Cross-Positional Attention for Debiasing Clicks

Honglei Zhuang, Zhen Qin, Xuanhui Wang, Michael Bendersky,
Xinyu Qian, Po Hu, Dan Chary Chen
Google
{hlz,zhenqin,xuanhui,bemike,xinyuqian,phu,lottie}@google.com

ABSTRACT

A well-known challenge in leveraging implicit user feedback like clicks to improve real-world search services and recommender systems is its inherent bias. Most existing click models are based on the examination hypothesis in user behaviors and differ in how to model such an examination bias. However, they are constrained by assuming a simple position-based bias or enforcing a sequential order in user examination behaviors. These assumptions are insufficient to capture complex real-world user behaviors and hardly generalize to modern user interfaces (UI) in web applications (e.g., results shown in a grid view). In this work, we propose a fully data-driven neural model for the examination bias, Cross-Positional Attention (XPA), which is more flexible in fitting complex user behaviors. Our model leverages the attention mechanism to effectively capture cross-positional interactions among displayed items and is applicable to arbitrary UIs. We employ XPA in a novel neural click model that can both predict clicks and estimate relevance. Our experiments on offline synthetic data sets show that XPA is robust among different click generation processes. We further apply XPA to a large-scale real-world recommender system, showing significantly better results than baselines in online A/B experiments that involve millions of users. This validates the necessity to model more complex user behaviors than those proposed in the literature.

CCS CONCEPTS

• **Information systems** → **Web searching and information discovery**; **Web mining**.

KEYWORDS

Examination bias; cross-positional attention; click models

ACM Reference Format:
Honglei Zhuang, Zhen Qin, Xuanhui Wang, Michael Bendersky, Xinyu Qian, Po Hu, and Dan Chary Chen. 2021. Cross-Positional Attention for Debiasing Clicks. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3442381.3450098

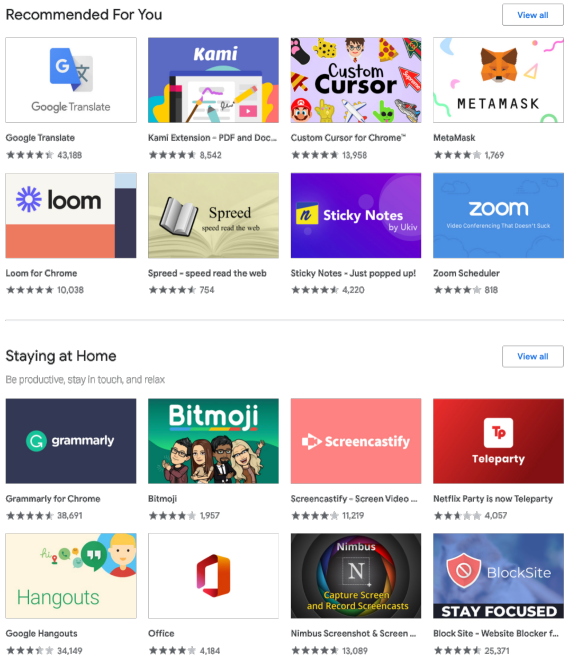


Figure 1: The recommendation UI layout of Google Chrome Web Store Homepage.

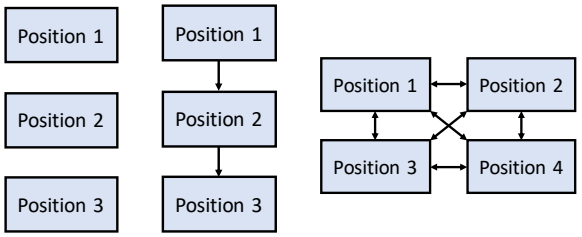
1 INTRODUCTION

Understanding user behaviors is key to improve effectiveness of many web applications, such as web search and e-commerce recommender systems. User clicks, as one of the most common user interactions in web applications, have attracted extensive interest in research and triggered a thread of studies on click modeling. Many click models have been proposed to explain or predict user clicks, enabling numerous applications such as user sessions simulation, improving ranking performance, and better evaluation metrics [8].

A common hypothesis in click models is the examination hypothesis. Users are usually assumed to examine displayed items before they click some of them, and there are often biases in the user examination behavior that are based on extra factors, e.g., positions on the user interface. Existing click models adopt various strategies in examination behavior modeling. Traditional click models based on the probabilistic graphical model (PGM) framework [8] often describe user examination behavior in a manually specified generative process based on some assumptions. For example, the popular Position-Bias Model (PBM) [33] assumes that the examination only depends on the position of the item without considering interaction between items (See Figure 2(a)). Another popular click model, Cascade Model (CM) [9], assumes that a user scans items in a list from top to bottom (See Figure 2(b)) until she finds a relevant item. Such assumptions leave the model little flexibility to accurately capture complex user behaviors.

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3450098



(a) Independent (b) Sequential (c) Fully-dependent

Figure 2: Typical user examination assumptions on items at different positions.

Recently, a few neural click models are proposed [3, 4, 7] to use distributed representations to describe user behaviors more flexibly than the restrictive assumptions made in PGM-based models. To the best of our knowledge, all existing neural click models make the implicit assumption that the results are shown in a sequence and use standard sequential neural models, including RNN [4], LSTM [3], and GRU [7], to model user examination bias explicitly [7] or implicitly [3]. This is inherited from the fact that click models were originally developed in web search, where a ranked list of results are presented to users.

However, modern web applications, such as recommendations on homepages, usually have a variety of UI layouts. For example, Figure 1 shows the layout of the homepage of Chrome Web Store, where there are multiple modules in a list, and extensions are organized in a grid view inside each module. Although there are some studies [42] of click models on grid view layouts, they are limited to sequential models and cannot be easily extended to more general layouts. Furthermore, recent studies [29, 43] show that models based on recurrent units (including RNN, GRU, and LSTM) are hard to train in practice and are usually not as effective as some alternatives, such as attention [36].

The drawbacks of existing click models not only hinder effective modeling of more flexible user behaviors, but also limit their usefulness in downstream applications. In this work, the major application we focus on is debiasing clicks to learn a good relevance model. This problem has been widely studied recently due to its practical value, as numerous real-world search and recommendation systems are using debiased models [18, 23, 40, 44]. How to model the bias is critical for the effectiveness of these methods. Nevertheless, most of them are generally built upon simple existing click models, typically the PBM model. For example, most unbiased learning to rank work [2, 21, 31, 40] treats the position bias as the propensity of observing relevance and debiases clicks using inverse propensity weighting (IPW) based methods. Other work [22] feeds the position of a document to the model but separates it from the rest of document features. Some recent works explore more complex models [19, 35], but are still limited to sequences. The performance and applicability of these approaches are constrained by the limitations of the underlying click models.

