamic adjustment of the preference weights without model retraining is strongly favored. First, it enables fast hyperparameter tuning. Even with prior knowledge, it is still not trivial to pick the most appreciated preference weights. Tuning methods like grid-search and trail-and-error can be greatly accelerated if repeated model training can be avoided. Second, it enables fast system response to environmental changes. Modern recommender systems such as Amazon need real-time flow control during Black Friday, adjusting their recommendation strategies based on users' behavior. Ideally, the adjustments should take effect immediately. In industrial applications, re-training a model takes hours or even days. Last but not the least, different groups of users may have diverse preferences. As a result, the best set of preference weights varies from group to group. Training multiple models for different user groups is often prohibitively expensive in industrial applications. Dynamic weight adaptation helps to serve diverse groups with a single model.

To achieve online dynamic preference adjustment with a single model, we propose a new controllable multi-objective re-ranking framework (CMR). Inspired by recent studies on multi-task leaning [18, 26, 28], CMR is designed to consist of a hypernetwork and a normal re-ranking model, where the hypernetwork generates special parameters of the re-ranking model for a given set of preference weights. With the generated parameters, the re-ranking model can produce suitable recommended lists for different preference weights. During the offline training, the preference weights are randomly sampled from a feasible distribution to simulate the different environment changes. The parameter generation policy of the hyper network and the re-ranking model are jointly optimized. During the online serving, the preference weights can be specified in real time as desired, without the need for model retraining.

Besides, many business-oriented issues need to be considered in industrial e-commerce recommender applications. For example, to increase sellers' engagement, a recommender system may place more seller-generated contents in results rather than auto-generated contents, even with a certain expense of lowering the relevance. To ensure the functional positioning of a scene, an e-commerce recommender system may prioritize new contents instead of old ones. We conducted a practical review of real-life business-oriented tasks and classify them into four main types: fixed-position insertions, flow control, diversity, and group ordering. Many real-life complex business problems can be broken down into these tasks or their combinations. In traditional recommender systems, these tasks are handled separately in a pipeline fashion, leading to sub-optimal performance. Apart from simplicity and feasibility, it is mainly due to the absence of any joint optimal solution. In this paper, we devise a new re-ranking model based on the Actor-Evaluator (AE) framework which can decently incorporate all four types of business-oriented tasks, offering an end-to-end optimal solution. It also serves as a sound realistic test bed for CMR.

We evaluated CMR with various re-ranking models and demonstrated its applicability across different models. The evaluations were conducted on a large-scale dataset from the Taobao app. Online experiments showcased the flexibility of CMR and the effectiveness of the proposed AE re-ranking model specifically for Taobao.

The main contributions of the paper are:

(1) Problem and framework. We propose the controllable multi-objective re-ranking framework (CMR) to address the online real-time preference adjustment of multi-objective modeling. It consists of a hyper network and a customizable re-ranking model.

(2) A controllable re-ranking model for business-oriented tasks. We classify real business-oriented tasks into four types and propose an AE-based re-ranking model that integrates all these task types. It provides an end-to-end optimal solution, replacing the previous pipeline approach and serving as a realistic test bed for CMR.

(3) Evaluation. Through extensive online and offline experiments, we validate the effectiveness and flexibility of CMR and demonstrate the capabilities of the proposed model on the Taobao App.

2 RELATED WORK

Re-ranking is the final stage of a Multi-stage Recommender System (MRS) that re-ranks top candidates to improve recommendation performance. Re-ranking models explicitly consider list-wise context, distinguishing them from earlier ranking stages. Early approaches used maximum marginal relevance to add items to the list [5], while recent works have adopted deep Neural Networks (NN) [6, 16]. Re-ranking works can be classified based on objectives, such as accuracy [13, 14], diversity [11, 20, 35, 36], and fairness [12, 27, 38, 48]. Re-ranking models have two training paradigms: learning by observed signals and learning by counterfactual signals. Models trained by observed signals require specifying the best list as labels, which can be challenging due to the large list space [1, 49]. In contrast, models trained by counterfactual signals are based on the AE framework and evaluate lists instead of specifying the best list [7, 9, 33]. Multi-task modeling based on the AE framework has not been well explored [21].

Multitask Learning (MTL) seeks to learn a single model to simultaneously solve several learning problems while sharing information among tasks [29, 45]. In recent years, a variety of deep MTL networks with hard or soft parameter sharing structures have been proposed [22, 25, 40]. Another set of works treats MTL as a multi-objective optimization problem that aims to identify Pareto stationary solutions across various tasks [19, 23, 37] and applied in recommender system [10, 17, 47]. These studies employ different approaches: some alternate between optimizing the joint loss and adjusting the weight of each loss [19], while others frame the dynamic weight optimization process as a personalized sequential decision-making problem and propose a solution using the RL paradigm [37]. Notably, these works prioritize the identification of optimal weights that can achieve Pareto efficiency across multiple objectives, rather than solely focusing on generating optimal results for specific preference weights. More recently, several studies [18, 26, 28] propose to learn the entire trade-off curve for MTL problems via hypernets, which has advantages in terms of runtime efficiency at the training phase for multiple preferences and real-time preference control at the inference phase. Inspired by the work of [18, 28], we propose to use hypernets to construct CMR.

3 PRELIMINARY

Given a set of M candidate items $C = \{c_i\}_{1 \le i \le M}$, a user $u \in \mathcal{U}$, and a list reward function $R_{\boldsymbol{w}}(\cdot)$ parameterized by the preference weights $\boldsymbol{w}$, our goal is to find the optimal item list $L_{\boldsymbol{w}}^*$ among all possible lists composed by items from C:

$$L_{\boldsymbol{w}}^* = \arg\max_L R_{\boldsymbol{w}}(L(u, C)),$$

where each list is of length N and it is obvious that $N \le M$. In this paper, we assume that $R_{\boldsymbol{w}}$ takes a linear form:

$$R_{\boldsymbol{w}}(L(u, C)) = \sum_{i=1}^{n_U} w_i U_i(L(u, C)),$$

where U stands for utility for a certain objective and n_U is the number of objectives of concern. Since $\boldsymbol{w}$ indicates the relative importance of utilities, it is called the preference weight. In previous multi-task learning works, $R_{\boldsymbol{w}}$ is fully predefined with a fixed $\boldsymbol{w}$ and a model parameterized by $\boldsymbol{\theta}$ is trained to approximate the mapping from (u, C) to $L_{\boldsymbol{w}}^*$ for the specific $\boldsymbol{w}$. Although the calculation of the utility functions remains predefined, we treat the preference weight $\boldsymbol{w}$ as a variable, which can take any value from a feasible set, and we hope to find the optimal list $L_{\boldsymbol{w}}^*$ for any given $\boldsymbol{w}$. A brute force solution is to train a model for each feasible value of $\boldsymbol{w}$. It can be prohibitively time and resource consuming even for a small number of $\boldsymbol{w}$. One feasible solution is to enable one trained re-ranking model to serve all feasible values of $\boldsymbol{w}$, as will be shown in our CMR framework.

