UNIVERSITY OF AMSTERDAM

UvA-DARE (Digital Academic Repository)

Unbiased Learning to Rank Meets Reality: Lessons from Baidu's Large-Scale Search Dataset

Hager, P.; Deffayet, R.; Renders, Jean-Michel; Zoeter, Onno; de Rijke, Maarten

DOI
10.1145/3626772.3657892

Publication date
2024

Document Version
Final published version

Published in
SIGIR '24

License
CC BY

Link to publication

Citation for published version (APA):
Hager, P., Deffayet, R., Renders, J.-M., Zoeter, O., & de Rijke, M. (2024). Unbiased Learning to Rank Meets Reality: Lessons from Baidu's Large-Scale Search Dataset. In *SIGIR '24: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval : July 14-18, 2024, Washington, DC, USA* (pp. 1546-1556). Association for Computing Machinery. https://doi.org/10.1145/3626772.3657892

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, P.O. Box 19185, 1000 GD Amsterdam, The Netherlands. You will be contacted as soon as possible.

UvA-DARE is a service provided by the library of the University of Amsterdam (https://dare.uva.nl)

Download date:10 Oct 2026

Unbiased Learning to Rank Meets Reality: Lessons from Baidu's Large-Scale Search Dataset

Philipp Hager*
University of Amsterdam
Amsterdam, The Netherlands
p.k.hager@uva.nl

Romain Deffayet*
University of Amsterdam
Amsterdam, The Netherlands
romain.deffayet@naverlabs.com

Jean-Michel Renders
Naver Labs Europe
Meylan, France
jean-michel.renders@naverlabs.com

Onno Zoeter
Booking.com
Amsterdam, The Netherlands
onno.zoeter@booking.com

Maarten de Rijke
University of Amsterdam
Amsterdam, The Netherlands
m.derijke@uva.nl

ABSTRACT

Unbiased learning-to-rank (ULTR) is a well-established framework for learning from user clicks, which are often biased by the ranker collecting the data. While theoretically justified and extensively tested in simulation, ULTR techniques lack empirical validation, especially on modern search engines. The Baidu-ULTR dataset released for the WSDM Cup 2023, collected from Baidu's search engine, offers a rare opportunity to assess the real-world performance of prominent ULTR techniques. Despite multiple submissions during the WSDM Cup 2023 and the subsequent NTCIR ULTRE-2 task, it remains unclear whether the observed improvements stem from applying ULTR or other learning techniques.

In this work, we revisit and extend the available experiments on the Baidu-ULTR dataset. We find that standard unbiased learning-to-rank techniques robustly improve click predictions but struggle to consistently improve ranking performance, especially considering the stark differences obtained by choice of ranking loss and query-document features. Our experiments reveal that gains in click prediction do not necessarily translate to enhanced ranking performance on expert relevance annotations, implying that conclusions strongly depend on how success is measured in this benchmark.

CCS CONCEPTS

• **Information systems** → **Learning to rank**.

KEYWORDS

Learning to rank, Counterfactual learning-to-rank, Click models

ACM Reference Format:
Philipp Hager, Romain Deffayet, Jean-Michel Renders, Onno Zoeter, and Maarten de Rijke. 2024. Unbiased Learning to Rank Meets Reality: Lessons from Baidu's Large-Scale Search Dataset. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24), July 14–18, 2024, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3626772.3657892

*Both authors contributed equally to the paper

This work is licensed under a Creative Commons Attribution International 4.0 License.

SIGIR '24, July 14–18, 2024, Washington, DC, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0431-4/24/07.
https://doi.org/10.1145/3626772.3657892

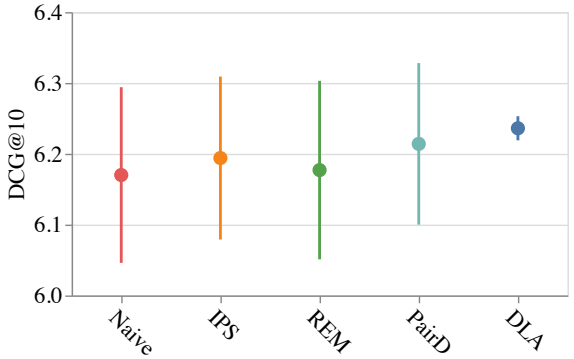

Figure 1: The original experiments on Baidu-ULTR show that none of the four compared ULTR methods outperform a naive method not correcting for position bias. Data from [65, Table 6] and visualization by us.

1 INTRODUCTION

Historically, the field of learning-to-rank employed human experts to annotate the relevance of search results to create training data for ranking models [20, 21, 44]. As expert labeling is expensive [11], infeasible in certain applications [56], and potentially misaligned with user preferences [50], many practitioners seek to leverage implicit user feedback, often collected in the form of clicks. However, clicks are usually a biased signal of relevance, as they often depend on the position of an item [33], its surrounding items [19, 64], or even the user's trust in the system to place relevant items on top [1, 33, 55]. Over the years, the field of unbiased learning-to-rank (ULTR) has proposed methods to mitigate biases when training ranking models on click data [29].

ULTR methods, especially in academic research, are often evaluated in synthetic or semi-synthetic setups [3, 34, 41, 54, 55]. The most common semi-synthetic setup goes back to the seminal work by Joachims et al. [34]. The setup uses learning-to-rank datasets from web search engines containing features and expert judgments of each query-document pair [11, 20, 44] and simulates clicks and biases based on synthetic user models (e.g., the position-based model (PBM) [19, 46] or the cascade model [19]).