In this work, we propose Cross-Positional Attention (XPA) to model user examination bias, mitigating the limitations of existing click models. Our model assumes that the examination probability of an item not only depends on its own position, but could also depend on all other positions and the relevance of corresponding items (See Figure 2(c)). The assumption is flexible enough to capture any complex user behaviors on arbitrary layouts. Moreover, the strong capacity of attention mechanism allows our proposed model to produce more accurate prediction than sequential neural models like LSTM, even on simulated data sets where users are assumed to follow a sequential browsing behavior.

The contributions of our work are:

- We propose a novel click model using cross-positional attentions (XPA). It is capable of modeling more flexible user behaviors than existing models, and can be applied to any user interface layouts.
- We show that XPA is effective and robust among different data generation processes of user click behaviors via synthetic experiments.
- We apply XPA to debias clicks in a real-world large-scale recommender system. Online A/B experiments involving millions of users validate the effectiveness of XPA and the necessity to model more flexible user behaviors and user interface layouts.

The rest of the paper is organized as follows. We review related work in Section 2. We describe preliminaries of click models in Section 3 and the proposed XPA model in Section 4. In Section 5, we validate the effectiveness of XPA on both offline synthetic data sets and online A/B experiments on a large-scale recommender system. We conclude this paper in Section 6.

2 RELATED WORK

To understand user behaviors and estimate the relevance of items, click models have been extensively studied, mainly for web search [8]. Most of the existing click models in the probabilistic graphical model (PGM) framework represent user behaviors as a sequence of observable or hidden random variables. Based on the classical Position-Bias Model [33] and Cascade Model [9], many click models are proposed as their extensions. Models including User Browsing Model (UBM) [13], Dynamic Bayesian Network (DBN) [6], Click Chain Model (CCM) [15], and Dependent Click Model (DCM) [16] have been proposed to cope with the limitation that CM only allows one click. Partially Sequential Click Model (PSCM) [37] is one of the few work that considers non-sequential examination behavior motivated by eye-tracking experiments, but the non-sequential behavior is still studied on sequential data (i.e., users can browse back and forth on a sequence). As various layouts in web applications emerge in recent years, there are also models specifically developed for each type of layout. For example, a click model [25] is developed for vertical results with different item types on mobile search to take into account vertical bias. Another click model [41, 42] is proposed for grid view layout in image search, still based on a sequential user browsing assumption. All dependencies in the above PGM-based click models are designed manually, which are likely to miss key aspects of complex and flexible real-world user behaviors. Diaz et al. [11] study mouse movement modeling in arbitrary layouts, which could potentially be used to model user examination behavior. However, their model cannot be used for relevance prediction.

Recently, some neural network based approaches have been proposed for better user modeling. User behaviors are often represented as vector states instead of binary random variables to better describe complex behavioral patterns. Borisov et al. [3] first attempt to use neural networks to model users' query level interaction sequence using an LSTM model. The Click Sequence Model (CSM) [4] maintains an encoder-decoder framework with an RNN backbone to predict the order in which a user will interact with search engine results. The Context-Aware Click Model (CACM) [7] leverages GRU to model behaviors among search sessions. Most existing neural click models are still constrained by the underlying recurrent units, such as LSTM, with the implicit assumption of sequential browsing behaviors, which can be limited in practice. Also, inherited from traditional click models, existing neural click models typically require exact query and document identifier for encoding or retrieving certain information. This can be very ineffective in practice especially for recommender systems where the interactions for a specific user-document pair is extremely sparse.

An important application of click models is to tackle the inherent bias in user click data when training a relevance model from click logs. This is an important research topic in search and recommendation recently [1, 2, 14, 21, 31, 35, 38, 40]. Unbiased learning works are often built upon existing click models, as many click models are based on the examination hypothesis to cope with the position bias problem [33]. However, virtually all existing work depends on PBM [1, 2, 21, 31, 38–40], with a very recent exception that explores Cascade Model [35]. Thus, the performance of unbiased learning is hindered by the underlying limitations of the click models mentioned above. Our model is much more flexible than PBM in capturing complex user behaviors and can be seamlessly plugged in many existing unbiased learning frameworks to achieve better performance. We leverage the attention [36] mechanism to effectively model flexible interactions. Attention has been used in some recent learning to rank work [28, 30, 32], but its application to debiasing implicit feedback has not been explored.

It is not straightforward to directly incorporate existing click models into unbiased learning models. A common reason is because traditional click models developed for web search often use the exact identifiers of queries and documents to encode or retrieve certain information. Some recent neural click models often inherit this setting. For example, Borisov et al. [3] estimate the relevance of a query-document pair using the click ratio (number of clicks divided by number of impressions) of the exact query-document pair from search logs. Borisov et al. [4] represent a query using a 2^N (N is the number of positions) dimensional vector, where each entry counts the number of times a click pattern was observed in query sessions for the exactly same query. Such a counting method is not applicable for unseen query-document pairs, and can be highly unreliable when clicks are extremely sparse as in many real-world web applications. Hence, we use feature vectors instead of identifiers for representations in our relevance component.

A study [27] also explores simultaneously modeling complex user behavior and relevance ranking. It is based on reinforcement learning in the online setting, while our focus is on learning relevance models from offline logs.

3 PRELIMINARIES

In this section, we give the formal description of a click model, the commonly used examination hypothesis and our problem formulation.

3.1 Click Model

We define a session s_i as a set of items along with their positions and clicks $s_i = (\mathbf{X}_i, \mathbf{P}_i, \mathbf{c}_i)$. $\mathbf{X}_i = \{\mathbf{x}_{ij}\}_{j=1}^m$ represents the set of items displayed in the i-th session, where each item is represented by a feature vector $\mathbf{x}_{ij} \in \mathbb{R}^v$. $\mathbf{P}_i = \{p_{ij}\}_{j=1}^m$ is a set of integer indices representing the positions of the displayed items in the i-th session, where p_{ij} is a unique identifier of item j's position in the i-th session. For generality, we do not assume any relative proximity between positions based on their values (i.e., position 2 is not necessarily closer to position 1 than position 20), although some sequential models may rely on such an assumption. $\mathbf{c}_i \in \{0,1\}^m$ is a binary vector of length m where the j-th element c_{ij} indicates whether item j is clicked in this session. Notice that we do not aim to model the order of user clicks, hence the order of clicks is not included in our formalization.

A general click model can be trained from a set of collected sessions $\mathbf{S}_{\text{Train}} = \{s_i\}_{i=1}^n$ with user click logs and learn to estimate the conditional probability of clicks $\mathbb{P}(\mathbf{c}|\mathbf{X}, \mathbf{P})$ for any session (without collected clicks) given the items $\mathbf{X}$ and their positions $\mathbf{P}$.

3.2 Relevance-Examination Factorization

Many click models [10, 13, 15, 16, 34] follow the examination hypothesis [9] where the click probability of an item j can be factorized into two elements: the tendency of users examining the item, and the relevance (or attractiveness) of the item [8]. The relevance of the item is usually thought to be position-agnostic and is only correlated with the item's own features $\mathbf{x}_j$, whereas the examination factor of an item would be correlated with the item's position p_j and sometimes with other items' relevance and positions in the session. In fact, the examination factor is related to user browsing behaviors and is generally regarded as a bias when one uses clicks as relevance labels to train a relevance model.