We represent each user u as an embedding vector $\boldsymbol{x}_u \in \mathbb{R}^{d_u}$. Similarly, a candidate item is represented by a feature vector $\boldsymbol{x} \in \mathbb{R}^{d_i}$. A candidate set is represented as a bag of embeddings $\{\boldsymbol{x}_i\}_{i=1}^M$ and a list of N items can be represented as a matrix $X \in \mathbb{R}^{d_i \times N}$, which is a stacking of individual item embeddings $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N]$. A user-item engagement is represented by $\boldsymbol{y} \in \mathbb{R}^{d_e}$ where each element in the engagement vector $\boldsymbol{y}$ could be, for example, exposure, click, purchase, or video viewing time, etc., depending on the specific applications. The user-list engagement is represented by the matrix $Y = [\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_N] \in \mathbb{R}^{d_e \times N}$. Note that we do not have the engagement of all M candidate items, since only N of them are in the final recommended list and a user does not have access to the rest. An offline log sample is a tuple of (u, C, X, Y) and a log data set $\mathcal{D}$ is $\{(u_i, C_i, X_i, Y_i)\}_{i=1}^{n_D}$ where n_D is the size of the data set. A utility function is a mapping from (u, X) to a scalar, $U : \mathbb{R}^{d_u} \times \mathbb{R}^{d_i \times N} \to \mathbb{R}$. Scalars d_i, d_e, and d_u denote the respective embedding dimensions.

4 OUR PROPOSAL

In the following sections, we will introduce the overall framework of CMR, our re-ranking model design, the business-oriented tasks, and the model training algorithm, respectively.

4.1 The CMR Framework

Fig. 1 shows the basic structure of the CMR framework, which consists of a hypernetwork and an arbitrary multi-objective re-ranking model. The hypernetwork $h(\boldsymbol{w}; \boldsymbol{\phi})$ takes in the preference weights $\boldsymbol{w}$ as inputs and generates parameters for the re-ranking model, which helps the re-ranking to generate recommendation lists according
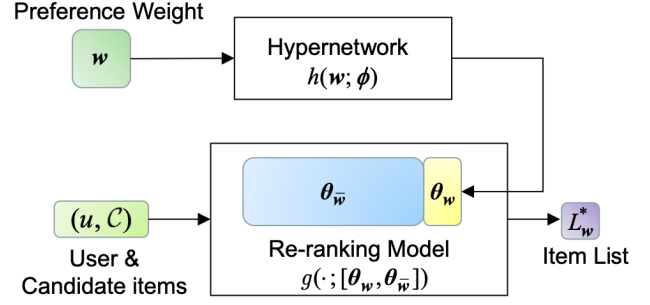

Figure 1: The proposed CMR framework.

to the specific preference weights $\boldsymbol{w}$. A re-ranking model based on modern deep neural networks can be as large as 50GB in size with tens of billions of parameters. It is infeasible for the hypernetwork to generate all of them. To circumvent this issue, one can split the parameter of the re-ranking model into $\boldsymbol{w}$-sensitive ones $\boldsymbol{\theta}_{\boldsymbol{w}}$ and $\boldsymbol{w}$-insensitive ones $\boldsymbol{\theta}_{\bar{\boldsymbol{w}}}$. The majority of $\boldsymbol{\theta}$ belongs to $\boldsymbol{\theta}_{\bar{\boldsymbol{w}}}$ which may include item embedding layers and representation learning layers. In contrast, $\boldsymbol{\theta}_{\boldsymbol{w}}$ can be a very compact part of $\boldsymbol{\theta}$, such as the last a few layers of a re-ranking model. Moreover, significant invasive structure modifications of a model usually require retraining of the entire model, which can be prohibitively expensive in industrial applications. By splitting $\boldsymbol{\theta}$, one can only modify the part involving $\boldsymbol{\theta}_{\boldsymbol{w}}$ and keep the $\boldsymbol{\theta}_{\bar{\boldsymbol{w}}}$ part intact, which helps to accommodate existing deployed models into the CMR framework.

Inspired by Conditional GAN [24], we use conditional training in the CMR framework as shown in Algorithm 1. For each training sample or batch, a preference weight $\boldsymbol{w}$ is sampled from its predefined feasible distribution $\mathcal{P}_{\boldsymbol{w}}$. Then it is fed into the hypernetwork to generate $\boldsymbol{\theta}_{\boldsymbol{w}}$ and the re-ranking model is invoked to generate a list L with parameters $[\boldsymbol{\theta}_{\boldsymbol{w}}, \boldsymbol{\theta}_{\bar{\boldsymbol{w}}}]$. The reward function $R_{\boldsymbol{w}}$ evaluates L with the sampled $\boldsymbol{w}$. Finally, the parameters of the hypernetwork $\boldsymbol{\phi}$ are updated according to the value of $R_{\boldsymbol{w}}$. Optionally $\boldsymbol{\theta}_{\bar{\boldsymbol{w}}}$ can be updated jointly if the re-ranking model is not fully trained. Various optimization methods can be used to update $\boldsymbol{\phi}$ and $\boldsymbol{\theta}_{\bar{\boldsymbol{w}}}$, from simple gradient descent to complex reinforcement learning algorithms. In principle, lists with high rewards should be generated with high probability. The feasible distribution of the preference weight $\mathcal{P}_{\boldsymbol{w}}$ can be derived from domain knowledge, business considerations, or practice experience. For example, relevance is typically considered more important than diversity, resulting in a higher weight for relevance utility than diversity utility. To test the generalization ability of CMR, we attempt to minimize our reliance on prior knowledge in the experiments and simply sample the preference weight of each utility function from a uniform distribution $U(0, w_i^{\max})$ independently, where $w_i^{\max}$ is the maximum value for a given preference weight, ranging from 0 to 1. The hypernetwork is deployed along with the re-ranking model and $\boldsymbol{w}$ is specified in real-time for each case during serving. The CMR framework enables a re-ranking model to generate the optimal recommendation list for any given $\boldsymbol{w}$. Determining optimal $\boldsymbol{w}$ at serving time is beyond the scope of this paper.

Algorithm 1 Conditional training in the CMR framework

Input: hypernetwork $h(\cdot; \boldsymbol{\phi})$ parameterized by $\boldsymbol{\phi}$; re-ranking model $g(\cdot; [\boldsymbol{\theta_w}, \boldsymbol{\theta_{\tilde{w}}}])$ parameterized by the $\boldsymbol{w}$-sensitive and insensitive parameters; a list reward function $R_{\boldsymbol{w}}$; training dataset $\mathcal{D} = \{(u, C, X, Y)_i\}_{i=1}^{n_D}$; a feasible distribution of the preference weight $\mathcal{P}_{\boldsymbol{w}}$

Output: a learned hypernetwork $h(\cdot; \boldsymbol{\phi}^*)$ and a re-ranking model g with insensitive parameters $\boldsymbol{\theta}_{\tilde{\boldsymbol{w}}}^*$

1: **repeat**
2:     sample a training instance $(u, C, X, Y) \sim \mathcal{D}$
3:     sample a preference weight $\boldsymbol{w} \sim \mathcal{P}_{\boldsymbol{w}}$
4:     $\boldsymbol{\theta_w} \leftarrow h(\boldsymbol{w}; \boldsymbol{\phi})$ {run the hypernetwork}
5:     $L \leftarrow g((u, C); [\boldsymbol{\theta_w}, \boldsymbol{\theta_{\tilde{w}}}])$ {run the re-ranking model}
6:     evaluate the list by $R_{\boldsymbol{w}}(L)$
7:     update $\boldsymbol{\phi}$ to modify the generating probability of L according to $R_{\boldsymbol{w}}(L)$ with proper algorithms
8:     optionally, update $\boldsymbol{\theta_{\tilde{w}}}$ to modify the generating probability of L according to $R_{\boldsymbol{w}}(L)$ with proper algorithms if the re-ranking model has not been well trained
9: **until** Converge
10: **return** $\boldsymbol{\phi}^*$ and $\boldsymbol{\theta}_{\tilde{\boldsymbol{w}}}^*$

4.2 The Proposed Re-ranking Model

Our re-ranking model is based on the AE framework which can incorporate various utilities easily. It contains two main modules: an actor for list generation and an evaluator for list evaluation.

