Conditional Sequential Slate Optimization

Yipeng Zhang*
University of California, Los Angeles
zyp5511@g.ucla.edu

Mingjian Lu
eBay Inc
mingjlu@ebay.com

Saratchandra Indrakanti
eBay Inc
sindrakanti@ebay.com

Manojkumar Rangasamy Kannadasan†
Facebook Inc
mkannadasan@fb.com

Abraham Bagherjeiran
eBay Inc
abagherjeiran@ebay.com

ABSTRACT

The top search results matching a user query that are displayed on the first page are critical to the effectiveness and perception of a search system. A search ranking system typically orders the results by independent query-document scores to produce a slate of search results. However, such unilateral scoring methods may fail to capture inter-document dependencies that users are sensitive to, thus producing a sub-optimal slate. Further, in practice, many real-world applications such as e-commerce search require enforcing certain distributional criteria at the slate-level, due to business objectives or long term user retention goals. Unilateral scoring of results does not explicitly support optimizing for such objectives with respect to a slate. Hence, solutions to the slate optimization problem must consider the optimal selection and order of the documents, along with adherence to slate-level distributional criteria. To that end, we propose a hybrid framework extended from traditional slate optimization to solve the conditional slate optimization problem. We introduce conditional sequential slate optimization (CSSO), which jointly learns to optimize for traditional ranking metrics as well as prescribed distribution criteria of documents within the slate. The proposed method can be applied to practical real world problems such as enforcing diversity in e-commerce search results, mitigating bias in top results and personalization of results. Experiments on public datasets and real-world data from e-commerce datasets show that CSSO outperforms popular comparable ranking methods in terms of adherence to distributional criteria while producing comparable or better relevance metrics.

CCS CONCEPTS

• **Computing methodologies** → **Knowledge representation and reasoning**.

KEYWORDS

slate optimization, recommendation system, reinforcement learning, e-commerce

*Work completed during internship at eBay
†Work completed during working at eBay

SIGIR eCom'21, July 15, 2021, Virtual Event, Montreal, Canada

ACM Reference Format:
Yipeng Zhang, Mingjian Lu, Saratchandra Indrakanti, Manojkumar Rangasamy Kannadasan, and Abraham Bagherjeiran. 2021. Conditional Sequential Slate Optimization. In *Proceedings of ACM SIGIR Workshop on eCommerce (SIGIR eCom'21).* ACM, New York, NY, USA, 10 pages.

1 INTRODUCTION

Ranking search results that match a user query is a critical component of information retrieval (IR) systems that power many popular websites. Typically, learning to rank (LTR) models are developed to score and sort the results, and they are trained based on labels obtained from user engagement with results. While pairwise or listwise loss functions are most commonly employed in training LTR models, scoring the results during inference is usually done in a unilateral fashion. Such an approach of scoring results independently fails to take inter-document dependencies into consideration. However, studies show that the perception of a search result is influenced by other neighboring results [1, 20, 27]. Such behavior is even more pronounced in domains like e-commerce search, where users typically compare and contrast choices presented on a results page as part of their shopping process. Modeling such dependencies is critical within the top results (for instance, results shown on the first page), which users predominantly engage with. Moreover, in many real-world applications, there is a requirement to ensure certain predetermined distributional criteria within results on a search results page. Managing the holistic composition of results within the page is critical towards user perception of a website, and is often driven by business objectives or long terms goals aimed at objectives such as user retention. For instance, ensuring that the spectrum of selection available for a query is represented within the top results is an important objective at eBay. Other examples of such business objectives include showcasing diversity in e-commerce inventory, mitigating bias in top results or personalization of results. Hence, the constitution of an optimal set of top results (documents, or referred to as items in this paper) must take any distributional criteria and inter-result dependencies into account, along with the individual goodness of results.

We refer to the top results for a search query, such as k items shown on the first page as perceived by the user (where $k \ll n$, n is the number of results in the recall), as the slate. In this paper, we present a novel approach and an architecture to generate an optimal slate. An optimal slate is envisioned as a slate that maximizes the likelihood of fulfilling search needs while satisfying predetermined query-specific distributional criteria. We jointly optimize for classical ranking metrics and adherence of the slate

to distributional criteria, while modeling intra-slate dependencies. Various approaches have been presented by researchers to produce a set-aware ranking towards slate optimization. Heuristic methods such as Maximal Marginal Relevance (MMR) [7], IA-Select [2], and xQuAD [24] aim to incorporate diversity into a slate or minimize user dissatisfaction as users browse the returned results. These approaches, however, tend to utilize simpler heuristics that may lack the expressive power of the more complex models that are increasingly being used to capture inter-document dependencies. More recent models adopt set-aware ranking architectures [3, 5]. By modeling a representation of the candidate set of documents and scoring documents cognizant of the candidate set, these approaches aim to produce an optimal slate in a set-aware fashion. For example, groupwise scoring functions (GSF) [3] proposes multivariate scoring functions to jointly score documents and Seq2Slate [5] adopts pointer networks to score documents sequentially. While these approaches are set-aware during inference, they are tailored to optimize classic ranking metrics such as normalized discounted cumulative gain (nDCG). They do not explicitly model the additional distributional criteria, which is an important consideration in practice. To the best of our knowledge, there are no existing neural network based methods directly addressing this specific problem observed in search and specifically e-commerce.

To that end, we propose conditional sequential slate optimization (CSSO), a novel framework that aims to address the slate optimization problem. It incorporates a sequential decoding approach built upon pointer network, that, during sequential decoding, manages the composition of the slate as per predetermined distributional criteria, while jointly optimizing traditional ranking metrics such as nDCG. This approach treats slate optimization problem as a re-ranking problem, where a *base-ranker* produces a ranking based on a unilateral scoring of items, and the *re-ranker* consumes a set of n ranked items to produce a slate of size k, where $k \ll n$. We express query-specific distributional criteria with respect to the slate by means of representative or domain-specific features of the items being ranked.

The hybrid optimization goal might lead to divergent optimization directions for each term as the model has to balance among ranking quality and all aspects within the predetermined distributional criteria set. Furthermore, due to the existence of non-unique solutions and the non-convex nature of this problem, coupled with the non-differentiable nature of metrics like nDCG, it is hard for traditional gradient based methods to find an optimal slate. To tackle this novel and challenging problem, we employ Reinforcement Learning (RL) for the training of CSSO. RL provides a systematic way to search in the feature space while progressively improving the non-differentiable optimization goal with the aid of policy gradient method. At the same time, with the exploration strategy in RL, the learnt model is able to generalize better than traditional supervised method by efficiently learning the meaningful features from the rollouts.

In the training, we utilize items' click-through data from IR systems as implicit relevance labels. Within the RL paradigm, a state is represented *jointly* by the current items on the slate, the distance from the predetermined distribution, and the encoded candidate set. An action is tantamount to the item selection process in each step of decoding, and reward is determined on the basis of the slate composition after the decoding process by considering both ranking quality and adherence to distributional criteria. The optimization problem, thus well defined, can be solved by policy gradient in the finite horizon with length k, and we adopt a baseline model to reduce the variance in training. The key contributions of our work can be summarized as follows:

- A framework to jointly optimize a slate for document relevance as well as adherence to a predetermined distribution, thus attempting to address an industry pain point of re-ranking to personalize, mitigate bias or introduce diversity in results.
- An enhancement to pointer network to include a conditional structure that facilitates representing slate composition during the decoding phase.