We formalize this factorization by modeling the conditional click probability of the item j in a session $s = (\mathbf{X}, \mathbf{P}, \mathbf{c})$ as:

$$\mathbb{P}(c_j|\mathbf{X}, \mathbf{P}) = f\Big(g(r(\mathbf{x}_j)) + h(e(\mathbf{x}_j, p_j, \mathbf{X}, \mathbf{P}))\Big) \tag{1}$$

where $r(\mathbf{x}_j)$ is the relevance score of item j and $e(\mathbf{x}_j, p_j, \mathbf{X}, \mathbf{P})$ is the examination score of item j in this session. $f(\cdot)$, $g(\cdot)$ and $h(\cdot)$ are univariate transformation functions. In this paper, we also use a short-hand notation $\hat{c}_j$ to represent the predictive probability $\mathbb{P}(c_j|\mathbf{X}, \mathbf{P})$ estimated by a click model.

A click model with a factorization in Equation (1) is more useful in many applications. For example, one important application is to mitigate the position bias. Some positions on a web page (usually those at the top) have larger probabilities of being examined by users. Hence, items that are more often displayed in these positions will be clicked more. If we directly train a ranking model based on click logs, the model would be affected by position bias. Instead, one can train a click model with the above factorization and

only leverage the relevance module $r(\cdot)$ to estimate the position-agnostic relevance of items and rank accordingly, which is popular in industrial applications [17, 44].

Connection to existing click models. In a probabilistic click model where each item can only be examined once, $r(\cdot)$ and $e(\cdot)$ are usually defined as the probability of item j being relevant $\mathbb{P}(R_j = 1|\mathbf{x}_j)$ and the probability of item j being examined $\mathbb{P}(E_j = 1|\mathbf{X}, \mathbf{P})$ respectively. By setting $f(x) = \exp(x)$ and $g(x) = h(x) = \log(x)$ we can obtain that

$$\mathbb{P}(c_j|\mathbf{X}, \mathbf{P}) = \mathbb{P}(R_j = 1|\mathbf{x}_j)\mathbb{P}(E_j = 1|\mathbf{X}, \mathbf{P})$$

which is a common assumption in existing click models. Here $R_j = 1$ indicates the event that a user finds item j relevant, and $E_j = 1$ indicates the event that a user examines item j.

3.3 Problem Formulation

Our aim is to develop a click model which can be trained merely with click logs but can be used for both predicting clicks from a given item layout and predicting item relevance without the bias in the click data. We focus on click models with the relevance-examination factorization as described above, which provides a model structure abstraction that can naturally fulfill these two tasks. We formalize our problem as:

Problem 1. *Given a set of sessions with click logs* $\mathbf{S}_{Train} = \{s_i\}_{i=1}^{n}$, *we aim to train a click model with the structure as Equation* (1), *s.t.*

(1) *Given a set of items with their feature vectors* $\mathbf{X}$ *and their positions* $\mathbf{P}$, *the model can estimate the probability* $\hat{c}_j = \mathbb{P}(c_j|\mathbf{X}, \mathbf{P})$ *of whether a user would click each item* $\mathbf{x}_j \in \mathbf{X}$;
(2) *In addition, the relevance scorer* $r(\cdot)$ *of the model can be estimated and used to rank any set of items based on their feature vectors* $\mathbf{X}$.

We note that some other studies [22] on similar problem do not adopt the factorization of relevance and examination in their models, but instead feed a constant position index for the relevance prediction/ranking task during inference. We do not adopt this methodology due to the black-box nature of the model. It is difficult to verify or fully understand the model behavior during inference, and it is hence not favored for real-world applications.

4 CROSS-POSITIONAL ATTENTION

In this section, we introduce a neural click model with a cross-positional attention mechanism to model interactions between different positions in each session.

We adopt the relevance-examination factorization of clicks to learn a relevance scorer $r(\cdot)$ and an examination scorer $e(\cdot)$ respectively and obtain the estimated click probability according to Equation (1). We assume that the examination score of item j could be correlated with *all* items' features $\mathbf{X}$ and positions $\mathbf{P}$ in the same session. Notice that this is a more general assumption compared with existing models where the examination score of item j is only related to its position p_j or is related to "previous items" in a given session. We argue that our click model can better handle various complicated real-world scenarios where the layout of positions is not necessarily a sequence and/or users do not necessarily examine in order. Figure 3 provides a graphical illustration of our proposed model. We describe the details below.

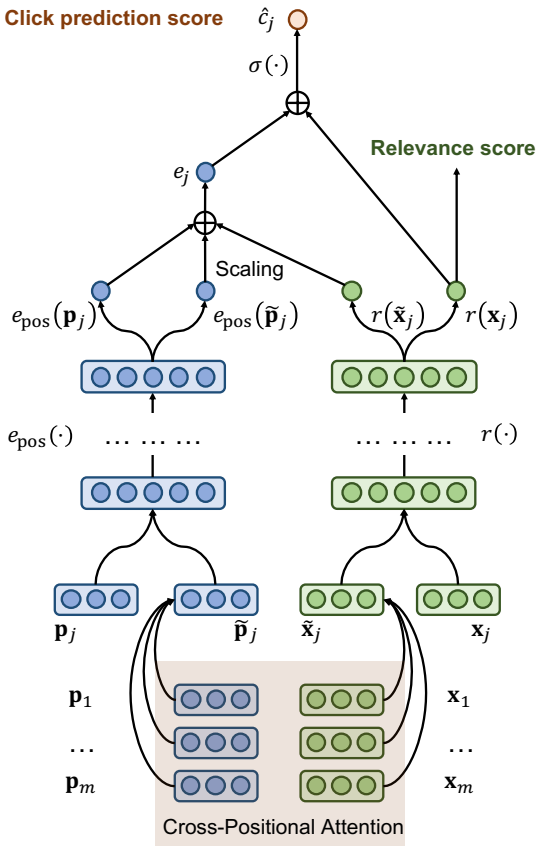


Figure 3: Graphical illustration of our proposed model.

4.1 Input Representation

The input of the model consists of the feature vectors of all items $\mathbf{X}$ in the session as well as their positions $\mathbf{P}$. The j-th item can directly be represented by its feature vector $\mathbf{x}_j$. Its position p_j, which is usually an index, can be represented by a corresponding embedding vector $\mathbf{p}_j$. The position embedding can be randomly initialized and then trained end-to-end with the model.

Notice that an alternative is to adopt a fixed sinuous embedding to instantiate position embedding, as reported in [36]. However, we do not find substantial improvement from such an instantiation in our experiments.