4.2.1 The Actor Module. The Actor has an encoder-decoder structure as illustrated in Fig.2(a). Before feeding into the encoder, feature augmentation and embedding lookup are carried out. The augmented features of an item come from the candidate set, such as the ranking of the item in terms of historical Click Through Rate (CTR) within the set. It is a very efficient and simple way of informing a model of the relationship between an individual item and its containing candidate set. Then ID features are transferred to real-valued embeddings to facilitate numerical computation.

The DeepSet-based Encoder. DeepSet [42] is selected as the encoder because it is insensitive to the order of the input items, as shown in Fig.2(c). It is preferable because, in complex applications, it is not trivial to find a good initial list mixing texts, images, and videos. Poor initial ordering may hurt the performance of a re-ranking model. It is neither necessary to provide an initial list since (u, C) contains all information a re-ranking model needs to know to generate a good list. The context embedding $\boldsymbol{e}_c$ is calculated as

$$\boldsymbol{e}_c = MLP_2\left(\sum_{i=1}^{M} MLP_1([\boldsymbol{x}_i; \boldsymbol{x}_u])\right),$$

where $\boldsymbol{x}_u$ means user vector and $[\boldsymbol{x}_i; \boldsymbol{x}_u]$ means the concatenation of the two vectors. *MLP* stands for multilayer perceptron. Note that the output of MLP_1 is used as the item embedding $\boldsymbol{e}$ for each item.

The PointerNet-based Decoder. The decoder is based on PointerNet [32] which picks one item from the candidate set at one time, updates context information immediately, and then picks the next, as shown in Fig.2(d). Item selection is based on a local context enhanced attention mechanism as shown in Fig.2(b).

At the very beginning of decoding, the context embedding $\boldsymbol{e}_c$ is used as the initial hidden state of a Recurrent Neural Network (RNN) cell and a special token "start" is fed into the RNN cell as the input. After that, at each step, the output embedding of the RNN cell acts as the state embedding $\boldsymbol{e}_s$, which is supposed to contain all necessary information for item selection. The local context enhanced attention $\boldsymbol{a} \in \mathbb{R}^M$ is calculated as

$$\boldsymbol{a} = \text{softmax}([MLP_3([\boldsymbol{e}_s; \boldsymbol{e}_i; \boldsymbol{e}_{si}])]_{i=1}^M), \tag{1}$$

where $\boldsymbol{e}_i$ is the item embedding and $\boldsymbol{e}_{si}$ is the local context embedding. It is termed "local" because the context is specific to the current state and the current candidate item. As illustrated in Fig. 2(b), the introduction of the local context is to facilitate the model's understanding of business rules and prior knowledge of a good list. For example, one may want an item list from diverse sellers. It can be achieved by controlling seller duplication in the result list. If a candidate item introduces a seller duplication, its attention value should be smaller so that it has a lower chance to be selected. Although the piece of information regarding seller duplication can potentially be abstracted implicitly by a complex neural network, it is not necessary and is low efficient. Instead, we can directly add a feature "the number of seller duplication of the current item in the preceding list" into the local context embedding $\boldsymbol{e}_{si}$.

Masking is applied to the attention values for two main reasons. First, an item that has been selected in the preceding steps should not be selected again. So their attention values are masked to zero. Second, business rules may ask a model to place a certain item at a certain position in the result list. In this case, the attention values of all items other than the target one are masked to zero. Incorporating a sampling mechanism is crucial for an actor because it lets the actor try different actions and ultimately find the optimized list-generating policy. We use Thompson Sampling, which samples an item proportionally to its masked attention value.

4.2.2 The Evaluator Module. The evaluator contains an ensemble of utility functions. Recall that a utility function is a mapping from (u, X) to a scalar, $U : \mathbb{R}^{d_u} \times \mathbb{R}^{d_i \times N} \rightarrow \mathbb{R}$, indicating the quality of the list in one perspective. Some of the utility functions come from business-oriented requirements and prior knowledge of good lists, such as item diversity. These utilities have analytical formulas and can be calculated directly. Other utilities, such as the probability of user-list engagement, have to be predicted by models, which are trained with offline data. Once trained, they are used as predefined utility functions. This section introduces the model used for the predictions.

The overall structure of the evaluator model is shown in Fig.3(a). After the same feature processing as that of in the actor, the item list is represented by a matrix $E = [\boldsymbol{e}_1, \boldsymbol{e}_2, ..., \boldsymbol{e}_N]$. It is processed by 5 channels that focus on different aspects of the list. The results are then concatenated and transferred to the final predictions through a *MLP*. The first channel is "sum pooling", which is an element-wise sum of the item embeddings as shown in Fig.3(b), resulting in $\boldsymbol{e}_{sp} := \sum_{i=1}^N \boldsymbol{e}_i$, The second channel is "forward & concat" which is good at abstracting useful information from each item as in Fig.3(c) and results in an embedding $\boldsymbol{e}_{fc}$ as

$$\boldsymbol{e}_{fc} = [MLP_4(\boldsymbol{e}_1); MLP_4(\boldsymbol{e}_2); \cdots ; MLP_4(\boldsymbol{e}_N)]. \tag{2}$$