While the semi-synthetic simulation setup has demonstrated the effectiveness of many ULTR methods, it is often unclear to academic researchers how these methods fare in a real-world setup. Recently,

Zou et al. [65] released Baidu-ULTR, an extensive real-world dataset for web search. It comprises over 1.2 billion user sessions with click data and 397, 572 annotated query-document pairs for evaluation. The dataset contains rich user feedback, including clicks, dwell time, whether a document was scrolled off-screen, and whether the user returned to the result page after clicking an item. In contrast to the classic LTR datasets used in semi-synthetic simulation setups, the dataset does not contain pre-computed ranking features, but the original search query, title, and abstract of each result (for privacy reasons in tokenized form, with a private vocabulary). Therefore, the dataset enables training and evaluating transformer-based rankers, such as MonoBERT [39] or MonoT5 [43].

Zou et al. [65] train a MonoBERT ranker and provide baseline results of this model fine-tuned on four standard ULTR methods: inverse propensity scoring (IPS) [34, 56], regression expectation-maximization (RegressionEM) [57], the dual learning algorithm (DLA) [3], and pairwise debiasing (PairD) [31]. Interestingly, Zou et al. [65] find that these prominent ULTR methods overall do not perform significantly better than a naive BERT model trained on clicks without any bias correction,[1] as we report in Figure 1. In subsequent work, the Baidu-ULTR dataset was used at multiple ranking competitions (the WSDM Cup 2023 [66, 67] and NTCIR's ULTRE-2 [38]), in which multiple teams reported noticeably higher ranking performance, with and without ULTR techniques. *It therefore remains unclear whether the observed improvements were due to unbiased learning-to-rank, and ultimately whether ULTR techniques improve performance on this dataset.*

In this work, we reproduce and extend the experiments by Zou et al. [65], in light of several considerations:

(1) The initial findings of ULTR methods bringing barely any benefits on the largest real-world dataset available for ULTR have substantial implications for the field and warrant more scrutiny.
(2) Three competitions have been conducted on the Baidu-ULTR dataset, with participants reporting vastly improved ranking results, though not necessarily using ULTR. A preliminary experiment by us revealed that a random ranking of the annotated test dataset achieves a DCG@10 $\approx$ 6.69, which is higher than all baselines in Figure 1 and suggests potential issues with the original experiments.
(3) According to the official codebase that comes with the Baidu-ULTR dataset,[2] the original comparison focuses on pointwise versions of the compared ranking methods (except for pairwise debiasing). Therefore, it remains open how pointwise, pairwise, and listwise ULTR methods compare on this dataset.
(4) And lastly, a closer look at the Baidu-ULTR dataset and the original codebase revealed some disputable design decisions. IPS, for example, requires an estimate of position bias on the given dataset, while the creators of the Baidu-ULTR dataset confirmed using a hardcoded position bias from the ULTRA library in their experiments [53]. We also found that placeholder documents make up almost 20% of the Baidu-ULTR dataset (see Section 3), with an unknown impact on the trained models.

By revisiting the experiments reported by Zou et al. [65] and the cups' participants, we address the following research questions:

(RQ1) Does unbiased learning-to-rank improve performance on the Baidu-ULTR dataset over naive, non-debiasing models?
(RQ2) How do ULTR methods fare against each other, and how do ranking losses and input features affect their performance?
(RQ3) Can ULTR methods be applied during language model training, and do they bring improved performance?

We answer (RQ1) negatively as we cannot find robust improvements in ranking performance in a fair comparison between ULTR-based and naive methods. For (RQ2), we report minor but existing differences between ULTR methods and showcase large and reliable differences due to ranking losses and input features. Regarding (RQ3), we find that while certain ULTR methods improve over their naive counterpart if applied during language model training, the interactions of ULTR and transformer-based models are not well understood and can even lead to decreased performance.

Besides answering our research questions, our contributions are:

- We release two smaller, cleaned, and pre-processed datasets derived from Baidu-ULTR,[3] along with transformer-based query-document embeddings to enhance the reproducibility of our work and ease access to Baidu-ULTR.
- We publish highly optimized Jax [8, 32] implementations of various standard ULTR methods to enable ULTR researchers to leverage the vast scale of the Baidu-ULTR dataset effectively.[4]
- We tune six ULTR methods and three naive ranking methods on transformer-based embeddings and learning-to-rank features.
- We train six MonoBERT models from scratch, including listwise and ULTR-based loss functions; our model weights are public.[5]

2 RELATED WORK

2.1 Unbiased Learning to Rank

The field of unbiased learning-to-rank can be broadly divided into click modeling and counterfactual learning-to-rank. Click modeling encodes assumptions on user behavior into probabilistic models [12, 19, 26]. Assumed effects, such as position bias or item relevance, are represented as (latent) variables that are jointly inferred via maximum likelihood estimation [18]. Notable examples include the *position-based model* (PBM) [19, 46], which assumes that users click only on positions they examine and items they find relevant, and the *cascade model* [19], which assumes that users scroll from top to bottom, click on the first relevant item, then leave the page. In recent years, more complex and flexible click models have been proposed using neural architectures, semantic embeddings, and bias features beyond position [7, 14, 58, 64]. We implement two neural versions of the original PBM: RegressionEM [57] and an additive two-tower model popular in industry applications [28, 61, 63].

The counterfactual learning-to-rank community has historically employed simpler user models such as the PBM [34, 56] or the affine model for trust bias [55] and focused on training more effective ranking models, mitigating biases with inverse propensity scoring [34, 41, 56]. Our work includes an extended binary cross-entropy loss [6, 49] as a pointwise IPS baseline and an extended

[1] Zou et al. [65] find that DLA outperforms a naive baseline on frequent head queries.
[2] https://github.com/ChuXiaokai/baidu_ultr_dataset/
[3] https://huggingface.co/datasets/philipphager/baidu-ultr_uva-mlm-ctr
[4] https://github.com/philipphager/ultr-reproducibility
[5] https://github.com/philipphager/baidu-bert-model

softmax loss as a listwise IPS baseline [3, 9]. Both methods require position bias estimations, for which we release the implementation of three intervention harvesting methods (more in Section 3.2). We also include two counterfactual methods jointly estimating position bias and item relevance, the dual learning algorithm (DLA) [3] and pairwise debiasing (PairD) [31]. We introduce all unbiased learning to rank methods used in this work in Section 4.