4.2 Relevance Scorer

The relevance scorer $r(\cdot)$ (See the right module in Figure 3) only takes the feature vector of a single item $\mathbf{x}_j$ as input. We simply instantiate the relevance scorer by an H-layer feed-forward network:

$$r(\mathbf{x}_j) = \text{FFNN}(\mathbf{x}_j|\mathbf{W}_{1,\dots,H}, \mathbf{b}_{1,\dots,H}) \tag{2}$$

where each layer is a fully-connected dense layer parameterized by $\mathbf{W}_h$ and $\mathbf{b}_h$, and the activation function is the Rectifier [26].

4.3 Cross-Positional Attention

In a real-world scenario, whether a user examines a specific item can be related to many factors: the position of this item, relevance of neighboring items, etc. Based on this intuition, we introduce a cross-positional attention layer to leverage signals from all items at all positions in the same session.

Formally, we derive a cross-positional attention matrix $\mathbf{A}$ where the element at (j, k) is derived by a scaled softmax:

$$a_{jk} = \frac{\exp(\lambda \mathbf{p}_j^\top \mathbf{p}_k)}{\sum_{k'} \exp(\lambda \mathbf{p}_j^\top \mathbf{p}_{k'})} \tag{3}$$

where λ is a trainable scalar variable, not a hyper-parameter [24]. The attention weight a_{jk} reflects the proximity between position p_j and position p_k and measures the influence and interactions between examination or click behaviors between these positions. Typically, positions that are visually close to each other would have higher attention weights.

For the item at position p_j, we derive both the attended item representation $\tilde{\mathbf{x}}_j$ and the attended position representation $\tilde{\mathbf{p}}_j$ by taking the attention weighted sum of item/position representation from other positions:

$$\tilde{\mathbf{x}}_j = \sum_{k=1}^{m} a_{jk} \mathbf{x}_k, \qquad \tilde{\mathbf{p}}_j = \sum_{k=1}^{m} a_{jk} \mathbf{p}_k$$

These attended representations aggregate signals from other positions weighted by how close they are to position p_j. They reflect how likely a neighboring item is relevant or will be examined. These signals will be used in the examination scorer of item j to model the correlation of user behaviors between these positions.

4.4 Examination Scorer

We first introduce a position-based examination scorer $e_{\text{pos}}(\cdot)$ instantiated by a feed-forward network (See the left module in Figure 3). For each item j, the scorer only takes its position embedding $\mathbf{p}_j$ as input to provide an examination score. The examination scorer can be written as:

$$e_{\text{pos}}(\mathbf{p}_j) = \text{FFNN}\big(\mathbf{p}_j | \mathbf{W}'_{1,\dots,H}, \mathbf{b}'_{1,\dots,H}\big) \tag{4}$$

Similarly, each layer is a fully-connected dense layer parameterized by $\mathbf{W}'_h$ and $\mathbf{b}'_h$, and the activation function is the Rectifier [26].

The examination score $e_{\text{pos}}(\mathbf{p}_j)$ only utilizes the signal from the position of item j. However, signals from other items and their positions can also have an impact on users' examination behavior on item j. For example, when there is a highly relevant item at a neighboring position of item j, users may be more likely to examine item j as they are attracted by the relevant item.

We can incorporate these signals by utilizing the attended item representation $\tilde{\mathbf{x}}_j$ and attended position representation $\tilde{\mathbf{p}}_j$ as they aggregate other items and their positions weighted by the cross-positional attention. We feed the attended position representation to the position-based examination scorer to depict to what extent neighboring positions will be examined, denoted as $e_{\text{pos}}(\tilde{\mathbf{p}}_j)$. We also feed the attended item representation to the relevance scorer to estimate how likely neighboring items will be clicked, denoted as $r(\tilde{\mathbf{x}}_j)$. Then we combine both scores with the original position-based examination scores to obtain the cross-positional attention-based examination score $e_{\text{cross-pos}}(\mathbf{p}_j, \tilde{\mathbf{x}}_j, \tilde{\mathbf{p}}_j)$, defined as:

$$e_{\text{cross-pos}}(\mathbf{p}_j, \tilde{\mathbf{x}}_j, \tilde{\mathbf{p}}_j) = e_{\text{pos}}(\mathbf{p}_j) + w_e \cdot e_{\text{pos}}(\tilde{\mathbf{p}}_j) + w_r \cdot r(\tilde{\mathbf{x}}_j) \tag{5}$$

where w_e and w_r are *not* hyper-parameters, but two weighting variables to be automatically trained.

Discussion. An alternative option is to apply the cross-positional attention mechanism to relevance scores of all other items $r(\mathbf{x}_k)$ instead of their original feature representation $\mathbf{x}_k$. We tried this design but failed to achieve good performance. A possible explanation is that the gradient to adjust the relevance scorer network $r(\cdot)$ would be distributed based on attention values, which may not be well-trained in the early stage of training.

Another possible design is to adopt a more complicated attention mechanism, such as the multi-head attention [36] and/or multiple layers of stacked attentions. However, none of these complicated mechanisms show significantly better results than the simple attention mechanism presented above. This could mean lack of long-range dependency in users' examination behaviors.

4.5 Click Probability

For item j, we can combine the examination and relevance scores to obtain the click probability. We follow the factorization in Equation (1) and set the univariate transformation $f(\cdot)$ to the sigmoid logistic function $\sigma(x) = \frac{\exp(x)}{1+\exp(x)}$, which ensures the output to be a probability in $[0, 1]$. The other two univariate transformation functions $g(\cdot)$ and $h(\cdot)$ are simply set to identity for simplicity. More formally, we have

$$\hat{c}_j = \sigma\big(r(\mathbf{x}_j) + e_{\text{cross-pos}}(\mathbf{p}_j, \tilde{\mathbf{x}}_j, \tilde{\mathbf{p}}_j)\big) \tag{6}$$

Notice that if we only take the position-based examination scorer $e_{\text{pos}}(\cdot)$ and plug it into the same equation, we will have a model merely based on position bias that is similar to the models in existing papers [17, 44].

4.6 Training

We train the model with cross-entropy loss applying to the marginal click probability of each item. For a training set $\mathbf{S}_{\text{Train}}$ of n sessions with clicks collected, we can optimize the following loss function:

$$L = -\sum_{i=1}^{n} \sum_{j=1}^{m} \Big[c_{ij} \log(\hat{c}_{ij}) + (1 - c_{ij}) \log(1 - \hat{c}_{ij}) \Big] \tag{7}$$

Remarks. If the desired relevance prediction objective is to achieve the best *ranking* performances, there are plenty of studies [2, 19, 21, 40] on unbiased learning to rank, which usually train the model with an unbiased ranking loss function instead of a cross-entropy loss. It is worth noting that many of these studies adopt a model structure with a similar relevance-examination factorization. Hence, one can seamlessly plug in our examination scorer with cross-positional attention into the unbiased learning to rank framework and train the model with ranking losses.