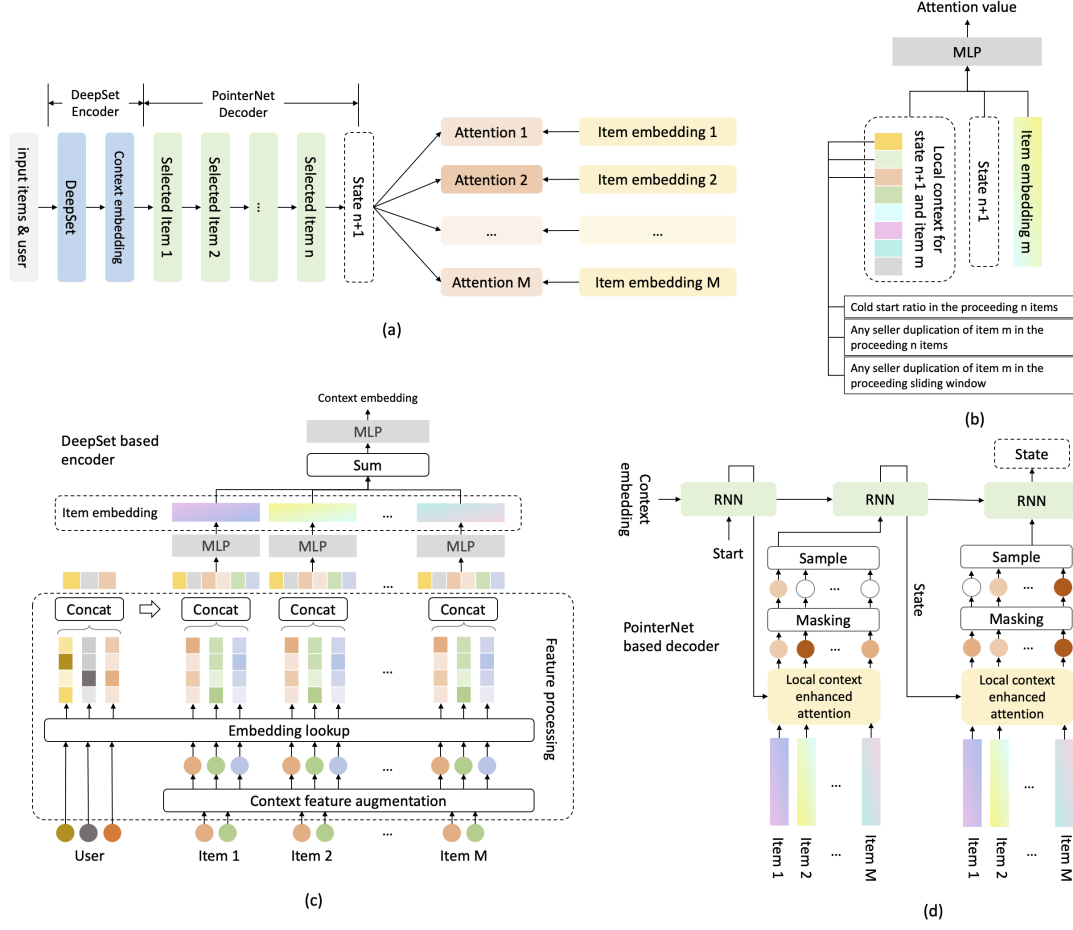The third channel is "multi-head self-attention" as in Fig.3(d) which

Figure 2: Model structure of the Actor. (a) the encoder-decoder structure; (b) local context enhanced attention for item selection; (c) the DeepSet-based encoder; (d) the PointerNet-based decoder.

aims to capture mutual influence between items. It outputs $\boldsymbol{e}_{mh}$ as

$$\boldsymbol{e}_{mh} = reduce\_sum([head_h(W_h^q E, W_h^k E, W_h^v E)]_{h=1}^H),$$

where W^q, W^k, and W^v are linear projection matrices and H is the number of heads. *head* stands for scaled dot-product attention. The *reduce_sum* of a matrix $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]$ means $reduce\_sum(X) = \sum_{i=1}^n \boldsymbol{x}_i$. The fourth channel is "RNN" as in Fig.3(e) which reveals the evolution trend of the list. The result embedding $\boldsymbol{e}_{rnn}$ is the output of the RNN cell at the final step. The fifth channel is "pair-wise comparison" as in Fig.3(f). As suggested by its name, it compares each item pair using inner product and the result embedding $\boldsymbol{e}_{pc} = [\langle \boldsymbol{e}_i, \boldsymbol{e}_j \rangle]_{1 \leq i \leq N, i < j \leq N}$. The final *Predictions* are obtained via a *MLP*

$$Predictions = MLP_5([\boldsymbol{e}_{sp}; \boldsymbol{e}_{fc}; \boldsymbol{e}_{mh}; \boldsymbol{e}_{rnn}; \boldsymbol{e}_{pc}]).$$

4.3 Business-Oriented Utilities

We classify business-oriented tasks into four main categories: fixed-position insertions, flow control, diversity, and group ordering. Many real-life complex business problems can be broken down into these tasks or their combinations. Traditionally, these tasks are

handled separately in a pipeline manner, leading to sub-optimal performance. This section incorporates all the tasks into our proposed re-ranking model which provides an end-to-end optimal solution.

4.3.1 Flow Control. Flow control ensures that the contents from certain groups get enough exposure to customers. It comes from several perspectives: from the fairness perspective where the groups are protected minorities, and from a platform ecology perspective where contents by new providers are distributed with priority.

The effectiveness of flow control is measured by exposure ratio. We may want the exposure ratio of a certain group to be greater than a predefined threshold. The flow control utility is defined as

$$U_g^f = -\frac{1}{|b|} \sum_{p \in b} \mathbb{I}\left(\frac{\sum_{i \in p} \mathbb{I}\left(i \in \mathcal{I}_g\right)}{|p|} \leq t_g^e\right), \tag{3}$$

where the superscript f means flow control, p means a page, b means a training batch, and t^e means the exposure ratio threshold. With a little abuse of notations, i stands for the item at position i as well as the position index of an item in the list, as long as it does not cause confusion. $\mathcal{I}_g$ is a set containing all items belonging to the group g. $|p|$ stands for the number of slots in the page. $\mathbb{I}$ is the
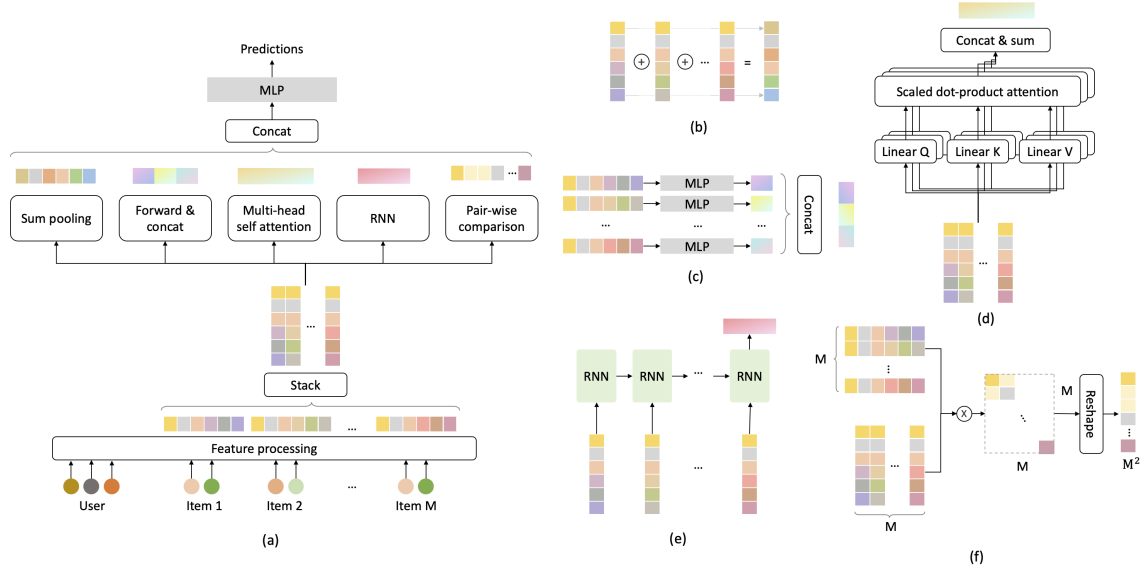
Figure 3: Model structure of the Evaluator. (a) the overall structure; (b) the sum pooling channel; (c) the forward & concat channel; (d) the multi-head self-attention channel; (e) the RNN channel; (f) the pair-wise comparison channel.

indicator function. Please note the negative sign at the beginning of the right-hand side of the equation. It means recommendation lists violating the requirement of flow control will receive a penalty.