In experiments performed on public and proprietary e-commerce datasets, CSSO produces results with comparable ranking metrics and better distribution conformity when compared with state-of-the-art (SOTA) slate optimization methods. On Yahoo [8] and Web30k [23] datasets with additional simulation for imitating users' real world behaviors, we demonstrate that both nDCG and GAP (distribution conformity metric which will be defined in the sections that follow) can be simultaneously optimized in the ranking task. Experiments on datasets from eBay.com search sessions show a significant improvement in GAP metrics and comparable ranking metrics to production ranker. In general, CSSO performs well with respect to both nDCG and GAP. We view CSSO as a practical approach to solve a very relevant industry problem of optimizing a slate for relevance and slate composition based on business objectives. We envision that the approach can be applied to solve problems such as ensuring diversity, mitigating bias, and adhering to business objectives across search and recommendation systems in various web domains including e-commerce, online media and web search.

2 RELATED WORK

Major LTR methods include pairwise and listwise approaches. Classic works within these two categories are widely used in current ranking production. For pairwise approaches, there are RankSVM [19] and RankBoost [12], LambdaMART [28], where the input is a feature vector of each single document and the output is a score for each single document; For listwise approaches such as ListNet [6], AdaRank [30] as well as LightGBM [22], the input of these models is a set of document features associated with a query and output is a ranked list. All these approaches focus on learning the optimal way of combining features through training. However, these methods score documents independently and fail to take inter-item dependencies into consideration during inference [16].

During inference, the top-k items in the slate could be recommended in a holistic manner considering them as a set rather than independent items. For example, instead of ranking each item independently, GSF [3] proposes a deep neural network based algorithm such that the relevance score of a document is determined jointly by multiple documents in the list. Different from just using relevance score to recommend, List-CVAE [17] uses a deep generative model to pick k items closest to each desired embedding from the original n items. Moreover some recent works are focusing more on using

deep reinforcement learning as an optimization tool and show performant results, such as [4, 9, 10, 15]. The most comparable work to ours is Seq2Slate [5] which defines the slate optimization problem as a sequential reranking problem. However, these methods mostly focus on the ranking quality of the slate but lack a distributional objective, which is critical in many real world and specifically e-commerce applications.

There are few works directly optimizing for the predetermined distributional criteria on the slate, such as [11, 26]. Moreover, diversification algorithms have been used to specifically diversify the search results without explicitly considering the distribution of the ranking results. Heuristic diversification algorithms such as [2, 7, 24] serve as a post-processing layer on the ranking score produced by original rankers. Some deep learning algorithms [18, 29] were also proposed with diversification loss terms to jointly optimize for diversity and ranking quality. However, these works do not account for query-specific distribution preferences, limiting the ability to tailor recommendations to specific queries. Our approach aims to expand the scope of query-specific slate optimization to fulfill search needs better and facilitate managing slate composition compared to previous methods.

3 APPROACH

3.1 Slate optimization overview

Typically the slate presented to a user is constructed by refining a base ranking result via re-ranking. We define slate optimization as a re-ranking problem that optimizes the slate holistically. For a query q, an ordered candidate set $\mathcal{X}$ consisting of n items is provided by a base ranker. Each item i is associated with a feature vector $\mathbf{x}^{(i)}$ $(i = 1, \ldots, n)$ where $\mathbf{x}^{(i)} \in \mathbb{R}^m$ and m is the dimension of the feature vector. In our problem, the slate optimization model selects the top-k ($k \ll n$) items from a permutation $\boldsymbol{\sigma}$ of the ordered candidate set $\mathcal{X}$ such that both the ranking and distribution metrics are jointly optimized. It has shown success in realistic applications in ranking items sequentially, where the re-ranker selects the next best item based on the items already selected. Inspired by the Seq2Seq model [25], the re-ranker selects items from the candidate set sequentially, generating a_t with the probability $p(a_t|a_1, \ldots, a_{t-1}, \mathcal{X})$, where $a_t \in \{1, \ldots, n\}$ denotes the index of the element in the input for the t^{th} selection. The top k ranked elements in the permutation are hereby denoted as $\boldsymbol{\sigma}_k$ i.e. $\boldsymbol{\sigma}_k = [a_1, \ldots, a_k]$.

3.2 Measure distribution divergence

To rigorously define the parameters of the re-ranker, we first introduce some notations. Within the feature vector of the i^{th} item, $\mathbf{x}^{(i)} \in \mathbb{R}^m$, categorical features are one-hot encoded. We denote a categorical variable using an index set $F_j \subset \{1, \ldots, m\}$ that stands for indices in the feature vector corresponding to the categorical variable, where $j \in \{1, \ldots c\}$ and c is the number of categorical variables. All of the categorical variables of interest can be denoted as $\mathcal{F} = \{F_1, \ldots, F_c\}$. For example, consider $m = 6$, and the feature vector of the i^{th} item $\mathbf{x}^{(i)} = [0.3, 0, 1, 1, 0, 5]^T$, where the first and sixth features are continuous, and the second, third, fourth and fifth features are one-hot representation of categorical variables $F_1 = \{2, 3\}$ and $F_2 = \{4, 5\}$. Here 2, 3, 4, 5 correspond to the indices of features in item $\mathbf{x}^{(i)}$. We define $\mathbf{x}^{(i)}_{F_j}$ as the categorical features belonging to F_j for the i^{th} item, i.e. $\mathbf{x}^{(i)}_{F_1} = [0, 1]^T$, $\mathbf{x}^{(i)}_{F_2} = [1, 0]^T$.

The slate distribution of a specific categorical variable F_j on the slate of size k is defined as the percentage of each categorical feature belonging to that categorical variable: $\mathbf{r}_j = \frac{1}{k}\sum_{t=1}^{k} \mathbf{x}^{(a_t)}_{F_j}$, $j = 1, \ldots, c$. Note that the sum of each element in $\mathbf{r}_j$ equals to 1. Accordingly, we can now specify the set of predetermined distributional criteria for a query q as $\mathcal{D}_q = \{\mathbf{d}_1, \ldots, \mathbf{d}_c\}$ where $\mathbf{d}_j$ is the predetermined distributional criterion as compared with $\mathbf{r}_j$. For example, $\mathbf{d}_1 = [0.3, 0.7]$ means that the first selected categorical variable is associated with two categories, and we would like the distribution of this feature in the slate, $\mathbf{r}_1$, to be: 30% of the first category, 70% of the second category. For the generated slate, we would like some of the items' categorical features to follow their corresponding predetermined distributional criteria. In order to measure the distance between a slate distribution $\mathbf{r}_j$ and it's distribution criterion $\mathbf{d}_j$, we introduce the categorical distribution gap as the metric which is defined as the max of the element-wise absolute distance between $\mathbf{r}_j$ and $\mathbf{d}_j$. Furthermore, we define the whole slate distribution gap (GAP) as the average of the categorical distribution gaps as shown in Equation 1

$$\mathrm{GAP}(\mathcal{D}_q, \{\mathbf{r}\}_{j=1}^{c}) = \frac{1}{c}\sum_{j=1}^{c} \|\mathbf{d}_j - \mathbf{r}_j\|_\infty. \tag{1}$$

3.3 Model architecture

To build a model that is capable of selecting items sequentially given an input sequence of item features, we adopt the pointer network architecture that can sequentially select items in the input sequence. In general, our CSSO does not only select item based on items already on the slate, but the selection is also based on the conditional information guiding the model to optimize towards the predetermined query-based distributional criteria.