5 EXPERIMENTS

In this section, we present the experimental results to evaluate the effectiveness of XPA on both offline synthetic experiments and a real-world recommender system with millions of users.

5.1 Data Sets

We conduct experiments on two data sets: one with synthetic click data and another with real click data.

YAHOO. Yahoo! Learning to Rank Challenge data [5] is a public learning-to-rank data set. We use the Set1 of the original data set, which contains 19,944 queries in the training partition, 2,994 and 6,983 queries in the validation and test partition respectively. There are on average 20+ documents in each query. Each document is represented by a numerical feature vectors with up to 700 features with a normalized range of $[0, 1]$.

For each query, documents are evaluated by human annotators and labeled with a 5-level relevance label from 0 to 4 where 4 is the most relevant and 0 is irrelevant. However, user click data is not provided in this data set. Therefore, we generate synthetic sessions with clicks based on various models, similar to many work that studies unbiased learning [2, 35]. More details are provided below.

Chrome Web Store (CWS). We also perform experiments on a data set collected from Chrome Web Store[1] user logs[2]. Each session logs a set of extensions displayed to the user, their positions in the layout, and the clicked extensions. An example of CWS homepage layout is shown in Figure 1, where 8 extensions are organized in a grid view as a module and multiple modules are presented vertically. We extract several numerical and categorical features for each extension, such as number of impressions in the last two weeks. The same set of features are used for all compared methods. For each categorical feature, we learn an embedding vector. We concatenate all the embedding vectors with the numerical feature vector as the representation of the item. We collect 20 days of sampled logs as training data and the following 9 days of sampled logs as test data.

5.2 Click Simulation

For YAHOO data set with no user click data available, we generate synthetic sessions with clicks for training and evaluation. Similar to previous studies [2, 21], we first train a Ranking SVM [20] as the initial ranker based on 1% of the labeled training data. We then use the initial ranker to rank all the items in each query and use the ranking starting from 1 as the position feature p_j for each item j. Based on the initial ranking, we simulate a user to generate click sessions based on the following models.

Position bias model (PBM). In this model [34], users are assumed to examine each item independently with a probability only related to its position. We define the examination probability as

$$\mathbb{P}(E_j = 1|p_j) = \frac{1}{p_j}$$

If an item is examined, users will determine its relevance based on the ground-truth relevance label (denoted as y_j) but with certain noise. We define the probability that users consider item j relevant as

$$\mathbb{P}(R_j = 1|y_j) = \epsilon + (1 - \epsilon)\frac{2^{y_j} - 1}{2^{y_{\max}} - 1} \tag{8}$$

where $\epsilon = 0.1$ is a noise constant and $y_{\max} = 4$ is the maximum possible label. Users will click an item once they examine the item and find it relevant. Hence, we can calculate the probability of an item being clicked by

$$\mathbb{P}(c_j = 1|p_j, y_j) = \mathbb{P}(E_j = 1|p_j) \cdot \mathbb{P}(R_j = 1|y_j)$$

Dependent click model (DCM). The dependent click model [16] is adapted from the popular cascade model [10] which assumes that a user would examine the items based on the displayed sequence one by one. Once the user finds a relevant item, the user would click the item. With a probability $\mathbb{P}(E_{j+1} = 1|c_j = 1)$, the user would continue to examine the following item. Otherwise the users would end the session. We simulate the user behavior by going through the items based on the initial ranking. We assume that a user finds an item to be relevant with the probability as defined in Equation (8). We set the probability to resume the session after a click $\mathbb{P}(E_{j+1} = 1|c_j = 1)$ as a constant 0.1.

Click propagation model (CPM). We also propose a novel click model to simulate users who do not follow the sequential browsing behavior. Generally, we still assume that a user follows PBM to browse, examine and click items. However, if the item j is clicked, the user will start a "subprocess" to examine other items k based on the following probability:

$$\mathbb{P}(E_{k|j} = 1|p_j, p_k) = \frac{1}{|p_k - p_j|}$$

where $E_{k|j}$ represents the event that the user examines item k after clicking item j. If the user finds any item relevant (based on Equation (8)) during this subprocess, she will also click the item (but not triggering another round of examination). When the subprocess is over, the user would go back to continue the PBM behavior. An item j clicked by multiple times is still logged as $c_j = 1$.

The model is similar to user browsing model [13] but the user behavior does not follow the sequential examination process.

5.3 Experiment Setup

Comparing methods. We compare the performance of the following methods:

- *No-Position (NoPos).* A naïve baseline is to directly train the relevance scorer $r(\cdot)$ as described in Equation (2) based on user clicks without using any position information. Since user click behavior is usually related to item position, the learned model is likely to be biased.
- *Position-Based Model (Pos).* A model structure similar to [17, 44], where the position-based examination scorer $e_{\text{pos}}(\cdot)$ is adopted to score item j only based on its own position p_j. The examination score and the relevance score are then added and fed into a sigmoid logistic function to obtain the click probability estimation.

[1] https://chrome.google.com/webstore

[2] Data collected and used is covered by the Google Privacy Policy and is used for product improvements for Chrome Web Store.

- *LSTM-Based Sequential Model (LSTM).* A recent model [19] feeds a sequence of item feature vectors into LSTM and derives examination scores based on LSTM output at each position. We implement a variation following our factorization of relevance and examination. The examination scorer is instantiated by a feed-forward network built on the LSTM output vectors. Unlike the original model, this variation supports multiple clicks in a session. For the YAHOO data set, we construct the sequence based on initial ranking; for the CWS data set, we construct the sequence by a left-to-right order within each row, and a top-down order for all the rows. Notice that due to different configurations of click simulation, the performances are not comparable to those reported in [19].
- *Cross-Positional Attention Model (XPA).* This is our proposed model in Section 4 which utilizes a cross-positional attention mechanism to leverage signals from all items and positions.

Evaluation methodology. We train all the models with click data and evaluate the model performances on two tasks: click prediction and item relevance prediction.

For the click prediction task, we apply each model on the hold-out test partition of the data set to predict which items would be clicked based on their features and positions. We measure the model performance by both loglikelihood and perplexity.

For the item relevance prediction task, we used the relevance scorer $r(\cdot)$ of each model to predict the relevance of each item from the hold-out test set based on its feature vector $\mathbf{x}_j$. For the YAHOO data set, we evaluate the performance by NDCG@k where $k \in \{1, 5, 10\}$ based on the human judged relevance labels. For the CWS data set, we perform online A/B experiments by serving each model to a small percentage of randomly selected users. We use Click-Through Rate (CTR) as our online evaluation metric.

Parameter configuration. For the feed-forward network in the relevance scorer (Equation (2)), we use a 3-layer network with hidden layer dimensions as 1024, 512 and 256. For the feed-forward network in the examination scorer (Equation (4)), we use a 3-layer network with hidden layer dimensions as 128, 64 and 32. All item features are transformed by a log1p [45] transformation. We apply batch normalization and dropout to both feed-forward networks where the dropout rate is set to 0.5. We set the dimension of embedding vectors for position features and other categorical features to 100. All models are trained with the AdaGrad [12] optimizer with learning rate tuned on the validation set.