2.2 ULTR on the Baidu-ULTR dataset

Most prior work on the Baidu-ULTR dataset has revolved around three public competitions: Two tracks at the WDSM Cup 2023 – *unbiased learning-to-rank* [67] and *pretraining for web search* [66] – and the ULTRE-2 track at NTCIR [38].

In the ULTR track of the WSDM cup, participants were restricted to train ranking models using click data. Two of the top three teams ended up applying ULTR techniques [17, 60], notably the winning team which employed a two-tower model with a softmax loss [17]. However, participants did not compare with non-ULTR methods (as the main objective is to win a competition). All three top-performing teams incorporated a BERT-based cross-encoder [39], with the first two teams training models from scratch [13, 17]. Chen et al. [13] voice concerns about the performance of their BERT model trained using the officially released starter-kit, as it plateaued around a DCG@10 $\approx$ 7. Their remark has led us to train our own BERT cross-encoder models from scratch.

The second task of the WSDM Cup allowed participants to pretrain language models from clicks before fine-tuning on expert annotations. All top teams combined the output of their BERT models with traditional LTR features and tuned gradient-boosting models on annotations [35, 36, 52]. Only the third-placed team incorporated a conventional ULTR approach with a softmax ranking loss and IPS during BERT pretraining.

In the NTCIR 17 ULTRE-2 track [38], the organizers released a subset of the data ($\approx$ 1 million sessions) to make Baidu-ULTR more accessible, using the best-performing BERT model trained by Li et al. [36] during the WSDM Cup to create query-document embeddings in combination with traditional lexical matching features.[6] In their baseline experiments, they find that DLA models perform better than non-ULTR models trained with a pointwise loss but only marginally better than those trained with a listwise loss. The only participating team, Yu et al. [59], proposed to alleviate selection bias on items that are relevant but never clicked by using a DLA model to re-annotate non-clicked documents. However, they do not compare with other ULTR and non-ULTR baselines.

Overall, participants in the three cups reported noticeably higher ranking performance than the original experiments by the Baidu-ULTR dataset authors [65], and identified helpful techniques, such as re-weighting queries to address the long-tail query distribution [52], negative sampling of documents [52], ignoring clicks on items displayed for a short time [36], or pseudo-relevance feedback [59]. However, it remains unclear whether observed improvements stemmed from debiasing the click feedback or such orthogonal techniques. In order to identify the contribution of click debiasing to the results, we extend the work by Niu et al. [38] and Zou et al. [65] to larger datasets, more ULTR methods, fairer baselines, and investigate the interactions of ULTR with language model training. Next, we introduce the dataset used in this work in more detail.

[6]Note that the first ULTRE-1 task at NTCIR 16 was not yet conducted on Baidu-ULTR but on a semi-synthetic simulation setup [62].

3 OVERVIEW OF BAIDU-ULTR

3.1 Description of the data

The Baidu-ULTR dataset [65] contains more than 1.2 billion search sessions (split into 2,000 dataset partitions) with user clicks and 7,008 annotated queries for evaluation. The dataset was collected in April 2022 by randomly sampling the search traffic of Baidu [65], reflecting the long-tail query distribution of real user traffic. The dataset contains logged queries, document titles, and abstracts of the search results presented to the user. The authors tokenized all released text with a private vocabulary for user privacy. In addition to tokenized text, the dataset also contains a multitude of logged user interactions, including clicks, dwell-time, and skipping behavior, as well as item presentation features, including document height, item type, and ranking position. In this work, we solely focus on item position and user clicks.

To foster the reproducibility of this work and to ease access to the vast Baidu-ULTR dataset for academic research, we release two smaller datasets: A *language modeling dataset*, comprised of the first 125 partitions of the dataset for training transformer models from scratch, and a smaller *reranking dataset* comprised of four partitions to compare ULTR on pre-computed query-document features. In the following, we describe the preprocessing common to both datasets and analyze the properties of our reranking dataset.

3.1.1 Pre-processing. (i) We use the md5 hash of the query tokens and document URL, respectively, as query and document identifiers. (ii) We discovered that over 20% of the Baidu-ULTR dataset comprised only two title-abstract token combinations. To avoid these documents polluting model training, we remove the *what other people searched* item from the ranking ($\approx$ 9% of documents) and skip all documents with only a dash in the title, indicating no available content ($\approx$ 13% of documents).[7] Note that documents displayed after a skipped document keep their original position and do not, e.g., move to the position of a skipped document. (iii) Third, we drop all queries with less than five documents to display, affecting $\approx$ 2% of the train queries and $\approx$ 0.3% of annotated queries, leaving a remainder of 6,985 annotated test queries. In contrast to previous work [38], we do not remove queries without any clicks. While this is common in ULTR as sessions without clicks do not contribute to popular pairwise ranking losses [34], they are essential to train and evaluate methods predicting calibrated click probabilities.

3.1.2 Analysis of the reranking dataset. The reranking dataset comprises three dataset partitions for training ($\approx$ 1.8 million user sessions, $\approx$ 11.7 million query-document pairs) and one partition for validation and testing ($\approx 590k$ user sessions, $\approx$ 4.8 million query-document pairs). Table 1 gives an overview of the statistics of the reranking dataset. While the dataset is a fraction of Baidu-ULTR, we highlight that it is still substantially larger than existing ULTR datasets [4] and contains more unique queries and query-document

[7]While Baidu-ULTR was tokenized with a private vocabulary, Zou et al. [65] confirmed to us that the tokens: [3742, 0111492, 0112169, 015061, 0116905] translate to *what other people searched* and token 21429 translates to "-" indicating missing content.

Table 1: Description of the reranking dataset we derive from Baidu-ULTR [65]; statistics after preprocessing.