The typical pointer network consists of encoder and attention decoder, which both contain RNNs to extract sequential information. In our architecture in Figure 1, we use LSTM [14] for encoder and decoder, while keeping the attention mechanism in the decoder. The items from the input sequence $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$ are first embedded by the embedding layer to generate $\mathbf{x}_e^{(1)}, \ldots, \mathbf{x}_e^{(n)}$. Then the LSTM in the encoder computes a sequence of outputs $H_{en} = (h_1, \ldots, h_n)$ iteratively:

$$h_t = \mathrm{Encoder}\Big(\mathbf{x}_e^{(t-1)}, h_{t-1}\Big). \tag{2}$$

The encoder encodes the embedded input sequence through the hidden state of the LSTM and the last hidden state of the encoder h_n can be viewed as a compact representation of the entire input sequence, which is sent to the decoder as the initial hidden state for the sequential decoding. Meanwhile the output of the encoder is also sent to the decoder for computing the attention matrix. In decoding step t, the decoder outputs an n-dimensional vector $\mathbf{u}_t$ to assign one score for each items in the input sequence. In order to prevent duplicate selection of items in the decoding process, we properly mask $\mathbf{u}_t$ before passing it into a softmax to produce a probability distribution, which ensures the probabilities of previously selected items are 0. Then the index of the next item a_t to put in the slate is sampled following this probability (sampling

strategy) or picked greedily (picking the one with maximal probability). After the selection, the new item's embedding $\mathbf{x}_e^{(a_t)}$ and the output of the hidden state are fed to the decoder for next step decoding. Additionally, in order to guide the model to select items based on the distributional criteria, we feed conditional information $CI_t = \{\mathbf{d}_j - \mathbf{r}_j^{(t)}\}_{j=1}^c$ where $\mathbf{r}_j^{(t)} = \frac{1}{t}\sum_{i=1}^t \mathbf{x}_{F_j}^{(a_i)}$ into the decoder as well, and update CI_t after each decoding step. Therefore, the model is aware of the current status on the slate and selects next items to approach the distributional criteria. The general decoding steps follow the equation below:

$$(h_t, \mathbf{u}_t) = \text{Decoder}\Big(h_{t-1}, \mathbf{x}_e^{(a_t)}, CI_t\Big). \tag{3}$$

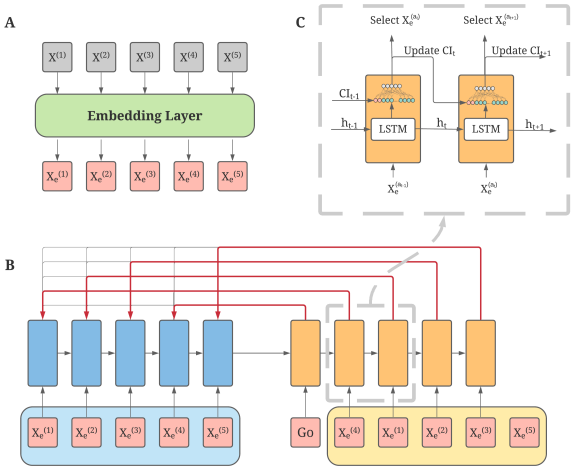

Figure 1: Architecture of the CSSO model. For the purpose of illustration, we assume that $n = k = 5$ and $x_{F_1}^{(i)}$ $(c = 1)$ is a two dimensional vector, and omit the attention mechanism in the figure. (A) The input sequence is transformed by the embedding layer. (B) The embedded input sequence is fed into the encoder LSTM to generate a compact representation of the entire sequence, then the attention decoder selects each item sequentially. (C) In the t^{th} step of decoding, the hidden state of decoder LSTM h_{t-1}, the condition information CI_{t-1}, and the previous selected item's embedding $\mathbf{x}_e^{(a_{t-1})}$ are fed into the decoder to guide the model to select the next best item a_t. Then we compute CI_t for next step of decoding.

3.4 Optimal slate

We define the optimal slate as both satisfying the predetermined distributional criteria and maximizing the ranking metrics. We adopt the GAP metric introduced in previous section to measure the distance of the categorical distributions on the slate to the distributional criteria. For the ranking quality metrics, we use nDCG calculated from the relevance or click-through information y_i for i^{th} item of the query following the reranking order. We introduce a weight α to balance between these two criteria which we want to optimize at the same time, i.e. maximizing the nDCG and minimizing the GAP. Concretely, the final metric of the slate with size k can be written in Equation 4.

$$\begin{aligned}\mathcal{R}(\boldsymbol{\sigma}_k, \mathbf{y}, \mathcal{D}_q, \{\mathbf{r}\}_{j=1}^c) =& \alpha\ (\text{nDCG@k}) \\ &- (1-\alpha)\text{GAP}(\mathcal{D}_q, \{\mathbf{r}_j\}_{j=1}^c)\end{aligned} \tag{4}$$

Note, the higher $R(\boldsymbol{\sigma}_k, \mathbf{y}, \mathcal{D}_q, \{\mathbf{r}\}_{j=1}^c)$ represents better permutation $\boldsymbol{\sigma}_k$ in other words, the slate.

3.5 RL based training

The employment of RL could enable the model to directly optimize for non-differentiable slate goodness measurement such as nDCG and distribution GAP. The crucial components in reinforcement learning are state, action, and reward.

The state summarizes the candidate set and the status of the slate generation. The action is to pick an item. An action transits the agent from a state to the next. In the sequential decoding, the agent goes through $k+1$ states, taking k actions. Concretely, in the tth step, we include the following components in the state s_t: the encoded candidate set H_{en} (for attention mechanism), the condition information CI_t, and the selection information till step t embedded in the hidden state h_t of the decoder LSTM. The action $a_t \sim \pi_\theta(\cdot|s_t)$ is defined as the agent's selection on the candidate set. Recall that $\pi_\theta(\cdot|s_t)$ is the output probability distribution normalized from $\mathbf{u}_t$ of the decoder and we have restricted that the probability in previously selected items' locations to be 0, i.e. $\pi_\theta(a_t = a_i|s_t) = 0, \forall i < t$ to forbid duplicate selections. The optimal policy will be able to generate the optimal slate by taking k actions.