Eq. (3) is too strict since it requires all pages to satisfy the threshold. We only want to control the exposure ratio at the population level, rather than on each page. Although the exposure ratio of the whole population is not accessible during training, the exposure ratio of a batch can work as a handy approximation:

$$U_g^f = -\frac{1}{|b|} \mathbb{I}\left(\text{RatioInBatch}(g) \leq t_g^e \right) \sum_{p \in b} \mathbb{I}\left(\frac{\sum_{i \in p} \mathbb{I}(i \in \mathcal{I}_g)}{|p|} \leq t_g^e \right).$$

(4)

where $\text{RatioInBatch}(g) = \frac{\sum_{p \in b} \sum_{i \in p} \mathbb{I}(i \in \mathcal{I}_g)}{|b||p|}$ is a gating. As long as the exposure ratio of the whole population meets the requirement, the exposure ratio of individual pages is no longer a concern.

In the scenarios where customers perceive items in a result list one by one, customers may leave at any time, and items at the bottom of a list may not be actually seen by the customers, which makes the above exposure ratio calculation inaccurate. To handle it, we can add another position term

$$U_g^f = -\frac{1}{|b|} \mathbb{I}\left(\text{RatioInBatch}(g) \leq t_g^e \right) \sum_{p \in b} \mathbb{I}\left(\frac{\sum_{i \in p} \mathbb{I}(i \in \mathcal{I}_g)}{|p|} \leq t_g^e \right) -$$

$$\frac{1}{|b|} \mathbb{I}\left(\text{PosInBatch}(g) \geq t_g^p \right) \sum_{p \in b} \mathbb{I}\left(\frac{\sum_{i \in p} \text{pos}_i \mathbb{I}(i \in \mathcal{I}_g)}{\sum_{i \in p} \mathbb{I}(i \in \mathcal{I}_g)} \geq t_g^p \right),$$

(5)

where $\text{PosInBatch}(g) = \frac{\sum_{p \in b} \sum_{i \in p} \text{pos}_i \mathbb{I}(i \in \mathcal{I}_g)}{\sum_{p \in b} \sum_{i \in p} \mathbb{I}(i \in \mathcal{I}_g)}$, and pos_i means the exposure position of item i and t^p is the exposure position threshold. Note that Eq. (3), Eq. (4), and Eq. (5) show possible choices of flow control utility for a particular group g. Flow control for more than

one group may co-exist in one model.

4.3.2 Diversity. Diversity means items in result lists should come from different groups. The groups may refer to sellers, categories, supply sources, as well as item exhibition types such as text, image, and video. The diversity utility of a group g can be written as

$$U_g^d = \frac{1}{|b||p|} \sum_{p \in b} \sum_{i \in p} \mathbb{I}(g_i \notin \mathcal{G}_{i-1}),$$

where the superscript d means diversity and g_i means the group of the item at position i. $\mathcal{G}_{i-1}$ is the group set of items in a sliding window before item i: $\mathcal{G}_{i-1} = \{g_j | j \in \{i - n_g^d, i - (n_g^d - 1), \ldots, i - 1\}, j > 0\}$ where n_g^d is the length of the diversity window. Setting $n_g^d = \infty$ promotes whole list-wise diversity.

4.3.3 Group Ordering. Group ordering means the items from certain groups are ranked before others with high probability. For example, in a new arrival promoting scene, it is preferred to show new items released within 3 days first and then others. Group ordering can be achieved by checking the group priority of item pairs

$$U_g^o = \frac{1}{|b|} \sum_{p \in b} \frac{1}{2|p||p-1|} \sum_{i \in p} \sum_{j > i} \mathbb{I}(\text{priority}(g_i) \geq \text{priority}(g_j)),$$

where the superscript o means group ordering and priority indicates the predefined priority function of groups.

4.3.4 Fixed Position Insertion. Fixed position insertion is a very strict task, which means a given item must be placed at a specified position of a result list with 100% probability. A customer may come to a recommendation scene by clicking a trigger item, it is mandatory to place the trigger item at the very top of a recommendation list. It may seem trivial at first glimpse since one can always add the trigger item to the top of a list after the list has been created. However, such a method may fail to handle the relations between

multiple objectives simultaneously. For example, adding a trigger item on top of a separately generated list may result in seller duplication of the first two items, which breaks the diversity requirement. Adding a utility function does not work well for this task because a utility cannot offer a strong probability guarantee. Instead, we use masking to achieve fixed position insertion as introduced in 4.2.1.

4.4 Model Training

The evaluator is fully trained before the actor in a purely supervised learning fashion. So far we only train classification models for the evaluator with the classic Cross Entropy: $\mathcal{L}_{eval} = -\sum_{i=1} y_i \log(p_i)$, where $y_i \in \{0, 1\}$ is the one-hot label for a class and p_i is the model prediction for the class. The models may predict whether the generated list will lead to a certain kind of user engagement, e.g. click, and the intensity of the engagement.

We use a REINFORCE-based method for actor training. In reinforcement learning, a policy is a mapping from a state to a probability distribution over possible actions which maximizes the total reward. In simple words, policy tells us which action to take in a given state. In our case, Eq. (1) plays the essential role of a policy, where the state is $[e_s; e_i; e_{si}]_{i=1}^M$ and the probability distribution over M possible items is a. Suppose for a user u and a candidate item set $C = \{c_1, c_2, ..., c_M\}$, the actor generate a list $L(u, C)$ as $[c_{\pi_1}, c_{\pi_2}, ..., c_{\pi_N}]$. π is the indicator of a list, where $\pi_n = m$ means the mth item in the candidate set is placed in the nth position in the output list. Here we assume an arbitrary ranking in the candidate set to get the index. The loss function for the actor $\mathcal{L}_{actor}$ is

$$\mathcal{L}_{actor} = -\left[R_{\boldsymbol{w}}(L(u, C)) - R_{\boldsymbol{w}}(L_{exp})\right] \sum_{n=1}^N \log(a_{\pi_n}),$$

where $L(u, C)$ is the list generated by the actor, L_{exp} is the recorded exposure list, the reward difference between the two $R_{\boldsymbol{w}}(L(u, C)) - R_{\boldsymbol{w}}(L_{exp})$ severs as the advantage function. In this loss, we assume each action contributes equally to the advantage value. We recognize advanced training options such as Proximal Policy Optimization (PPO) [30]. Nevertheless, the REINFORCE-based method works well so we leave the choice of the options to further studies. We observe that gradient clipping greatly benefits the training.

5 EXPERIMENTS

We conducted offline and online experiments to verify the effectiveness of the proposed CMR framework. The source code of offline experiments has been share at `https://github.com/lyingCS/Controllable-Multi-Objective-Reranking`.

5.1 Offline Experiments

We trained CMR on the benchmark LibRerank[1] with the public recommendation dataset Ad2[2]. We conducted experiments to answer the following two questions: i) Can CMR take effect on various re-ranking models? ii) How does our method perform compared with other rule-based controllable re-ranking baselines?

[1] https://github.com/LibRerank-Community/LibRerank
[2] https://tianchi.aliyun.com/dataset/56

5.1.1 Experiment Setting. We conducted our offline experiments on the public benchmark LibRerank[1], which can automatically perform re-ranking experiments and integrate a major collection of re-ranking algorithms.