	Annotations	Implicit feedback	
		train	test
Unique queries	6,985	1,378,901	501,215
Unique documents	381,552	9,455,953	3,557,825
Total query/doc pairs	382,038	11,715,447	4,209,900
Total sessions	-	1,779,017	593,930
Total impressions	-	14,526,276	4,848,878
Avg. docs per session		8.165	
Avg. clicks per session		0.688	
Avg. clicks per document		0.084	
% of sessions with $\geqslant$ 1 click		46.581%	
% of sessions with $\geqslant$ 2 clicks		13.082%	
% of train queries occurring in the annotated set		0.064%	
% of annotated queries occurring in the train set		12.713%	

pairs than the LTR datasets commonly used in semi-synthetic simulation for ULTR [11, 20, 21, 44].

Besides query-document text and identifiers (as described in the preprocessing section), the reranking dataset also contains three sets of pre-computed query-document features: the CLS token of the BERT cross-encoder released by Zou et al. [65], the CLS token of a BERT cross-encoder trained by us in the same fashion as the original model, and classic lexical learning-to-rank (LTR) features, including BM25 [48], Tf-IDF [47], and query-likelihood (with Jelinek-Mercer and Dirichlet smoothing [16]). We compute the required inverted index for the methods above on the *language modeling dataset* that we use to train our BERT cross-encoder for fair comparison. As we restrict ourselves to solely training on click data and only use the expert annotations in the final evaluation, we compute all lexical matching features with default parameters instead of tuning them on annotations. Also, LTR parameters published by Chen et al. [13] do not lead to major improvements over untuned default parameters in our setting. We publish two versions of the dataset on Huggingface, one using the Baidu cross-encoder[8] and one with our cross-encoder[9] with a list of all pre-computed lexical ranking features and their respective hyperparameters.

As the reranking dataset is a random sample of the larger (preprocessed) dataset, we can use it to analyze the basic properties of Baidu-ULTR. After preprocessing, the average search session contains 8 documents, with an average of 0.68 clicks per session (number of clicks/number of sessions) and $\approx$ 46.5% of sessions containing at least one click (see Table 1). Notably, we measure the overlap between queries (md5 hash of query tokens) in the click dataset and the annotated test set and find that only 12.7% of annotated queries occur in the training dataset. While they appear during training, they make up $< 0.1\%$ of the training dataset.

3.2 Position bias

Standard counterfactual LTR methods using IPS require an estimation of position bias, which we estimate on our reranking dataset.

[8] https://huggingface.co/datasets/philipphager/baidu-ultr_baidu-mlm-ctr
[9] https://huggingface.co/datasets/philipphager/baidu-ultr_uva-mlm-ctr

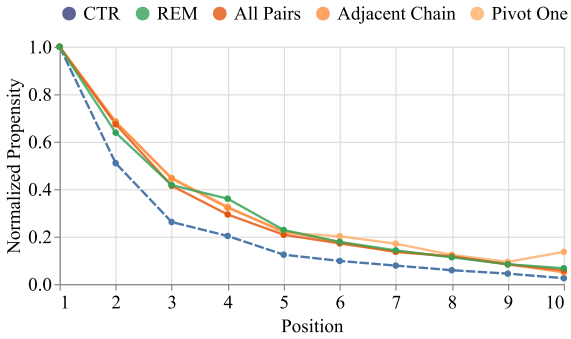

Figure 2: Position bias as estimated by RegressionEM and three intervention harvesting methods compared to the mean CTR. Propensities were normalized by position one.

We implement three intervention harvesting techniques that leverage the co-occurrence of the same query-document pair at different ranks [2]. Intervention harvesting mines query-document pairs logged in various positions, ideally due to distinct rankers in an A/B test ranking items differently. We can use this natural variability of encountering a document in different positions to estimate bias using click ratios between neighboring positions (Adjacent Pair), each position and a fixed rank (Pivot Rank), or between all ranking positions (All Pairs). We refer the reader to Agarwal et al. [2] for a detailed introduction to intervention harvesting.

We also estimate position bias using RegressionEM (REM) [57], which leverages query-document embeddings. REM estimates position bias over co-occurrences of query-document pairs with similar features rather than strictly identical query-document pairs, like intervention harvesting. In our case, we use the semantic embeddings of our naive BERT cross-encoder. Figure 2 displays our propensity estimations and the click-through rate per rank. The estimated position bias is broadly consistent across methods. REM and intervention harvesting arrive at similar estimations from different query-document representations. Our finding suggests the presence of a noticeable, top-heavy position bias affecting user behavior on Baidu-ULTR for which ULTR methods should be helpful.

4 UNBIASED LEARNING-TO-RANK METHODS

This section introduces the different ULTR methods used throughout this work. Similarly to Zou et al. [65], we restrict ourselves to methods that address position bias according to the PBM and focus on benchmarking the learning algorithm used by each method.

First, we introduce the notation we will use to describe each method. Let q be a user query and D_q the ordered list of documents for q. Let $d \in D_q$ be a document in the list displayed at position k. We use $R_{q,d}$, E_k and $C_{q,d,k}$, respectively, to denote the random events for: (i) document d being relevant for query q, (ii) position k of the result page being examined by the user, and (iii) document d placed in position k for query q being clicked. We assume users follow the position-based model (PBM), meaning that users click only if they observed the position k of an item and deemed the displayed document d to be relevant:

$$P(C_{q,d,k}) = P(E_k) \times P(R_{q,d}). \quad (1)$$

In the following, we use $\mathcal{L}$ to reference a ranking loss function; in this work, either binary cross-entropy, softmax cross-entropy [9], or

LambdaRank [10]. All ranking methods aim to estimate document relevance $r(q,d) = P(R_{q,d})$ and optionally examination $e(k) = P(E_k)$ (or related quantities) from observed clicks $c \in \{0,1\}$. We use $\tilde{x}$ to denote estimators of a quantity x. We can now define the methods compared in our study:

Naive: is the common ULTR term for applying a ranking loss without any position bias correction. This means that we naively interpret clicks as positive/negative user feedback:

$$\mathcal{L}^{\text{Naive}} = \mathcal{L}(\tilde{c}(q,d); c). \tag{2}$$

We implement a pointwise version of this model using binary cross-entropy and two listwise versions using softmax cross-entropy [9] and LambdaRank [10], respectively.