The optimization goal of the CSSO in RL setting is to maximize the expected return $\mathcal{R}(\boldsymbol{\sigma}_k, \mathbf{y}, \mathcal{D}_q, \{\mathbf{r}\}_{j=1}^c)$ which is a function of permutation $\boldsymbol{\sigma}_k$, item relevance $\mathbf{y}$, distributional criteria $\mathcal{D}_q$, and slate distribution $\{\mathbf{r}\}_{j=1}^c$.

$$\max_\theta \quad \mathbb{E}_{a\sim\pi_\theta}[\mathcal{R}(\boldsymbol{\sigma}_k, \mathbf{y}, \mathcal{D}_q, \{\mathbf{r}\}_{j=1}^c)] \tag{5}$$

One can use REINFORCE, a policy gradient method, together with stochastic gradient ascent to optimize policy parameter θ. The gradient is computed by the equation below:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[G_t \nabla_\theta \ln \pi_\theta(a_t|s_t)]. \tag{6}$$

Here the return G_t for $t = 1, \ldots, k$ is $\mathcal{R}(\boldsymbol{\sigma}_k, \mathbf{y}, \mathcal{D}_q, \{\mathbf{r}\}_{j=1}^c)$ defined by Equation 4, which is a linear combination of nDCG@k and GAP based on the top k items from $\boldsymbol{\sigma}_k$. G_t is the same for all t.

By introducing variance reduction technique, the parameters of the model can be updated in a batch with size N via Equation 7. Note that $\mathcal{R}_i$ is the reward of the i^{th} training sample in the batch:

$$\begin{aligned}&\nabla_\theta J(\theta) \approx \sum_{i=1}^N \frac{1}{N}\left[(\mathcal{R}_i - b)\sum_{t=1}^k \nabla_\theta \ln \pi_\theta(a_t|s_t)\right], \\ &\text{where} \quad b = \sum_{i=1}^N \frac{\mathcal{R}_i}{N}.\end{aligned} \tag{7}$$

3.6 Addressing the shortcoming of RL

The nDCG and GAP used in RL are not differentiable, hence limiting the usage of much more efficient gradient-based methods that directly optimize for the optimal slate. Moreover, RL suffers from low sampling efficiency and is often not stable to train. Therefore we introduce a differentiable objective to evaluate the action at each

decoding step to enable direct supervised learning. For the sake of simplicity and consistency, we borrow the notation s_t and $\pi_\theta(\cdot|s_t)$ from section 3.5. s_t is defined as the combination of the encoded candidate set H_{en}, the condition CI_t, and the selection information till step t embedded in the hidden state h_t of the decoder LSTM, while $\pi_\theta(\cdot|s_t)$ is defined as the output probability distribution normalized from $\mathbf{u}_t$ given by the decoder.

The objective takes two parts into account: the ranking quality and the distributional conformity. For optimizing the ranking quality, we formulate the problem in a supervised learning setting by posing some selection preferences. Instead of optimizing for the whole slate after k steps selections, at each step, the items with higher relevance or click-through labels are preferred among the remaining set. However, there is no unique answer for the decision at each step, so we want the model to equally prefer the items with the same relevance/click-through label. The selection at a single step can therefore be framed as a multiclass classification problem: the model assigns a probability for each of the candidate item. Therefore, at the step t of decoding, we want the model to output a probability distribution $\mathbf{p}_t = \pi_\theta(\cdot|s_t)$ aligned with the distribution given by the labels $\mathbf{y}_t$ after masking. Specifically, the p_i within vector $\mathbf{p}_t$ from previous selections are zero-masked and their corresponding labels are also masked to ensure we only consider the distribution over the labels from the remaining items, i.e.

$$\mathbf{y}_t = [y_1, \ldots, y_n], \quad \text{where} \quad y_{a_i} = 0, \forall i < t. \tag{8}$$

Then we define the step-wise loss for the ranking metric as

$$l_{rank}(\mathbf{p}_t, \mathbf{y}_t) = -\sum_i^n \frac{y_i}{\sum_j y_j} \log(p_i). \tag{9}$$

Next, the slate ranking loss can be defined as the weighted sum of each step's loss, where the weights are to mimic the logarithmic decay in nDCG which takes the ranking position into account. At each step, the label $\mathbf{y}_t$ and $\mathbf{p}_t$ change according to selections from previous steps. Therefore, we define $\mathcal{P} = \{\mathbf{p}_t\}_{t=1}^k$ and $\mathcal{Y} = \{\mathbf{y}_t\}_{t=1}^k$ to denote the set of model probabilities and the corresponding labels among k decoding steps respectively.

$$L_{rank}(\mathcal{P}, \mathcal{Y}) = \sum_{t=1}^k w_t \; l_{rank}(\mathbf{p}_t, \mathbf{y}_t), \; w_t = \frac{1}{\log(t+1)}. \tag{10}$$

For optimizing the feature distribution quality, we formulate distribution on the set as the sum of the step-wise distribution selection. At each step, we can compute the step-wise selection distribution,

$$\boldsymbol{\gamma}_j^{(t)} = \sum_{i=1}^n p_i x_{F_j}^{(i)}, \tag{11}$$

which is the feature distribution among the candidate set marginalizing over the model's selection probabilities. Note that $p_{a_i} = 0, \forall i < t$ to prevent double counting and $\boldsymbol{\gamma}_j^{(t)}$ is differentiable, w.r.t. the model parameter θ. Therefore the slate distribution for a specific j is the sum of the step-wise selection distribution.

$$\mathbf{r}'_j = \frac{1}{k} \sum_{t=1}^k \boldsymbol{\gamma}_j^{(t)} \tag{12}$$

Recall that we define the distance between category distribution on the slate and the distributional criteria as GAP, so we can formulate the differential version of GAP as

$$\text{GAP}_\theta(\mathcal{D}_q, \{\mathbf{r}'\}_{j=1}^c) = \frac{1}{c} \sum_{j=1}^c \|\mathbf{d}_j - \mathbf{r}'_j\|_\infty. \tag{13}$$

By having the slate-wise ranking and distribution quality loss, we again introduce a weight α to balance between the ranking quality and distributional criteria. Additionally we include a hyperparameter β as these two loss terms are not in the same scale.

$$\begin{aligned} &\mathcal{L}_\theta(\{\mathcal{P}, \mathcal{Y}, \{\mathbf{r}'\}_{j=1}^c, \mathcal{D}_q) = \\ &\qquad \alpha\beta L_{rank}(\mathcal{P}, \mathcal{Y}) + (1-\alpha)\text{GAP}_\theta\left(\mathcal{D}_q, \{\mathbf{r}'_j\}_{j=1}^c\right) \end{aligned} \tag{14}$$

The optimization goal of the CSSO in supervised learning setting is to minimize the loss $\mathcal{L}_\theta(\mathcal{P}, \mathcal{Y}, \mathcal{D}_q, \{\mathbf{r}'\}_{j=1}^c)$ of the slate which is a function of output probability in each step $\mathcal{P} = \{\mathbf{p}_t\}_{t=1}^k$, item relevance $\mathcal{Y}$, distributional criteria $\mathcal{D}_q$, and slate distribution $\{\mathbf{r}'\}_{j=1}^c$.

$$\min_\theta \quad \mathbb{E}_{\pi_\theta}\left[\mathcal{L}_\theta(\mathcal{P}, \mathcal{Y}, \mathcal{D}_q, \{\mathbf{r}'\}_{j=1}^c)\right] \tag{15}$$

Since we need to minimize the loss, we adopt the gradient descent. The model parameters can be updated similarly as trained by RL via Equation 16.

$$\begin{aligned} &\nabla_\theta J(\theta) \approx \sum_{i=1}^N \frac{1}{N}\left[(\mathcal{L}_i - b)\sum_{t=1}^k \nabla_\theta \ln \pi_\theta(a_t|s_t) + \mathcal{L}_{\theta_i}\right] \\ &\text{where} \quad b = \sum_{i=1}^N \frac{\mathcal{L}_i}{N} \end{aligned} \tag{16}$$

4 EXPERIMENTS

In this section, we measure the performance of our CSSO model and the state-of-the-art on two popular learning-to-rank datasets and one proprietary dataset from ebay.com search session.