To answer the first question, we extend the single accuracy objective in the original benchmark to multi-objective by adding diversity objective. Specifically, inspired by MDP-DIV[36], we define the diversity reward function as the promotion of diversity metric caused by choosing the i-th item:

$$R_i^{div}(L(u, C)) = ERR\_IA[L_i(u, C)] - ERR\_IA[L_{i-1}(u, C)].$$

The loss function is designed to be $\mathcal{L}_{actor} := \lambda \mathcal{L}_{actor}^{acc} + (1 - \lambda)\mathcal{L}_{actor}^{div}$, where $\lambda \in [0, 1]$ is the trade-off parameter and $\mathcal{L}_{actor}^{acc}$, $\mathcal{L}_{actor}^{div}$ stand for losses in terms of utilities of accuracy and diversity.

Dataset: The original Ad dataset records 1 million users and 26 million ad display/click logs, with 8 user profiles (e.g., id, age, and occupation), 6 item features (e.g., id, campaign, and brand). LibRerank transformed the records of each user into ranking lists according to the timestamp of the user browsing the advertisement. Items that have been interacted with within 5 minutes are sliced into a list. The final Ad dataset contains 349,404 items and 483,049 lists.
Baselines: We chose several representative methods as baselines: **Seq2Slate** [1]: a sequence-to-sequence model that formulates the re-ranking as a sequence generation problem, and sequentially selects the next items by pointer network; **EG-Rerank** [9]: a re-rank model that adopts an evaluator generator paradigm—with a generator to generate feasible permutations and an evaluator to evaluate the listwise utility of each permutation; **MMR** [5]: a heuristic approach with the documents selected sequentially according to maximal marginal relevance; **APDR** [31]: a learning model that aims to improve the diversity and personalization of image search results.

Evaluation metrics: We selected MAP, NDCG as the accuracy metrics, and ILAD[8], ERR_IA[39] as the diversity metrics. Specifically, we chose MAP@5, MAP@10, NDCG@5, NDCG@10, ILAD@5, ERR_IA@5 and ERR_IA@10. Note that MAP and NDCG are greedy metrics, which means the highest evaluation is reached when items are ranked strictly according to their "relevance". However, such lists may not lead to the best online performance. Instead, the AE framework-based re-ranking models aim to maximize list-wise user engagement, without putting too much emphasis on in-page item ordering. The user engagement is predicted by the evaluator. As a result, the best online performing AE re-ranking models may seem sub-optimal in terms of MAP and NDCG. We stick with the metrics because they are popular and can be easily applied to various methods, whether based on the AE framework or not.

5.1.2 Performance Analysis. To answer the first question, we investigate the following models in the CMR framework: Seq2Slate, EG-Rerank, and ours. The results are reported in Table 1. The first row of the table shows the metrics of the list after sorting by ranker. In this experiment, the ranker we used is lambdaMART[4]. Then we tested the two baselines and our model at three different accuracy preference(0, 0.5, 1), and counted their 4 accuracy metrics and 3 diversity metrics. Note that any permutation makes no difference to the calculation of ILAD@10, so we do not present this metric in the table. We notice that as the accuracy preference increases, the accuracy metrics show an upward trend, and the diversity metrics

Table 1: Offline evaluation results of sequential re-ranking models. The experiments are repeated 3 times with different random seeds. We display the mean performance and standard deviation. The best results of diversity and accuracy metrics under the condition that λ(acc_prefer) is 0 and 1 are highlighted in bold respectively and the second-best results are underlined.

Method	λ	map@5	map@10	ndcg@5	ndcg@10	ilad@5	err_ia@5	err_ia@10
Initial	–	0.59961	0.60321	0.68092	0.69531	0.64535	1.29619	1.32672
Seq2Slate	0	0.5924±0.0011	0.5963±0.0011	0.6750±0.0010	0.6901±0.0008	<u>0.6449±0.0005</u>	<u>1.2957±0.0003</u>	<u>1.3264±0.0002</u>
	0.5	0.6001±0.0012	0.6038±0.0013	0.6815±0.0010	0.6958±0.0010	0.6448±0.0002	1.2955±0.0001	1.3262±0.0001
	1	**0.6021±0.0005**	**0.6057±0.0005**	**0.6829±0.0003**	**0.6972±0.0004**	0.6444±0.0002	1.2954±0.0002	1.3261±0.0001
EG-Rerank	0	0.5988±0.0007	0.6026±0.0007	0.6807±0.0006	0.6948±0.0005	0.6444±0.0005	1.2952±0.0005	1.3259±0.0004
	0.5	0.6000±0.0005	0.6038±0.0004	0.6815±0.0003	0.6957±0.0003	0.6434±0.0002	1.2944±0.0001	1.3253±0.0000
	1	0.6009±0.0012	0.6046±0.0011	0.6822±0.0009	0.6963±0.0009	0.6424±0.0007	1.2935±0.0005	1.3247±0.0003
CMR (ours)	0	0.5949±0.0046	0.5987±0.0045	0.6772±0.0036	0.6919±0.0034	**0.6485±0.0004**	**1.2991±0.0003**	**1.3289±0.0004**
	0.5	0.5971±0.0035	0.6009±0.0034	0.6789±0.0028	0.6935±0.0026	0.6484±0.0007	1.2987±0.0001	1.3286±0.0002
	1	<u>0.6016±0.0018</u>	<u>0.6053±0.0017</u>	<u>0.6825±0.0014</u>	<u>0.6969±0.0013</u>	0.6480±0.0008	1.2983±0.0003	1.3282±0.0001

Table 2: Offline comparisons of CMR and rule-based baselines. No standard deviation is reported since random seeds do not affect APDR and MMR experimental results. The best results of diversity and accuracy metrics under the condition that λ(acc_prefer) is 0 and 1 are highlighted in bold respectively and the second-best results are underlined.

| Method	λ	map@5	map@10	ndcg@5	ndcg@10	ilad@5	err_ia@5	err_ia@10
Initial	–	0.5996	0.6032	0.6809	0.6953	0.6454	1.2962	1.3267
APDR	0	0.5965	0.6001	0.6787	0.6932	**0.6673**	**1.3153**	**1.3409**
	0.5	0.5967	0.6005	0.6788	0.6933	0.6617	1.3101	1.3373
	1	0.5971	0.6010	0.6789	0.6936	0.6453	1.2962	1.3268
MMR	0	0.5992	0.6027	0.6808	0.6951	<u>0.6656</u>	<u>1.3147</u>	<u>1.3402</u>
	0.5	0.5996	0.6032	0.6810	0.6953	0.6565	1.3054	1.3335
	1	<u>0.5996</u>	<u>0.6032</u>	<u>0.6809</u>	<u>0.6953</u>	0.6454	1.2962	1.3267
CMR (ours)	0	0.5949	0.5987	0.6772	0.6919	0.6485	1.2991	1.3289
	0.5	0.5971	0.6009	0.6789	0.6935	0.6484	1.2987	1.3286
	1	**0.6016**	**0.6053**	**0.6825**	**0.6969**	0.6480	1.2983	1.3282
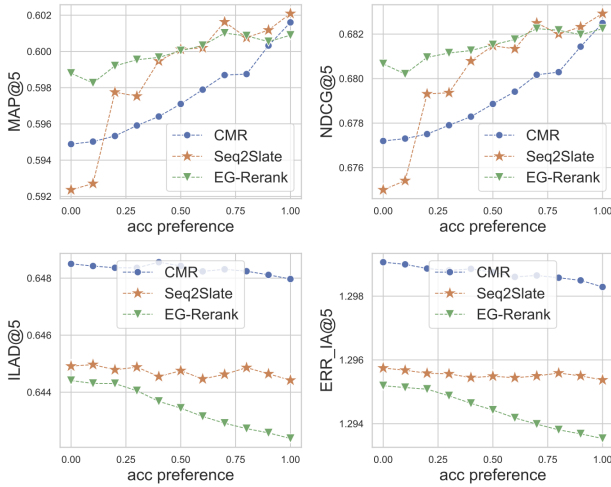
Figure 4: Controllable effects of sequential re-ranking models.

are basically the opposite. We then conducted controllable experiments of the above three re-ranking models on the test set. The final results are shown in Fig. 4, where the horizontal axis is the preference for accuracy and the vertical axis is the corresponding metric.