Two-Tower [58, Two-Tower Model]: jointly learns the maximum-likelihood parameters of a relevance model $\tilde{r}$ and an examination model $\tilde{e}$, directly mirroring the PBM assumptions:

$$\mathcal{L}^{\text{TwoTower}} = \mathcal{L}\left(\sigma\left(\tilde{e}(k) + \tilde{r}(q,d)\right); c\right). \tag{3}$$

We use the additive formulation [58, 61] with $\tilde{r}$ and $\tilde{e}$ representing logits, σ the sigmoid function, and $\mathcal{L}$ the binary cross-entropy loss.

RegressionEM [57, Regression expectation-maximization]: learns a maximum likelihood estimate of relevance and examination parameters through expectation-maximization [24]:

$$\begin{aligned}\mathcal{L}^{\text{REM}} = &\mathcal{L}(\tilde{r}(q,d); c + (1-c)\overline{r}(q,d)) \\ &+ \mathcal{L}\left(\tilde{e}(k); c + (1-c)\overline{e}(k)\right),\end{aligned} \tag{4}$$

where $\overline{r}(q,d) = \frac{\tilde{r}(q,d)(1-\tilde{e}(k))}{1-\tilde{r}(q,d)\tilde{e}(k)}$ and $\overline{e}(k) = \frac{\tilde{e}(k)(1-\tilde{r}(q,d))}{1-\tilde{r}(q,d)\tilde{e}(k)}$ are the posterior relevance and examination probabilities. In practice, we use binary cross-entropy as the base loss and replace the original expectation-maximization procedure with a numerically stable gradient-based version suggested in TF-Rank [42] by performing the posterior computation with logits.

IPS [34, Inverse propensity scoring]: Given known examination probabilities e_k, IPS re-weights the click labels by the propensity (i.e., normalized examination probability) of the current position:

$$\mathcal{L}^{\text{IPS}} = \mathcal{L}\left(\tilde{r}(q,d); \frac{\max(\tau, e_1)}{\max(\tau, e_k)} c\right). \tag{5}$$

In our experiments, we use propensities estimated by the AllPairs intervention harvesting method [2] and reduce variance by clipping examination probabilities to a minimum value of $\tau = 0.1$ [34]. We implement a pointwise [6, 49] and listwise IPS variant [3, 9].

DLA [3, Dual learning algorithm]: separately learns tunable relevance scores with fixed propensities and tunable propensities with fixed relevance scores, essentially applying IPS twice:

$$\mathcal{L}^{\text{DLA}} = \mathcal{L}\left(\tilde{r}(q,d); \frac{\tilde{e}(1)}{\tilde{e}(k)} c\right) + \mathcal{L}\left(\tilde{e}(k); \frac{\tilde{r}(q,d_1)}{\tilde{r}(q,d)} c\right), \tag{6}$$

where d_1 is the document ranked first on the current result page. We use softmax cross-entropy as the base loss and perform a softmax-normalization of examination and relevance probabilities $\tilde{r}$ and $\tilde{e}$ within a result page, as in the original paper [3].

PairD [31, Pairwise debiasing]: learns positive $\tilde{e}^+$ and negative $\tilde{e}^-$ propensities (i.e., corresponding to clicks and non-clicks, respectively) with a constraint on the norm of the learned propensities:

$$\mathcal{L}^{\text{PairD}} = \frac{\mathcal{L}(\tilde{r}(q,d); c)}{\tilde{e}^+(k)\tilde{e}^-(k)} + \|\tilde{e}^+\|_1 + \|\tilde{e}^-\|_1. \tag{7}$$

In practice, we use the L_1-norm for propensity regularization and learn the relevance scores using the LambdaRank loss [10], as in the original paper [31]. Note that the theoretical validity of this method under non-trivial position bias has been challenged by Oosterhuis [40]. Still, we include it in our comparison as it has demonstrated strong empirical performance in past comparisons [5].

5 EXPERIMENTAL SETUP

5.1 Training and evaluation procedure

All models are trained on our reranking dataset, and we use a 50/50 random split of our test click dataset (see Table 1) for validation and testing. We evaluate ranking performance on 6,985 annotated queries, where experts rated each query-document pair's relevance on a scale from 0 to 4. We point to Zou et al. [65] for more details on the annotation process. In this work, we do not use available side information such as dwell-time, off-screen scrolling, and returns to the result page. Instead, we focus on query and document content, document position and click label as our training data.

We measure ranking performance on the annotated set using discounted cumulative gain (DCG) at different truncation levels and mean reciprocal rank (MRR) at 10, as well as negative log-likelihood (NLL) for click prediction.

5.2 Models

Traditionally, unbiased learning-to-rank practitioners train small ranking models with LTR features such as BM25 [48] or TF-IDF [47] as input [3, 41, 55]. To investigate the role of semantic embeddings and the interaction of ULTR with language model training on such a large dataset, we train two types of models: transformer-based *language models* trained from scratch following the MonoBERT cross-encoder [39], and multi-layer perceptron-based *reranking models*, that take as input either traditional LTR features or fixed query-document embeddings obtained by the language models.

5.2.1 Language models. We train cross-encoders on the Baidu-ULTR dataset from scratch in Jax [8], building on the FlaxBERT implementation from Huggingface.[10] The input to our BERT models is: `[CLS] query [SEP] doc_title [SEP] doc_abstract`. Following [17, 65], we truncate the input to a maximum length of 128 tokens. Compared to the original BERT training scheme [25], we keep the masked-language modeling task (masking 30% of input tokens at random) but discard the next-sentence prediction task. Instead, we follow the MonoBERT [39] setup and apply a linear layer to the BERT CLS token to output a click prediction score. We train multiple models with different click-based loss functions: naive click prediction with a pointwise (binary cross-entropy) and a listwise (softmax cross-entropy) loss, pointwise and listwise IPS, a pointwise two-tower loss, and a listwise DLA loss (see Section 4). All loss functions were built on top of the Rax library [32].