4.1 Implementation details

The embedding layer in the policy network (pointer network) is a fully-connected (FC) layer projecting the input features as 256-dimension vectors followed by a ReLU and dropout with a 10% dropout rate. LSTMs in both encoder and decoder have one hidden layer with 256 units. The output hidden state from the decoder is fed into two consecutive FC layers with ReLU activation, 10% dropout, and batch normalization in between. Then softmax is applied to produce the probability distribution with the same length as the candidate sequence. The baseline (b) in equation 7 and 16 is the running average with 0.99 exponential decay to whiten the return. The policy network is trained using the AdaBelief optimizer [31] with the learning rate of 1e−4 and mini-batches of 1024 training examples. The training of the policy network uses the sampling strategy. At inference time, we report metrics from the greedy decoding as the evaluation results. To avoid overfitting, we employ a validation set to validate the model performance.

4.2 Learning-to-Rank benchmarks

The experiments are conducted on two popular learning-to-rank datasets which are the Yahoo Learning to Rank Challenge dataset (set 1) and the MSLR-WEB30k (Web30k) dataset.

4.2.1 Dataset simulation. There are two barriers preventing the direct usage of the datasets. First, both datasets do not have prescribed query-based distributional criteria. Second, the labels in both public datasets are per-item relevance scores, lacking higher-order interactions between the clicks that are prevalent in real world scenarios. Thus, we augment these two datasets. First, we select binary feature columns with highest in-query variance, i.e. within a query the feature is not highly skewed towards one category. We then establish distributional criteria for each query on selected column feature. Next, we simulate user interactions to incorporate higher-order interactions between the clicks based on methodology adopted from [21].

The detailed procedure and parameters are in Appendix A. In sum, we simulate 7 datasets from Yahoo and Web30k datasets with their corresponding query-based distributional criteria objectives as shown in Table 1. In our experiments, we start with optimizing for one distributional criterion using Dataset $\{1, 2, 3, 4, 5, 6\}$ to take a close look at how nDCG and GAP are balanced in the optimal solutions given by various algorithms, and how CSSO generalizes among different datasets. Next, we investigate how CSSO performs over a more complex setup i.e. jointly optimizing two distributional criteria using Dataset 7.

Table 1: Simulated dataset for benchmark

Number	DatasetName	BaseRanker	$\mathcal{D}_q$ Column(s)
1	Yahoo	LightGBM	628
2	Yahoo	MART	628
3	Web30k	LightGBM	95
4	Web30k	LightGBM	99
5	Web30k	LambdaMART	95
6	Web30k	LambdaMART	99
7	Web30k	LightGBM	95 & 99

4.2.2 Compare with State-of-The-Art. We compare CSSO with SOTA ranking models: LightGBM, MART, and LambdaMART using the augmented datasets. As these SOTA methods do not directly optimize for the distributional criteria, we apply a variant of MMR algorithm to compare the model performance fairly. MMR is a reranking layer; by greedily selecting the item of largest score in sequence, it constructs a slate accounting for both distributional criteria and ranking quality. In MMR, the score is calculated by a linear combination of scores from the SOTA ranking models and the CI_t in each step. A factor λ is introduced to balance the model's preference between ranking quality and GAP. When $\lambda = 1$, the MMR purely optimizes for nDCG, and when $\lambda = 0$, it prioritizes the GAP over nDCG, and still considers ranking score when candidate items equally satisfy GAP metrics at one step of reranking. We describe our MMR algorithm in detail in Appendix B.

4.2.3 Experiments. In our experiments, we set slate length $k = 10$ and all performance metrics are computed on these 10 items. Since both nDCG and GAP are bounded between 0 to 1, we define the slate goodness, $R_s = 0.5\text{nDCG} - 0.5\text{GAP} + 0.5$ considering the ranking quality and the distributional criteria evenly. The R_s is one of the main metrics used to compare performances between models, and it is designed to be in range from 0 to 1.

We train two CSSO models with reinforcement learning (*CSSO-RL*) and supervised learning (*CSSO-SL*) respectively. Additionally, we set $\beta = 0.1$ to scale the ranking loss term for CSSO-SL as it is accumulated over the length of the slate. We adopt early stopping on validation set to avoid overfitting and report the R_s on the test set. Since MMR could produce multiple combinations of nDCG and GAP with varying λ values, we report the best R_s among all combinations in the following tables.

Yahoo dataset. We generate Dataset 1 and 2 based on the Yahoo dataset as described in Table 1 and Appendix A. In Table 2, we compare the performance on the test set with LightGMB+MMR (denoted as LightGMB for short) and MART+MMR (MART) with their best R_s respectively. The table displays that on Dataset 1 and 2, two CSSO models perform well compared with LightGBM and MART in term of R_s. Further, to better visualize the performance comparison with LightGBM and MART using the full range of λ, we plot the nDCG against GAP for different λ in Figure 2. We observe that with similar GAP metrics, CSSO can have a much better nDCG value than LightGBM and MART.

Table 2: Benchmark on Dataset 1 and Dataset 2

	Dataset 1			Dataset 2		
Metrics@10	nDCG ↑	GAP ↓	R_s ↑	nDCG ↑	GAP ↓	R_s ↑
CSSO-RL	0.782	0.045	**0.869**	0.779	0.054	**0.863**
CSSO-SL	0.783	0.063	**0.860**	0.775	0.072	0.852
LightGBM	0.716	0.040	0.838	0.742	0.042	0.850
MART	0.642	0.038	0.802	0.755	0.044	0.855

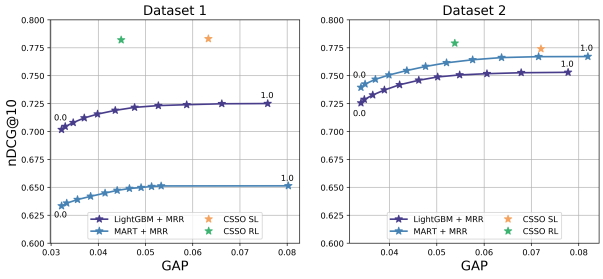

Figure 2: Performance comparisons as we swept the λ value in MMR from 0 to 1. Note that a good slate requires a higher nDCG and a lower GAP. When CSSO-RL and CSSO-SL have similar GAP metrics as SOTA algorithms, our proposed models result in significant improvements in nDCG.

Web30k. We generate Dataset 3, 4, 5, and 6 based on Web30k dataset as described in Table 1 and Appendix A. We explore the performance of CSSO to demonstrate the generalizability of the model in different datasets and various distributional criteria. Each dataset, in this section, is produced from different feature columns and base rankers. In Table 3, we compare the performance of CSSO-RL and CSSO-SL versus LightGBM+MMR (LightGBM) and LambdaMART+MMR (LambdaMART) on the data simulated from the same base ranker over different distributional criterion (Dataset 3 & Dataset 4). From Table 3, we observe that our proposed models significantly outperform the LightGBM and LambdaMART in terms of the R_s which indicates the generated slates with high ranking quality and distributional conformity.