The upper two subgraphs are the accuracy metrics MAP@5 and NDCG@5, and the lower ones are the diversity metrics ILAD@5 and ERR_IA@5. As we can see, there is a clear trend that as the accuracy preference increases, the corresponding accuracy metrics increase and the corresponding diversity metrics decreases, with the most obvious and smoothest change in CMR. It can also be seen that for the accuracy metrics, the effect of Seq2slate is slightly better than that of the other two re-ranking models, followed by CMR. But for diversity metrics, CMR is much better. So our CMR framework can be well adapted to various re-ranking models.

For the second question, We compared our model with the existing rule-based controllable methods APDR and MMR. The experiment results are performed in Table 2. We observe that though under diversity metrics CMR underperforms APDR and MMR, it can break through the limitations of the rule-based methods and outperforms these baselines in terms of accuracy.

5.2 Online Experiments

We conducted online experiments on the "Subscribe" scene in the Taobao App, a leading e-commerce platform in China. Its main entrance is the "Subscribe" button on top of Taobao's main landing page, and it is a stream of various elements including item lists, posters, coupons, etc.

In the first experiment, we evaluated whether an online performance metric changes well with a corresponding preference weight.

We train a CMR re-ranking model from scratch and assign random preference weight during serving. We picked one representative from each of the four types of business-oriented tasks: a flow control utility to ensure the exposure ratio of cold start contents as in Eq. (5), a seller account diversity utility to promote the number of sellers in result lists, a group ordering utility to place more new contents in the front of result lists, and a traditional click utility to promote user engagement. Trigger contents are guaranteed to sit at the top of result lists with fixed position insertion masking. The result is shown in Fig. 5, where the horizontal axis is the weight of a utility and the vertical axis is the relative improvement of a corresponding metric online. Logged samples are grouped into successive bins according to preference weights, and the average metric within each bin is calculated. The metrics are normalized so that the minimum value is 1 and the relative improvement is calculated with respect to the minimal value. As we can see, there is a clear trend that as a preference weight goes up, the corresponding online metric increases. Although the range of the preference weights is the same from 0 to 1, the adaptive ranges of the metrics vary from 1.4% to 7%. It is likely an essential reflection of the nature of the recommender system. Note that currently the CMR model takes 20 inputs and outputs a list of 10, which limits the adaptive ranges of the metrics. We are working on increasing the input size. The blue metric curves are not ideally monotonical. We mainly attribute it to data sparsity. As online experiments may hurt real users' experience, we only use a small portion of the App traffic.
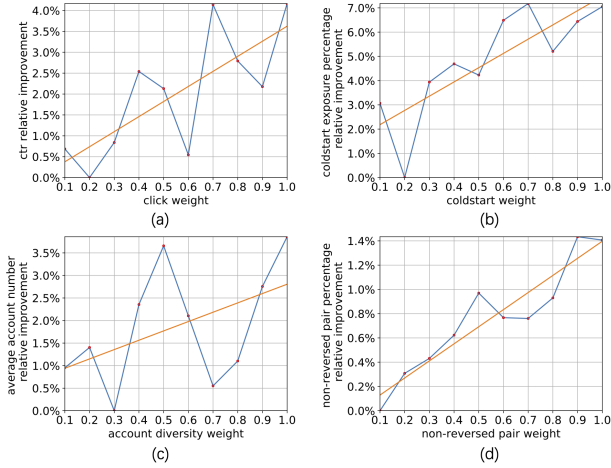

Figure 5: Online metric changes w.r.t. preference weights. The orange lines are the linear fitting of the blue metric curves.

In the second experiment, we aim to show the superiority of the joint optimal solution provided by the proposed re-ranking model. The baseline is a classic module pipeline solution. In the first module, the click probability of each content is predicted as the basic ranking score to form the initial recommend list. In the second module, a freshness bias is added to the ranking score so that new contents are placed relatively in front of result lists. In the third module, a heuristic diversity algorithm is applied to promote seller diversity[31]. In the fourth module, cold start contents are inserted

into the list. In the final module, trigger contents are placed at the top of the list. This is the default solution of the online system and the involved hyper-parameters are kept tuned manually for years. Our CMR re-ranking model gets rid of the pipeline and offers an end-to-end one-step solution. One CMR model is trained and the preference weights are tuned manually online. The model config, such as the utilities involved, is the same as in the first online experiment. As no other re-ranking model considers such a big scope, we show the A/B test results of the base solution and the CMR solution only. Table 3 shows the results of a 7-day online A/B test. The first four metrics are directly related to the utility functions in the CMR model and all of them are improved, which are content click number per user, seller exposure number per user, cold start exposure ratio, and chronological ordering. An interesting observation is that stay time per user and content exposure number per user are also improved, although we do not model them directly. In our experience, it is a sweet byproduct of AE based re-ranking models.

Table 3: Online A/B test results.

Metric	Relative improvement
content click number per user	0.62%
seller exposure number per user	2.43%
cold start exposure ratio	4.27%
chronological ordering	1.40%
stay time per user	1.41%
content exposure number per user	0.73%

6 CONCLUSION

This paper presents a controllable multi-objective re-ranking (CMR) framework to adapt the recommendation re-ranking models according to the preference weights in a dynamic manner, avoiding the costs of re-training the models. The key of CMR is using policy hypernetworks to generate a part of the parameters in the re-ranking models. A new re-ranking model based on AE is proposed which offers an end-to-end joint optimal solution for complex business-oriented tasks. Offline and online experiments on Taobao app demonstrated the effectiveness of the proposed CMR framework and the re-ranking model.

REFERENCES

[1] 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).

[2] John R Birge and Francois Louveaux. 2011. *Introduction to stochastic programming*. Springer Science & Business Media.

[3] Fedor Borisyuk, Krishnaram Kenthapadi, David Stein, and Bo Zhao. 2016. CaS-MoS: A framework for learning candidate selection models over structured queries and documents. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 441–450.

[4] Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning* 11, 23-581 (2010), 81.

[5] Jaime Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. 335–336.

[6] Yoav Goldberg. 2017. Neural network methods for natural language processing. *Synthesis lectures on human language technologies* 10, 1 (2017), 1–309.

[7] Yu Gong, Yu Zhu, Lu Duan, Qingwen Liu, Ziyu Guan, Fei Sun, Wenwu Ou, and Kenny Q Zhu. 2019. Exact-k recommendation via maximal clique optimization. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 617–626.