We keep the architecture of the original $\text{BERT}_{\text{base}}$ model, using 12 transformer layers, 12 attention heads, 768 output dimensions, and, following Chen et al. [17], a vocabulary size of 22,000. All language models are trained with a batch size of 256 (4×64) for 2 million gradient steps, i.e., on 512 million documents. Training each base model on four GPUs (NVIDIA H100-80GB) takes around

[10] https://huggingface.co/docs/transformers/model_doc/bert#transformers.FlaxBertForPreTraining

46 hours, which is a speed-up of almost 50% compared to an early PyTorch implementation of ours using the exact same library, architecture, and hardware. We use the AdamW optimizer [37] with a fixed learning rate of 5×10^{-5} and a weight decay of 0.01. Given the substantial compute invested to train each language model, we rely on prevalent default parameters for BERT (listed in our repository).

5.2.2 Reranking models. To investigate the interactions of ULTR with semantic query-document embeddings and traditional LTR features on Baidu-ULTR, we train several smaller feed-forward networks as reranking models, as is common in the ULTR community [3, 41, 54]. Our reranking models are composed of linear layers with ReLU activations and, optionally, dropout regularization. We found no benefit in applying Layer or Batch Normalization, as our BERT embeddings are already normalized. As LTR features can span a large range, we find that scaling features with $\log1\text{p}(x) = \log_e(1+|x|) \odot \text{sign}(x)$ before the first layer, as suggested by Qin et al. [45], works well. Note that the feed-forward network takes the query-document features as input and, depending on the ULTR method, outputs a relevance or click prediction. We add a single learnable model parameter per position for methods that jointly estimate position bias. In contrast to models leveraging multiple bias features [58, 62], we found no additional benefit by using a multi-layer perceptron for position bias estimation in our setting.

5.3 Hyperparameter tuning

We perform extensive hyperparameter tuning for a fair comparison of the reranking models in our experiments. Given the immense combinatorial space of hyperparameters, methods, and datasets, we adopt an incremental hyperparameter tuning strategy as advocated by Godbole et al. [27]. First, we tune our model architecture per set of input features based on the pointwise naive model. We tune the number of hidden dimensions $\in \{64, 128, 256, 512, 1024\}$ and the number of layers $\in \{2, 3, 4, 5\}$ over three random seeds. As both network depth and width can interact with the learning rate, we tune each parameter combination over three learning rates $\in \{0.001, 0.0005, 0.0001\}$, i.e., we treat the learning rate as a nuisance parameter [27]. We adopt this incremental tuning procedure as we found no major discrepancies in model architecture between ULTR methods but instead between sets of input features. Subsequently, we adopt the architecture of the pointwise naive model per dataset for all other methods.

A five-layer perceptron with 512 hidden dimensions yields optimal results for our LTR features, while a five-layer perceptron with 256 dimensions performs best for the Baidu BERT embeddings. Additionally, a two-layer perceptron with 256 dimensions was the most suitable choice for our BERT embeddings. Given each base model architecture, we tune the final dropout $\in \{0, 0.3\}$ and learning rate $\in \{0.001, 0.0005, 0.0003, 0.0001\}$ for each method and dataset over three random seeds. We use the AdamW optimizer [37] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\varepsilon = 1e^{-8}$. Most models work best with a learning rate of 0.0001 and do not benefit from dropout. The full list of hyperparameters is available in our online repository. Given the final hyperparameters for each method, we train all methods for 50 epochs, stopping early after five epochs of no improvement of the validation loss computed on clicks. Lastly, while this tuning step improved the performance of all methods, our main findings are consistent across many different hyperparameter combinations.

6 RESULTS

Before presenting our findings, we recall our research questions:

(RQ1) Does unbiased learning-to-rank improve performance on the Baidu-ULTR dataset over naive, non-debiasing models?

(RQ2) How do ULTR methods fare against each other, and how do ranking losses and input features affect their performance?

(RQ3) Can ULTR methods be applied during language model training and do they bring improved performance?

6.1 Unbiased learning-to-rank yields no or tiny improvements on Baidu-ULTR

In our first experiment, we train reranking models with pointwise and listwise ULTR loss functions and their respective naive, non-debiasing counterparts on three different types of input features: the CLS token of Zou et al. [65]'s pointwise cross-encoder, the CLS token of our pointwise cross-encoder, and learning-to-rank features (see Section 3.1.2). We list comprehensive results in Table 3. First, we focus on the ranking performance on expert annotations as measured in DCG and MRR. We display the DCG@10 in Figure 3.

At first glance at Figure 3, echoing previous work [38, 62], we observe that *we can get a much higher ranking performance from our cross-encoder than the initially released BERT cross-encoder.* Models trained on LTR features and even of a trivial baseline assigning random scores (grey dotted line) to the annotated documents beat all models trained on the original Baidu BERT features.

Second, given a set of input features, the choice of ranking loss used as a base for ULTR methods matters greatly (comparing the colored groups in Figure 3). Methods based on LambdaRank outperform methods based on the listwise softmax loss, which outperform those based on the pointwise binary cross-entropy loss. In contrast, *applying ULTR techniques yields marginal ranking improvements at most compared to their naive, non-debiasing counterpart.*

In detail, we observe that the pointwise two-tower and IPS models do not consistently and significantly improve performance compared to the naive pointwise loss – IPS is even significantly worse on both BERT features – and RegressionEM is the only pointwise ULTR method that brings small but sometimes significant improvements. Regarding listwise methods, IPS and DLA significantly but marginally improve on our BERT and LTR features. Finally, PairD is no better and sometimes worse than its naive LambdaRank counterpart. *These results are broadly consistent across input features, suggesting that standard ULTR methods struggle to bring improvement to Baidu-ULTR, regardless of the query-document features.*

Overall, we can answer research questions (RQ1) and (RQ2): on the Baidu-ULTR dataset, unbiased learning-to-rank methods do not consistently improve ranking performance on expert annotations, particularly when contrasted with the significant and reliable differences based on the choice of query-document features and ranking loss. Our reranking results confirm the findings of Zou et al. [65].