Table 3: Benchmark on Dataset 3 & Dataset 4

	Dataset 3			Dataset 4		
Metrics@10	nDCG ↑	GAP@95 ↓	R_s ↑	nDCG ↑	GAP@99 ↓	R_s ↑
CSSO-RL	0.707	0.038	**0.834**	0.707	0.039	**0.834**
CSSO-SL	0.704	0.077	**0.814**	0.710	0.074	**0.818**
LightGBM	0.647	0.037	0.805	0.651	0.039	0.806
LambdaMART	0.635	0.036	0.799	0.638	0.037	0.800

Next, we conduct similar experiments as in Table 3 with a different base ranker (Dataset 5 & Dataset 6). We demonstrate the performance comparisons in Table 4. In this experiment, the CSSO-RL consistently outperforms the SOTA algorithms LightGBM and LambdaMART. However, CSSO-SL does not stand out in term of R_s. By comparing the break down of each term in R_s, we observe that CSSO-SL still produces a higher nDCG but with worse distributional conformity. Note that the SOTA algorithms can produce extremely low GAP values, which is due to the greedy selection strategy towards the distributional conformity. However, for CSSO-SL, the GAP is parameterized as a function GAP_θ then optimized jointly with ranking quality via gradient based method. We hypothesize that in this case, the gradient based method might fail to descend our complex optimization goal due to the non-convex property of the problem, which prevents the CSSO-SL from producing an extremely low GAP value.

Table 4: Benchmark on Dataset 5 & Dataset 6

	Dataset 5			Dataset 6		
Metrics@10	nDCG ↑	GAP@95 ↓	R_s ↑	nDCG ↑	GAP@99 ↓	R_s ↑
CSSO-RL	0.689	0.044	**0.823**	0.689	0.041	**0.824**
CSSO-SL	0.679	0.081	0.799	0.683	0.078	0.803
LightGBM	0.660	0.038	0.811	0.660	0.041	0.809
LambdaMART	0.671	0.037	0.817	0.674	0.040	0.817

Additionally we plot the trade-off between nDCG and GAP using different λ of the LightGBM and LambdaMART, and compare with our proposed framework in Figure 3.

In order to demonstrate the performance of CSSO models in a more complex setup, we train them to jointly optimize for two categorical distribution criteria (column 95 & 99) on Dataset 7. Table 5 shows that CSSO models outperform LightGBM and LambdaMART significantly on this complex problem. Compared with optimizing for single distributional criterion, jointly optimizing for multiple distributional criteria is harder due to the potentially contradicting optimization goals, and may result in larger GAP and lower

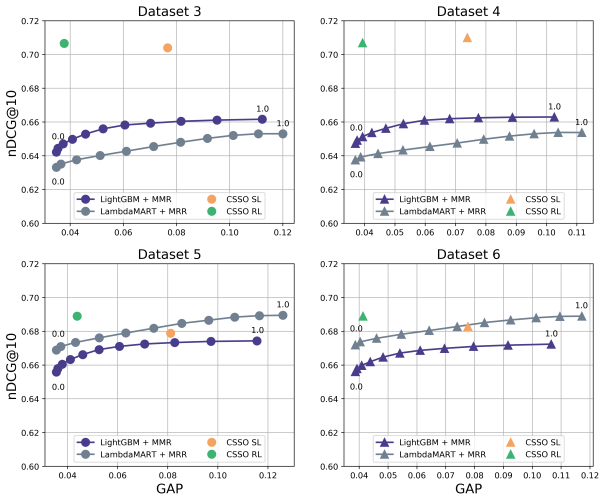

Figure 3: Performance comparisons between CSSO and SOTA algorithms on Dataset $\{3, 4, 5, 6\}$. It demonstrates the CSSO-RL significantly outperforms other SOTA algorithms in different datasets and different distributional criteria.

.

nDCG. Remarkably, both CSSO-SL and CSSO-RL achieve consistent and better performance on all categorical distribution criteria across Table 3 and 5. However, other algorithms have inferior performances. This suggests that CSSO is capable of optimizing for multiple categorical criteria robustly.

Table 5: Benchmark on Dataset 7

Metrics@10	nDCG ↑	GAP@95 ↓	GAP@99 ↓	GAP Avg ↓	R_s ↑
CSSO-RL	0.704	0.041	0.046	0.044	**0.830**
CSSO-SL	0.705	0.079	0.077	0.078	**0.814**
LightGBM	0.653	0.079	0.074	0.077	0.788
LambdaMART	0.649	0.108	0.100	0.104	0.772

Similarly, we plot the trade-off curve between nDCG and GAP, for these two distributional criteria, in Figure 4. The left panel is the detailed gap values for each distributional criterion sharing the same nDCG value, and the right panel is GAP (the average of categorical distribution gaps). The superior of both nDCG and GAP value from CSSO-RL and CSSO-SL demonstrates the CSSO models' potential to be versatile and stable tools for slate optimization.

4.3 Real-world data

We demonstrate the applicability of the model to a real-world use case using e-commerce search session data from ebay.com. CSSO models are trained as rerankers on top of a base ranker which produces unilateral scoring of eBay items. The aim of reranking, in this case, is to ensure diversity within the search results on the first page. Diversity is quantified based on the composition of the slate with respect to an ideal distribution established for a query computed from historic purchase patterns.

Training data is sampled from ebay.com search sessions, with user engagement with impressed items such as clicks and purchases used as relevance labels. The training data consists of hundreds

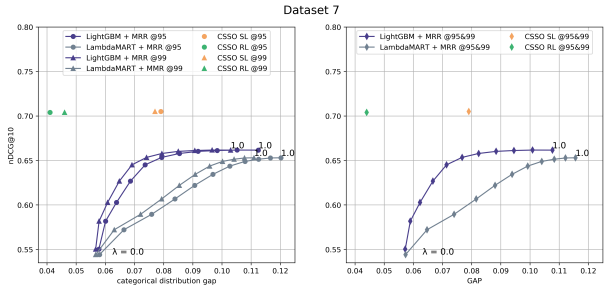

Figure 4: Performance comparisons for jointly optimizing two distributional criteria: column 95 & 99 on Dataset 7. We sweep the λ for MMR while tracking GAP and nDCG values. The left figure is the detailed gap values for two distributional criteria sharing the same nDCG value, and the right figure is the slate distribution gap (the average of gaps for each distributional criterion). For the purpose of illustration, we merge overlapped annotations of λ.

of thousands of search queries with the items impressive to users, their corresponding features, and the respective engagement received. Query-specific purchase behaviors are aggregated from historical search sessions to establish a distributional criterion. Desired distributions are computed for each query through an offline process using observed purchase distribution with respect to item-specific categorical features such as condition (for example, new, used, or refurbished conditions). Since labels are based on actual user engagement rather than human-judged samples, inter-item dependencies are inherently captured within the labels. CSSO is trained to optimize for nDCG with respect to the engagement-based relevance labels while adhering to the query-specific distributional criteria. In the reranking process, the input sequence is fed from the production ranker, and the conditional information is based on item-specific categorical features. Offline experiments on eBay search datasets show that CSSO produces a significant reduction in the GAP metric while producing a comparable nDCG with respect to production ranking models as shown in Table 6.