5.4 Results on Simulated Click Data

We first evaluate all the methods on the YAHOO data set where the click data is simulated.

Click prediction. Table 1 shows the performance comparison of the click prediction tasks. For all simulated click models on the YAHOO data set, both LSTM and XPA show better performance than NoPos and Pos. This aligns with our expectation as both methods take the correlation between clicks on different positions into account.

We also notice that LSTM achieves comparable performance with XPA when clicks are simulated by PBM and DCM. This is because the sequential model structure of LSTM is capable of capturing the examination patterns in these two models, where users either examine each position independently or browse in a sequence. However, when the simulated user browsing behavior does not follow the sequential order like in CPM, our proposed XPA achieves significantly better performance than LSTM and other baselines. In real-world applications, especially when the layout is not a ranked list, there is no guarantee that users would strictly follow a specific sequential order to browse different positions. Therefore, LSTM may not achieve satisfying performances on such real-world data sets.

Relevance prediction. A more practically useful task is to utilize the relevance scorer module of the click model to estimate the relevance of each item. Since the ground-truth relevance labels are

Table 1: Click prediction performance comparison on the YAHOO data set. Results that are statistically significantly better ($\alpha = 0.05$) than *Pos* are marked with an asterisk (*) and those significantly better than *LSTM* are marked with a dagger (†). The best results are bolded.

Click model	Method	Loglikelihood	Perplexity
PBM	NoPos	-0.1365	1.1462
	Pos	-0.1246	1.1327
	LSTM	**-0.1219***	**1.1297***
	XPA	-0.1227*	1.1305*
DCM	NoPos	-0.1353	1.1449
	Pos	-0.1218	1.1295
	LSTM	-0.1198*	1.1273*
	XPA	**-0.1195***	**1.1269***
CPM	NoPos	-0.2267	1.2545
	Pos	-0.2184	1.2441
	LSTM	-0.2153*	1.2373*
	XPA	**-0.2117***†	**1.2358***†

Table 2: Relevance prediction performance comparison on the YAHOO data set (%). Results that are statistically significantly better ($\alpha = 0.05$) than *Pos* are marked with an asterisk (*) and those significantly better than *LSTM* are marked with a dagger (†). The best results are bolded.

Click model	Method	NDCG@1	NDCG@5	NDCG@10
PBM	NoPos	63.75	66.75	71.49
	Pos	66.34	69.13	73.62
	LSTM	**67.03**	68.63	73.14
	XPA	66.81	**69.57***†	**73.85**†
DCM	NoPos	64.05	66.45	71.11
	Pos	65.50	67.60	72.13
	LSTM	65.15	67.29	71.86
	XPA	**67.08***†	**68.68***†	**73.07***†
CPM	NoPos	64.67	67.78	72.50
	Pos	67.82	70.26	74.54
	LSTM	67.08	68.82	73.13
	XPA	**67.95**†	**70.61***†	**74.86***†

Table 3: Click prediction performance comparison. Results that are statistically significantly better ($\alpha = 0.05$) than *Pos* are marked with an asterisk (*) and those significantly better than *LSTM* are marked with a dagger (†). The best results are bolded.

Data set	Method	Loglikelihood	Perplexity
CWS	NoPos	-0.1044	1.1100
	Pos	-0.1019	1.1072
	LSTM	-0.1004*	1.1057*
	XPA	**-0.0978***†	**1.1027***†

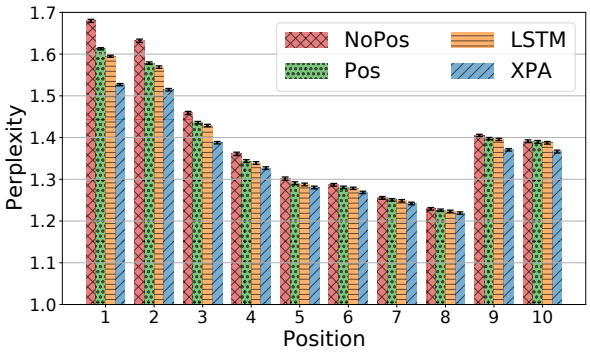


Figure 4: Model perplexity comparison at different positions on the CWS data set.

provided in YAHOO, we can rank the items in the test data set based on the relevance scorer of each model and evaluate the ranking performance. Table 2 shows the results on YAHOO data set with different click simulation models.

Pos, LSTM and XPA have similar performance with simulated clicks from PBM. The performance of XPA and LSTM is not significantly better than Pos, since they are better at modeling cross-positional correlation of clicks, while the examination probability of an item only depends on its own position in PBM.

When the simulated clicks are generated from DCM or CPM, XPA significantly outperforms all the other baselines including Pos and LSTM in terms of NDCG@5 and NDCG@10. The results illustrate the strength of XPA in the application of debiasing clicks when the user examination behaviors are complex with cross-positional correlation. An additional advantage of XPA is that the relevance component $r(\cdot)$ of XPA can be trained with additional signals as it is also used in modeling user examination behaviors. This might explain why XPA outperforms LSTM even with the sequential click simulation model DCM. On the other hand, LSTM again fails to achieve better performance with CPM, showing that complex user behaviors which do not strictly follow the sequential assumption can cause severe bias in relevance prediction.

Overall, XPA achieves comparable or significantly better performance than baselines with all the three click simulation models. The results verify the generalizability and robustness of XPA which can be universally applied to data with different user click patterns without modification.

Table 4: Online experiments on the CWS data set. Relative increase in Click Through Rate compared with *NoPos* is shown. Results that are statistically significantly better ($\alpha = 0.05$) than *NoPos* are marked with an asterisk (*) and those significantly better than *Pos* are marked with a dagger (†). The best result is bolded.

Method	NoPos	Pos	XPA
Click Through Rate	0.00%	5.58%*	**11.80%***†

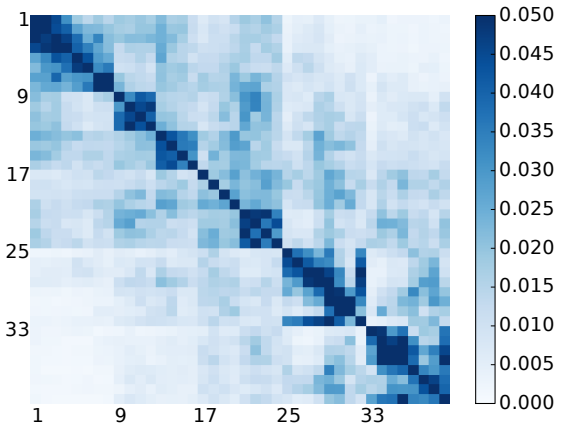


Figure 5: Visualizing the cross-positional attention matrix A learned from the CWS data set.