6.2 Language model training is sensitive to the choice of unbiased learning-to-rank method

Given the poor performance of ULTR on our reranking datasets (see Section 6.1), we further investigate whether training language models with ULTR is a promising direction for future research. We train six MonoBERT [39] cross-encoders with different pointwise and listwise loss functions, including ULTR loss functions.

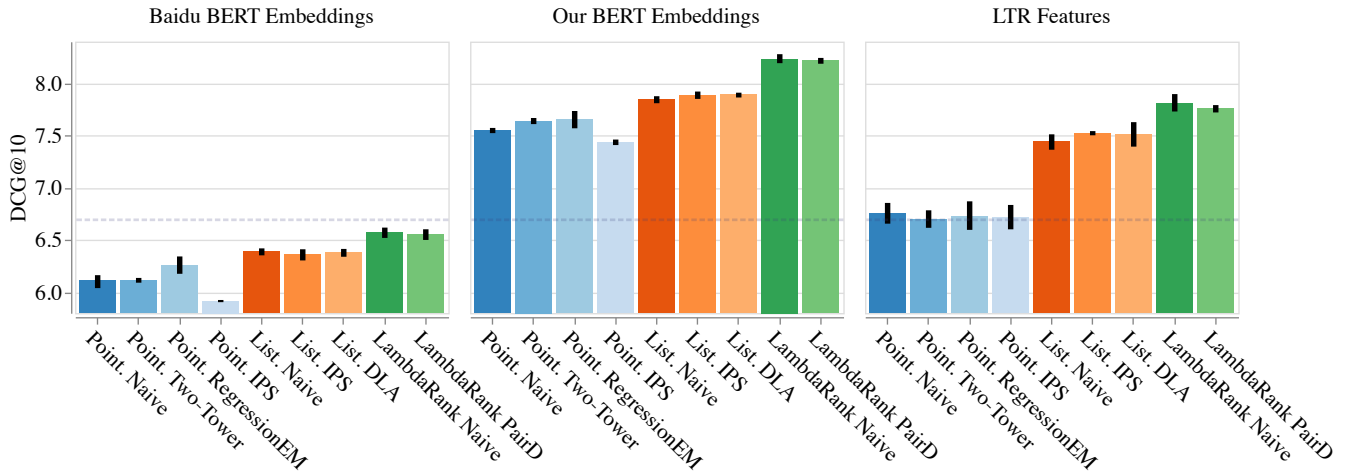

Figure 3: Comparing ULTR methods on pre-trained BERT embeddings and LTR features. We display the average ranking performance measured in DCG@10 over five independent runs and plot a bootstrapped 95% confidence interval. The grey dotted line indicates the performance of a random ranker.

Table 2 gives an overview of applying ULTR methods during language model training. In contrast to the inefficacy of ULTR-based rerankers, *we observe stark differences when applying ULTR methods during language model training*. Similar to the reranking task, approaches based on the listwise softmax loss perform better than those based on pointwise binary cross-entropy. The additional application of ULTR brings considerable improvements with the pointwise two-tower objective. However, we also observe substantial degradations with both IPS and DLA. The best method is the naive listwise softmax loss without any debiasing objective.

These inconclusive results make us cautious about answering (RQ3): ULTR methods substantially impact language model training more than reranking with fixed embeddings. However, the influence of ULTR on ranking performance and learned query-document representations needs further investigation.

6.3 Click prediction does not imply ranking performance on annotations

In addition to evaluating ranking performance, we display the negative log-likelihood (NLL) of predicted click scores in Figure 4. We restrict this evaluation to pointwise ULTR reranking methods, making well-defined click predictions. All ULTR methods robustly improve click prediction compared to a naive loss, with the two-tower model showing the largest improvement on all three datasets. Moreover, as discussed in Section 3.2, the propensities learned by RegressionEM are close to those discovered through intervention harvesting and show a strong position bias. This finding suggests that unbiased learning-to-rank robustly captures position bias and can better predict user clicks because of it. *Yet, this improved click prediction does not translate to enhanced ranking performance on expert annotations.*

Critically, we can see in Table 3 that ranking documents according to their (untuned) BM25 [48] scores yields better ranking performance than all click-based models we trained. Even more surprisingly, using BM25 scores as part of the input for the reranking models – it is one of the LTR features – and training these models on clicks lowers ranking performance. This phenomenon was also

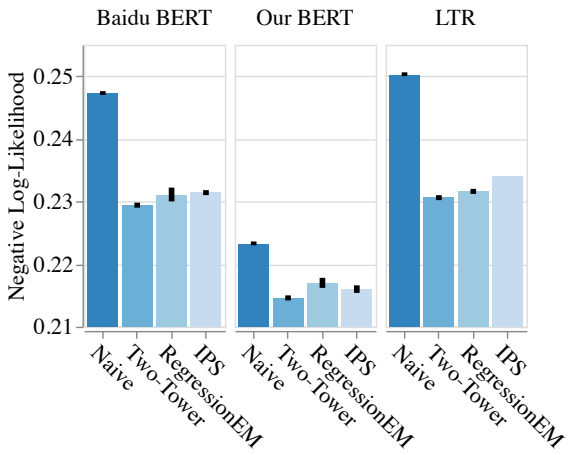

Figure 4: Click prediction performance of pointwise methods measured in negative log-likelihood; lower is better.

described by Sun et al. [52] during the WSDM Cup. These concurrent results suggest that *click prediction and ranking performance on annotations are diverging objectives on Baidu-ULTR and training models on clicks does not guarantee improvements in ranking metrics.*

7 DISCUSSION

In light of the puzzling results reported in Section 6, we review potential reasons for the failure of ULTR in Section 7.1 as well as the limitations of our study in Section 7.2. Finally, we discuss the implications for the field of unbiased learning-to-rank in Section 7.3.