Table 6: Results from offline experiments on eBay dataset. Metrics are relative to the baseline production ranker and numbers reported as relative change due to confidentiality requirements. The first two rows are the performance of our proposed models for eBay dataset. For the ablation study (the last two rows), we train both models in the same setting but with removal of condition information and name them as CSSO-SL w/o CI_t and CSSO-RL w/o CI_t respectively. The results demonstrate that the utilization of CI_t significantly improves the GAP metrics.

Metrics@10	nDCG ↑	GAP ↓	R_s ↑
CSSO-SL	+0.995%	-6.265%	**+2.766%**
CSSO-RL	-0.984%	-7.931%	**+2.687%**
CSSO-SL w/o CI_t	+1.065%	-1.942%	+1.133%
CSSO-RL w/o CI_t	-0.619%	-2.184%	+0.613%

4.4 Ablation study

To understand how the conditional information guides the model to optimize for distribution conformity, we surgically remove the $CI_t = \{\mathbf{d}_j - \mathbf{r}_j^{(t)}\}_{j=1}^{c}$ in the model while keeping the same training setting; in each decoding step, there are no explicit guidance of the distance to the distribution conformity. We observed the inferior performance of GAP for both *CSSO-SL w/o* CI_t and *CSSO-RL w/o* CI_t as shown in last two rows of Table 6. While the ranking performance of the models is comparable, the substantially lower GAP metrics of the models with CI_t verifies the significance of introducing conditional information guidance in our CSSO.

4.5 Observations

Experiments show that CSSO-RL usually achieves better GAP than CSSO-SL. Reinforce based method might be able to search with exploration and jump out of some bad local minimum to produce a better slate with high ranking quality and good distribution conformity. However, in-spite of slightly inferior GAP metrics, CSSO-SL provides a good compromise by achieving its best performance at least five times faster than CSSO-RL and even faster than SOTA algorithms, indicating that the gradient-based method does overcome the sampling inefficiency of the reinforce-based method.

5 CONCLUSION

In this paper, we address an industry pain point of producing a slate with predetermined distributional criteria, while jointly optimizing for slate composition and ranking metrics. Conventional practices of employing unilateral scoring of results during inference, or using simple heuristics based rerankers produce sub-optimal slates. More recent approaches of expressive set-aware ranking models optimize exclusively for classic ranking metrics. To that end, we proposed CSSO, a novel reranking architecture building upon pointer network to produce an optimal slate, with optimality defined on a combination of relevance and holistic slate composition. We introduced a conditional structure to the decoding phase to represent a distributional preference and adherence to it. We constructed a reward and loss-function to capture the duality of ranking quality and adherence to desired slate composition and presented approaches to train the model within both the reinforcement learning and supervised learning paradigms. Experiments on Yahoo and Web30k public datasets with simulated click behavior and distributional preferences demonstrate the ability of CSSO to adhere to a prescribed slate composition while improving ranking and relevance metrics. Experiments on proprietary e-commerce datasets using real-world eBay search behavioral data, with the aim of introducing diversity in search results, show the ability of the model to outperform existing production rankers.

Top search results have a profound impact on user engagement and experience on most IR systems. Web-based businesses have both domain-specific and business-driven priorities that dictate the composition of the top results. We believe that the proposed architecture presents an approach to optimally model such priorities. Several critical real-world problems such as mitigating bias and introducing diversity in top results, addressing under-represented intents, and personalizing a slate based on user-specific priors can be systematically solved using CSSO.

REFERENCES

[1] Deepak Agarwal, Shaunak Chatterjee, Yang Yang, and Liang Zhang. 2015. Constrained optimization for homepage relevance. In *Proceedings of the 24th International Conference on World Wide Web*. 375–384.

[2] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. 2009. Diversifying Search Results. In *International Conference on Web Search and Data Mining (WSDM)* (international conference on web search and data mining (wsdm) ed.). Association for Computing Machinery, Inc. https://www.microsoft.com/en-us/research/publication/diversifying-search-results/

[3] Qingyao Ai, Xuanhui Wang, Sebastian Bruch, Nadav Golbandi, Michael Bendersky, and Marc Najork. 2019. Learning groupwise multivariate scoring functions using deep neural networks. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*. 85–92.

[4] Xueying Bai, Jian Guan, and Hongning Wang. 2019. A Model-Based Reinforcement Learning with Adversarial Training for Online Recommendation. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 10734–10745. https://proceedings.neurips.cc/paper/2019/hash/e49eb6523da9e1c347bc148ea8ac55d3-Abstract.html

[5] Irwan Bello, Sayali Kulkarni, Sagar Jain, Craig Boutilier, Ed Chi, Elad Eban, Xiyang Luo, Alan Mackey, and Ofer Meshi. 2019. *Seq2Slate: Re-ranking and Slate Optimization with RNNs*. Technical Report. https://arxiv.org/abs/1810.02019

[6] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to Rank: From Pairwise Approach to Listwise Approach. In *Proceedings of the 24th International Conference on Machine Learning* (Corvalis, Oregon, USA) *(ICML '07)*. Association for Computing Machinery, New York, NY, USA, 129–136. https://doi.org/10.1145/1273496.1273513

[7] 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.

[8] Olivier Chapelle and Yi Chang. 2010. Yahoo! Learning to Rank Challenge Overview. In *Proceedings of the 2010 International Conference on Yahoo! Learning to Rank Challenge - Volume 14* (Haifa, Israel) *(YLRC'10)*. JMLR.org, 1–24.

[9] Haokun Chen, Xinyi Dai, Han Cai, Weinan Zhang, Xuejian Wang, Ruiming Tang, Yuzhou Zhang, and Yong Yu. 2019. Large-scale interactive recommendation with tree-structured policy gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 3312–3320.

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

[11] Marina Drosou and Evaggelia Pitoura. 2010. Search result diversification. *ACM SIGMOD Record* 39, 1 (2010), 41–47.

[12] Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. 2003. An efficient boosting algorithm for combining preferences. *Journal of machine learning research* 4, Nov (2003), 933–969.

[13] Jerome Friedman. 2001. Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics* 29 (10 2001), 1189–1232. https://doi.org/10.2307/2699986

[14] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

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

[16] Saratchandra Indrakanti, Svetlana Strunjas, Shubhangi Tandon, and Manojkumar Kannadasan. 2019. Influence of Neighborhood on the Preference of an Item in eCommerce Search. In *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 2284–2288.

[17] Ray Jiang, Sven Gowal, Yuqiu Qian, Timothy Mann, and Danilo J. Rezende. 2019. Beyond Greedy Ranking: Slate Optimization via List-CVAE. In *International Conference on Learning Representations*. https://openreview.net/forum?id=r1xX42R5Fm

[18] 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.