5.5 Results on Real-World Data

We move forward to compare the model performances on the CWS data set, which contains real user click logs reflecting the more complicated real-world user behavior.

Click prediction. We first measure the predictive power of each model on the click data from the hold-out test partition of the CWS data set. Table 3 summarizes the performance. As one can observe, XPA significantly outperforms all the other baselines on both loglikelihood and perplexity. This result shows that the cross-positional attention mechanism provides larger model capacity to better model complex user behaviors in real-world.

We further slice the CWS data based on positions and plot the perplexity of each model. Figure 4 plots the results as well as the 95% confidence intervals. We notice that XPA outperforms the other two baselines on almost all positions significantly. The gap is particularly large for specific positions in each module like position 1, 2 and 9, 10. We also notice that these positions have larger overall perplexity from all models, indicating that they are more difficult for click prediction.

Online relevance prediction. We cannot run offline evaluation of relevance prediction on the CWS data set as there is no human judged relevance labels. Instead, we deploy the compared models in online A/B experiments. We used the relevance scorer to rank a candidate set of extensions and serve the top results on "Recommended for You" module on CWS homepage (See Figure 1). We only

perform experiments with Pos and XPA due to limited resources — although LSTM achieves better click prediction performance on the CWS data set, we observe that the advantage in the click prediction task does not necessarily translate to the relevance prediction task, where LSTM suffers in general (Table 2). The experiments were run for 7 days and the performances are measured by CTR. Due to the proprietary nature of the data, we only report relative improvement. The results are shown in Table 4.

It can be observed that both Pos and XPA perform significantly better than NoPos. This shows the importance of removing positional bias when training relevance models on click logs. More importantly, XPA achieves a 11.80% improvement relative to NoPos, which is significantly higher than the performance of Pos. The results show that XPA can more effectively model and mitigate the bias in click logs introduced by complex user behavior.

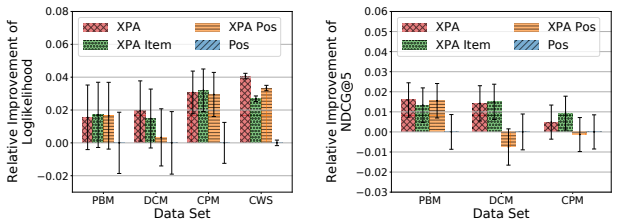
Visualization of cross-positional attention. We visualize the cross-positional attention matrix $\mathbf{A}$ from the XPA model trained for the CWS data set, as illustrated in Figure 5. We only show the sub-matrix for the top-40 positions. It can be clearly observed that there are dense interactions formed between every 4 or 8 positions. For example, there are very dense interactions between position 1 to 8, and then dense interactions between 9 to 12 etc. An explanation is that users tend to click items within the same module after they click a specific item. The pattern aligns with the current CWS homepage layout where every module shows 8 extensions on the homepage. The connection between position 9-12 and position 13-16 is not dense probably due to the screen size constraint which does not show the entire module on the homepage. It is also worth noting that this pattern repeats itself with almost equivalent strength even in positions in the bottom side of the homepage. For example, similar dense interaction block also appears between position 33 to 40. This shows the significance of this user behavior pattern and its possible bias impact on the click data.

5.6 Ablation Study

We also perform an ablation study of our proposed model on both synthetic and real-world data sets. We run experiments with two variations of our model: one only using the attended position representation (XPA Pos) and the other only using the attended item representation (XPA Item). We then compare their performances as well as the full XPA model performance with the Pos baseline. Relative improvements on the click prediction task (measured by log-likelihood) and the relevance prediction task (measured by NDCG@5) are reported in Figure 6.

For the click prediction task, as illustrated in Figure 6(a), both the XPA Pos and XPA Item variations significantly outperform the Pos baseline on the CWS data set. This shows that both attended representations improve the performance. The full XPA model further outperforms the two variations on the CWS data set, which shows the power of simultaneously leveraging both signals in the model. On the synthetic data set, we can also observe that both variations seem to improve the performance.

Figure 6(b) plots the results of the relevance prediction task. The XPA Item variation which only uses the attended item representation always outperforms the Pos baseline, whereas the XPA Pos



(a) Click prediction (b) Relevance prediction

Figure 6: Ablation study with XPA variations that only use the attended position representation (XPA Pos) or the attended item representation (XPA Item) respectively. Performances on both click prediction and relevance prediction tasks are plotted. PBM, DCM and CPM stand for the click simulation models on the YAHOO data set.

variation does not. Given that XPA Pos still achieves better performance in the click prediction task, the results might suggest that the relevance scorer could be over-corrected by the attended positional representation in the current model for ranking tasks. As mentioned earlier, more sophisticated unbiased learning to rank models can be further developed on top of our click model to optimize relevance ranking performance.

6 CONCLUSION

In this paper, we propose a cross-positional attention mechanism to model user examination bias in clicks. It considers the correlation between clicks on a specific item and other items in the same session, including their relevance and positions. The model does not make restrictive assumptions about UIs or user behaviors and is thus capable of capturing a variety of user behavior patterns with arbitrary layouts. In addition, the relevance scorer component can be directly used to estimate the relevance of any item. In our experiments, we show that our model can better mitigate the positional and cross-positional bias in click logs and learn a more effective relevance scorer on multiple synthetic data sets and a real-world recommendation data set.

There are several interesting future research directions based on our work. First, in addition to position signals, there could be other ones affecting user click behaviors. For example, the date and time information and users' browsing history [7] are some other possible contributing factors. How to incorporate such signals and mitigate the bias they introduce to relevance prediction would be an interesting direction to explore. Second, how to design a more efficient attention mechanism is also worth studying. The current attention mechanism models the correlation between every pair of items, which is proportional to m^2 where m is the number of items in each session. If we have more information about the layout and relative relations between positions, it is possible to simplify the attention layer for better performance. Third, a natural extension is to study how to leverage the proposed click model to build a better unbiased learning to rank model.

REFERENCES

[1] Aman Agarwal, Ivan Zaitsev, Xuanhui Wang, Cheng Li, Marc Najork, and Thorsten Joachims. 2019. Estimating position bias without intrusive interventions. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 474–482.

[2] Qingyao Ai, Keping Bi, Cheng Luo, Jiafeng Guo, and W Bruce Croft. 2018. Unbiased learning to rank with unbiased propensity estimation. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*. 385–394.

[3] Alexey Borisov, Ilya Markov, Maarten de Rijke, and Pavel Serdyukov. 2016. A Neural Click Model for Web Search. In *Proceedings of the 25th International Conference on World Wide Web*. 531–541.

[4] Alexey Borisov, Martijn Wardenaar, Ilya Markov, and Maarten de Rijke. 2018. A Click Sequence Model for Web Search. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*. 45–54.