[8] Qi Hao, Tianze Luo, and Guangda Huzhang. 2021. Re-ranking with constraints on diversified exposures for homepage recommender system. *arXiv preprint arXiv:2112.07621* (2021).

[9] Guangda Huzhang, Zhenjia Pang, Yongqing Gao, Yawen Liu, Weijie Shen, Wen-Ji Zhou, Qing Da, Anxiang Zeng, Han Yu, Yang Yu, et al. 2021. AliExpress Learning-To-Rank: Maximizing online model performance without going online. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[10] Dietmar Jannach. 2022. Multi-objective recommendation: Overview and challenges. In *Proceedings of the 2nd Workshop on Multi-Objective Recommender Systems co-located with 16th ACM Conference on Recommender Systems (RecSys 2022)*, Vol. 3268.

[11] Zhengbao Jiang, Ji-Rong Wen, Zhicheng Dou, Wayne Xin Zhao, Jian-Yun Nie, and Ming Yue. 2017. Learning to diversify search results via subtopic attention. In *Proceedings of the 40th international ACM SIGIR Conference on Research and Development in Information Retrieval*. 545–554.

[12] Chen Karako and Putra Manggala. 2018. Using image fairness representations in diversity-based re-ranking for recommendations. In *Adjunct Publication of the 26th Conference on User Modeling, Adaptation and Personalization*. 23–28.

[13] George Karypis, Eui-Hong Han, and Vipin Kumar. 2005. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)* 22, 1 (2005), 143–177.

[14] Yehuda Koren. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[15] Denis Kotkov, Shuaiqiang Wang, and Jari Veijalainen. 2016. A survey of serendipity in recommender systems. *Knowledge-Based Systems* 111 (2016), 180–192.

[16] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436–444.

[17] Dingcheng Li, Xu Li, Jun Wang, and Ping Li. 2020. Video recommendation with multi-gate mixture of experts soft actor critic. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1553–1556.

[18] Xi Lin, Zhiyuan Yang, Qingfu Zhang, and Sam Kwong. 2020. Controllable pareto multi-task learning. *arXiv preprint arXiv:2010.06313* (2020).

[19] Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qing-Fu Zhang, and Sam Kwong. 2019. Pareto multi-task learning. *Advances in neural information processing systems* 32 (2019).

[20] Jiongnan Liu, Zhicheng Dou, Xiaojie Wang, Shuqi Lu, and Ji-Rong Wen. 2020. DVGAN: a minimax game for search result diversification combining explicit and implicit features. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 479–488.

[21] 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).

[22] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Philip S Yu. 2017. Learning multiple tasks with multilinear relationship networks. *Advances in neural information processing systems* 30 (2017).

[23] Debabrata Mahapatra and Vaibhav Rajan. 2020. Multi-task learning with user preferences: Gradient descent with controlled ascent in pareto optimization. In *International Conference on Machine Learning*. PMLR, 6597–6607.

[24] Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).

[25] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. 2016. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3994–4003.

[26] Aviv Navon, Aviv Shamsian, Gal Chechik, and Ethan Fetaya. 2020. Learning the

[27] pareto front with hypernetworks. *arXiv preprint arXiv:2010.04104* (2020). Harrie Oosterhuis. 2021. Computationally efficient optimization of plackett-luce ranking models for relevance and fairness. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1023–1032.

[28] Dripta S Raychaudhuri, Yumin Suh, Samuel Schulter, Xiang Yu, Masoud Faraki, Amit K Roy-Chowdhury, and Manmohan Chandraker. 2022. Controllable Dynamic Multi-Task Architectures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10955–10964.

[29] Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098* (2017).

[30] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[31] Choon Hui Teo, Houssam Nassif, Daniel Hill, Sriram Srinivasan, Mitchell Goodman, Vijai Mohan, and SVN Vishwanathan. 2016. Adaptive, personalized diversity for visual discovery. In *Proceedings of the 10th ACM conference on recommender systems*. 35–38.

[32] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. *Advances in neural information processing systems* 28 (2015).

[33] Fan Wang, Xiaomin Fang, Lihang Liu, Yaxue Chen, Jiucheng Tao, Zhiming Peng, Cihang Jin, and Hao Tian. 2019. Sequential evaluation and generation framework for combinatorial recommender system. *arXiv preprint arXiv:1902.00245* (2019).

[34] Mark Wilhelm, Ajith Ramanathan, Alexander Bonomo, Sagar Jain, Ed H Chi, and Jennifer Gillenwater. 2018. Practical diversified recommendations on youtube with determinantal point processes. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 2165–2173.

[35] Long Xia, Jun Xu, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. 2016. Modeling document novelty with neural tensor network for search result diversification. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 395–404.

[36] Long Xia, Jun Xu, Yanyan Lan, Jiafeng Guo, Wei Zeng, and Xueqi Cheng. 2017. Adapting Markov decision process for search result diversification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 535–544.

[37] Ruobing Xie, Yanlei Liu, Shaoliang Zhang, Rui Wang, Feng Xia, and Leyu Lin. 2021. Personalized approximate pareto-efficient recommendation. In *Proceedings of the Web Conference 2021*. 3839–3849.

[38] Chen Xu, Sirui Chen, Jun Xu, Weiran Shen, Xiao Zhang, Gang Wang, and Zhenhua Dong. 2023. P-MMF: Provider Max-min Fairness Re-ranking in Recommender System. In *Proceedings of the ACM Web Conference 2023*. 3701–3711.

[39] Le Yan, Zhen Qin, Rama Kumar Pasumarthi, Xuanhui Wang, and Michael Bendersky. 2021. Diversification-aware learning to rank using distributed representation. In *Proceedings of the Web Conference 2021*. 127–136.

[40] Yongxin Yang and Timothy Hospedales. 2016. Deep multi-task representation learning: A tensor factorisation approach. *arXiv preprint arXiv:1605.06391* (2016).

[41] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Kumthekar, Zhe Zhao, Li Wei, and Ed Chi. 2019. Sampling-bias-corrected neural modeling for large corpus item recommendations. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 269–277.

[42] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. 2017. Deep sets. *Advances in neural information processing systems* 30 (2017).

[43] Xiao Zhang, Sunhao Dai, Jun Xu, Zhenhua Dong, Quanyu Dai, and Ji-Rong Wen. 2022. Counteracting user attention bias in music streaming recommendation via reward modification. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2504–2514.

[44] Xiao Zhang, Haonan Jia, Hanjing Su, Wenhan Wang, Jun Xu, and Ji-Rong Wen. 2021. Counterfactual reward modification for streaming recommendation with delayed feedback. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 41–50.

[45] Yu Zhang and Qiang Yang. 2021. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[46] Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, and Ed Chi. 2019. Recommending what video to watch next: a multitask ranking system. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 43–51.

[47] Yong Zheng and David Xuejun Wang. 2022. A survey of recommender systems with multi-objective optimization. *Neurocomputing* 474 (2022), 141–153.

[48] Ziwei Zhu, Jingu Kim, Trung Nguyen, Aish Fenton, and James Caverlee. 2021. Fairness among new items in cold start recommender systems. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 767–776.

[49] Tao Zhuang, Wenwu Ou, and Zhirong Wang. 2018. Globally optimized mutual influence aware ranking in e-commerce search. *arXiv preprint arXiv:1805.08524* (2018).