[19] Thorsten Joachims. 2002. Optimizing Search Engines Using Clickthrough Data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Edmonton, Alberta, Canada) *(KDD '02)*. Association for Computing Machinery, New York, NY, USA, 133–142. https://doi.org/10.1145/775047.775067

[20] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. 2005. Accurately Interpreting Clickthrough Data as Implicit Feedback. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Salvador, Brazil) *(SIGIR '05)*. Association for Computing Machinery, New York, NY, USA, 154–161. https://doi.org/10.1145/1076034.1076063

[21] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased learning-to-rank with biased feedback. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. 781–789.

[22] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc., 3146–3154. https://proceedings.neurips.cc/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf

[23] Tao Qin and Tie-Yan Liu. 2013. Introducing LETOR 4.0 Datasets. *CoRR* abs/1306.2597 (2013). http://arxiv.org/abs/1306.2597

[24] Rodrygo L.T. Santos, Craig Macdonald, and Iadh Ounis. 2010. Exploiting Query Reformulations for Web Search Result Diversification. In *Proceedings of the 19th International Conference on World Wide Web* (Raleigh, North Carolina, USA) *(WWW '10)*. Association for Computing Machinery, New York, NY, USA, 881–890. https://doi.org/10.1145/1772690.1772780

[25] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2* (Montreal, Canada) *(NIPS'14)*. MIT Press, Cambridge, MA, USA, 3104–3112.

[26] Shubhangi Tandon, Saratchandra Indrakanti, Amit Jaiswal, Svetlana Strunjas, and Manojkumar Rangasamy Kannadasan. 2020. Addressing Purchase-Impression Gap through a Sequential Re-ranker. *arXiv preprint arXiv:2010.14570* (2020).

[27] Yue Wang, Dawei Yin, Luo Jie, Pengyuan Wang, Makoto Yamada, Yi Chang, and Qiaozhu Mei. 2016. Beyond ranking: Optimizing whole-page presentation. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. 103–112.

[28] Qiang Wu, Christopher JC Burges, Krysta M Svore, and Jianfeng Gao. 2010. Adapting boosting for information retrieval measures. *Information Retrieval* 13, 3 (2010), 254–270.

[29] 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.

[30] Jun Xu and Hang Li. 2007. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. 391–398.

[31] Juntang Zhuang, Tommy Tang, Yifan Ding, Sekhar Tatikonda, Nicha Dvornek, Xenophon Papademetris, and James S. Duncan. 2020. AdaBelief Optimizer: Adapting Stepsizes by the Belief in Observed Gradients. arXiv:2010.07468 [cs.LG]

APPENDIX

A DATASET SIMULATION PROCEDURE

In the absence of distributional criteria in the public datasets, we infer $\mathcal{D}_q$ by computing query-wise distributions of the selected categorical variables in the entire item set, i.e. $\mathbf{d}_j = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_{F_j}^{(i)}$. We treat binary features from the selected dataset as bi-class categorical variables and apply one-hot encoding to them. Specifically, for a query q, we infer $\mathbf{d}_j$ by computing the percentage of 1's and 0's in the selected column among all items belonging to it.

The distributional criteria are trivially satisfied if the feature is query-specific, i.e within a query the feature is highly skewed towards one category. To avoid that scenario, columns are selected because they show the highest within-query variance, allowing for meaningful GAP metric comparison and evaluation.

In order to simulate higher-order interactions between the clicks, we adopt the procedure proposed by Joachims et al .[21] to synthesize click-through data. We first train a base ranker model on the dataset with original relevance labels and rank items within each query on both datasets. Then we adopt the user "cascade" model by introducing a parameter η in order to quantify the decreasing attention as users browse items from higher rank to lower rank. Specifically, items in each query will have a probability of $1/i^{\eta}$ to be observed where i is the rank of the item. Subsequently, v sequences of users' interactions for a given query are sampled from the observation probability. The labels of sampled items will be converted to click-through labels (clicked: $\{2, 3, 4\}$, skip: $\{0, 1\}$). In addition, a "diverse clicks" mechanism adopted from previous work [5] is applied such that items similar to previously clicked items will not be clicked. Specifically, we use the median euclidean distance amongst all possible item-pairs for a given query as the threshold to determine if two items are "similar." In the end, we truncate the sequences that are longer than 30 items to the first 30 items and pad sequences that are shorter than 30 items with trailing padding value v_p for all columns. We choose $v = 25$ to emulate a real-world scenario in which there are 25 users interacting with the IR system for each query.

The details for generating the datasets are described below. We first train two different base rankers for each dataset respectively: MART [13] and LightGBM for the Yahoo dataset, LambdaMART and LightGBM for the Web30k dataset. The base rankers MART and LambdaMART are from the Ranklib package[1] and trained with default parameters. The parameter settings of LightGBM are consistent with official LightGBM benchmarks [22]. In the click simulation, we adopt the values of η from previous work [5], i.e. $\eta = 0.1$ for Yahoo and $\eta = 0.3$ for Web30k. We first infer the D_q on the original items belonging to each query q. After click simulation, each query is augmented into 25 different sub-queries with different subsets of candidates from the original candidate set. All sub-queries share the same query-based distributional criteria $\mathcal{D}_q$. We augment the data with inferred $\mathcal{D}_q$ and simulated click-through labels for selected combination of base ranker and feature selection $\mathcal{F}$ (for specifying $\mathcal{D}_q$). The simulation is applied on both training, validation, and testing set, and the overview of datasets is shown in Table 1.

[1] https://sourceforge.net/p/lemur/wiki/RankLib/

B VARIANT OF MMR

Algorithm 1: MMR algorithm for one query Q

Define: $C_{F_j} : \mathbb{R}^m \mapsto \mathbb{N}_+$, $C_{F_j}(\mathbf{x}) := \arg\max(\mathbf{x}_{F_j})$. That is, $C_{F_j}(\mathbf{x}^{(i)})$ returns the index of the category of the categorical variable F_j;

Input: Item categorical variable F_j, Item features $\mathcal{X} = \{\mathbf{x}^{(i)}\}_{i=1}^{n}$, Items score from ranker $\mathbf{s} \in \mathbb{R}^n$, Coefficient λ, Distributional criteria $\mathcal{D}_q = \{\mathbf{d_j}\}_{j=1}^{c}$;

Initialize empty slate SL with length = k, $\{\mathbf{d_j}'\}_{j=1}^{c} = \{\mathbf{d_j}\}_{j=1}^{c}$; $\mathbf{s}'[i] \leftarrow \dfrac{\mathbf{s}[i] - \min(\mathbf{s})}{\max(\mathbf{s}) - \min(\mathbf{s})}$, for i from 1 to n = len($\mathbf{s}$);

while *SL is not full* **do**

  $j \leftarrow \arg\max_{i \notin SL} \left(\lambda \mathbf{s}'[i] + (1-\lambda)\frac{1}{c}\sum_{j=1}^{c} \mathbf{d_j}'[C_{F_j}(\mathbf{x}^{(i)})]\right)$, breaks the tie by larger $\mathbf{s}'[i]$;

  $\mathbf{d}'[C_{F_j}(\mathbf{x}^{(j)})] \leftarrow \mathbf{d}'[C_{F_j}(\mathbf{x}^{(j)})] - \frac{1}{k}$, $j = 1, \ldots, c$;

  Put j into slate SL;

end while