[5] Olivier Chapelle and Yi Chang. 2011. Yahoo! learning to rank challenge overview. In *Proceedings of the Learning to Rank Challenge*.

[6] Olivier Chapelle and Ya Zhang. 2009. A Dynamic Bayesian Network Click Model for Web Search Ranking. In *Proceedings of the 18th International Conference on World Wide Web*. 1–10.

[7] Jia Chen, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2020. A Context-Aware Click Model for Web Search. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 88–96.

[8] Aleksandr Chuklin, Ilya Markov, and Maarten de Rijke. 2015. *Click Models for Web Search*. Morgan & Claypool.

[9] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. 2008. An Experimental Comparison of Click Position-Bias Models. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*. 87–94.

[10] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. 2008. An experimental comparison of click position-bias models. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*. 87–94.

[11] Fernando Diaz, Ryen White, Georg Buscher, and Dan Liebling. 2013. Robust models of mouse movement on dynamic web search results pages. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*. 1451–1460.

[12] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12, Jul (2011), 2121–2159.

[13] Georges E Dupret and Benjamin Piwowarski. 2008. A user browsing model to predict search engine click data from past observations.. In *Proceedings of the 31st International ACM SIGIR Conference on Research and Development in Information Retrieval*. 331–338.

[14] Zhichong Fang, Aman Agarwal, and Thorsten Joachims. 2019. Intervention harvesting for context-dependent examination-bias estimation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 825–834.

[15] Fan Guo, Chao Liu, Anitha Kannan, Tom Minka, Michael Taylor, Yi-Min Wang, and Christos Faloutsos. 2009. Click Chain Model in Web Search. In *Proceedings of the 18th International Conference on World Wide Web*. 11–20.

[16] Fan Guo, Chao Liu, and Yi Min Wang. 2009. Efficient Multiple-Click Models in Web Search. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*. 124–131.

[17] Huifeng Guo, Jinkai Yu, Qing Liu, Ruiming Tang, and Yuzhou Zhang. 2019. PAL: a position-bias aware learning framework for CTR prediction in live recommender systems. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 452–456.

[18] Ziniu Hu, Yang Wang, Qu Peng, and Hang Li. 2019. Unbiased LambdaMART: An Unbiased Pairwise Learning-to-Rank Algorithm. In *The World Wide Web Conference*. 2830–2836.

[19] Jiarui Jin, Yuchen Fang, Weinan Zhang, Kan Ren, Guorui Zhou, Jian Xu, Yong Yu, Jun Wang, Xiaoqiang Zhu, and Kun Gai. 2020. A Deep Recurrent Survival Model for Unbiased Ranking. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 29–38.

[20] Thorsten Joachims. 2006. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 217–226.

[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] Xiaoliang Ling, Weiwei Deng, Chen Gu, Hucheng Zhou, Cui Li, and Feng Sun. 2017. Model ensemble for click prediction in Bing search ads. In *Proceedings of the 26th International Conference on World Wide Web Companion*. 689–698.

[23] Dugang Liu, Pengxiang Cheng, Zhenhua Dong, Xiuqiang He, Weike Pan, and Zhong Ming. 2020. A General Knowledge Distillation Framework for Counterfactual Recommendation via Uniform Data. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 831–840.

[24] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 1412–1421.

[25] Jiaxin Mao, Cheng Luo, Min Zhang, and Shaoping Ma. 2018. Constructing Click Models for Mobile Search. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*. 775–784.

[26] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *International Conference on Machine Learning*. 807–814.

[27] Harrie Oosterhuis and Maarten de Rijke. 2018. Ranking for relevance and display preferences in complex presentation layouts. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*. 845–854.

[28] Liang Pang, Jun Xu, Qingyao Ai, Yanyan Lan, Xueqi Cheng, and Jirong Wen. 2020. SetRank: Learning a Permutation-Invariant Ranking Model for Information Retrieval. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 499–508.

[29] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*. 1310–1318.

[30] Rama Kumar Pasumarthi, Honglei Zhuang, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2020. Permutation Equivariant Document Interaction Network for Neural Learning to Rank. In *Proceedings of the 2020 ACM SIGIR on International Conference on Theory of Information Retrieval (ICTIR '20)*. 145–148.

[31] Zhen Qin, Suming J. Chen, Donald Metzler, Yongwoo Noh, Jingzheng Qin, and Xuanhui Wang. 2020. Attribute-Based Propensity for Unbiased Learning in Recommender Systems: Algorithm and Case Studies. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2359–2367.

[32] Zhen Qin, Le Yan, Honglei Zhuang, Yi Tay, Rama Kumar Pasumarthi, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2021. Are Neural Rankers still Outperformed by Gradient Boosted Decision Trees?. In *Proceedings of the 9th International Conference on Learning Representations (ICLR '21)*.

[33] Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting Clicks: Estimating the Click-through Rate for New Ads. In *Proceedings of the 16th International Conference on World Wide Web*. 521–530.

[34] Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting clicks: estimating the click-through rate for new ads. In *Proceedings of the 16th international conference on World Wide Web*. 521–530.

[35] Ali Vardasbi, Maarten de Rijke, and Ilya Markov. 2020. Cascade Model-Based Propensity Estimation for Counterfactual Learning to Rank. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2089–2092.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[37] Chao Wang, Yiqun Liu, Meng Wang, Ke Zhou, Jian-yun Nie, and Shaoping Ma. 2015. Incorporating Non-Sequential Behavior into Click Models. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 283–292.

[38] Nan Wang, Zhen Qin, Xuanhui Wang, and Hongning Wang. 2021. Non-Clicks Mean Irrelevant? Propensity Ratio Scoring As a Correction. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*.

[39] Xuanhui Wang, Michael Bendersky, Donald Metzler, and Marc Najork. 2016. Learning to Rank with Selection Bias in Personal Search. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '16)*. 115–124.

[40] Xuanhui Wang, Nadav Golbandi, Michael Bendersky, Donald Metzler, and Marc Najork. 2018. Position bias estimation for unbiased learning to rank in personal search. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 610–618.

[41] Xiaohui Xie, Yiqun Liu, Xiaochuan Wang, Meng Wang, Zhijing Wu, Yingying Wu, Min Zhang, and Shaoping Ma. 2017. Investigating examination behavior of image search users. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 275–284.

[42] Xiaohui Xie, Jiaxin Mao, Maarten de Rijke, Ruizhe Zhang, Min Zhang, and Shaoping Ma. 2018. Constructing an interaction behavior model for web image search. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*. 425–434.

[43] Jingyu Zhao, Feiqing Huang, Jia Lv, Yanjie Duan, Zhen Qin, Guodong Li, and Guangjian Tian. 2020. Do RNN and LSTM have Long Memory? *arXiv preprint arXiv:2006.03860* (2020).

[44] 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.

[45] Honglei Zhuang, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2020. Feature transformation for neural ranking models. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1649–1652.