7.1 Potential reasons for the failure of ULTR

No position bias. A first straightforward explanation for the inefficacy of ULTR techniques would be that position bias is not prevalent in the Baidu-ULTR dataset. However, our results in Section 3.2 show a strongly decreasing click-through rate with increasing position and a strong position bias estimated by techniques based on different methodologies. Recall that intervention harvesting uses CTR ratios of query-document pairs appearing in multiple positions

Table 2: Comparison of cross-encoder models trained from scratch on the Baidu-ULTR dataset with ULTR loss functions.

Model	DCG@10 ↑	NLL ↓
Pointwise Naive	7.251	0.227
Pointwise Two-Tower	7.456	0.218
Pointwise IPS	6.296	0.222
Listwise Naive	8.478	-
Listwise IPS	7.450	-
Listwise DLA	7.802	-

while RegressionEM uses our BERT features and expectation maximization. The fact that these results agree strongly suggests the existence of position bias in this dataset and that ULTR methods like RegressionEM and Two-Tower models could capture it.

More complex user behavior. Zou et al. [65] suggest that the actual user model might be more complicated than the PBM considered in this work. This hypothesis might explain why listwise approaches bring such considerable improvements over pointwise methods and would not be surprising as user studies have identified more complex biases [1, 51, 64]. However, ULTR methods bringing no benefits would mean that the substantial position bias we identified is negligible against other types of biases. This assumption, in turn, would contradict related work where PBM-based models trained on more complex user behaviors still improved performance, albeit by not as much as the correct user model [23, 54].

Lack of identifiability. Past studies have identified that a lack of variability in the logged data, i.e., not encountering a query-document pair at different positions, can lead to the dataset not being identifiable, in the sense that there exist infinite valid combinations of relevance and bias parameters [15, 40]. However, as this work's four bias estimation methods converged to a similar position bias, we deem this explanation rather unlikely.

Distribution shift. Poor ranking performance may also be caused by the distribution shift between the query-document pairs in the training and annotated test set. While methods train only on the top-10 search results, the annotation dataset also includes presumably much less relevant documents sampled from the top-1000 candidate documents. This shift in the input feature distribution, in conjunction with the low overlap between train and test queries (see Section 3.1.2) and the long-tail query distribution during training, might all potentially impact ULTR. More research and, ideally, clicks for queries with expert annotations are needed to better understand the impact of the various distribution shifts on ULTR.

Strong logging policy. While we do not have access to logging policy scores and rankings on the annotated test queries, the limitations imposed by an already strong production system have been well-documented [22]. A side observation of ours reinforces this possibility: methods that show strong ranking performance correlate highly with the logging policy of the click dataset (as estimated through the original item position). For future dataset releases, we recommend including logging policy scores to enable assessments of the improvement made by training new rankers on click data compared to the logging policy [22, 30].

User-annotator disagreement. Finally, the divergence of click prediction and ranking performance hints at a deeper issue: annotators may not find the same documents relevant as users scrolling through a result page, which might be personalized and part of a multi-query session to find actionable information for the current user needs. We hope further analysis of Baidu-ULTR or parallel datasets of clicks and expert annotations can clarify this hypothesis.

7.2 Limitations

First, we only considered the correction of position bias under the position-based model, while other biases might have to be mitigated [1, 54]. Second, we did not include bias features available in Baidu-ULTR beyond positions. We highlight that Chen et al. [17] report improved ranking performance by considering additional bias features, including item height or media type. Third, the variability in item position that we used for position bias estimation is likely not due to natural variability but due to unobserved confounding and algorithmic choices based on additional context information rather than a stochastic logging policy. Fourth, our work only applied ULTR during language model pre-training and on fixed BERT embeddings for reranking. We did not yet explore multi-stage pre-training schemes, such as pre-training BERT on a naive click prediction task and later fine-tuning BERT using ULTR.

Lastly, we stress that our findings are strictly limited to the Baidu-ULTR dataset. While the dataset was collected from one of the most used search engines globally, it cannot be considered representative of all real-world search scenarios. Yet, we believe that results on this large-scale dataset are important for the research community.

7.3 Implications for the field

Our results starkly contrast with many studies in semi-synthetic simulation [3, 5, 31, 34, 49, 53] and call for adjusting semi-synthetic experimental setups to reflect real-world challenges better. The richness of this dataset allows for seeding simulations with more plausible data and, therefore, might bridge the gap between simulations and reality. Moreover, the prevalence of transformer-based models and text embeddings in current IR research encourages exploring the interaction of ULTR methods with such models. As shown in Section 6.2, the same techniques applied during language model training, instead of on a small re-ranking model, can yield vastly different results.

More broadly, the puzzling results we obtained, especially the apparent divergence between clicks and relevance annotations, prompt us to rethink how we measure success in ULTR. Expert annotations are static and might not reflect user context, so results obtained on annotated datasets can be polluted by distribution shifts or logging policy performance. In particular, we believe that, whenever possible, we should evaluate ULTR methods on the tasks they are trained to accomplish, including relevance estimation, bias estimation, CTR maximization, fairness of exposure, etc.

Finally, we would like to stress that this paper is not a judgment on the ULTR field. As mentioned above, results may differ on other datasets, and most ULTR methods are theoretically justified, meaning only their validity in this scenario has been challenged. In fact, our experiments validate some intuitions formulated in the ULTR community. For instance, the stark differences between loss functions justify the motivation of Joachims et al. [34] to connect simple user models with more powerful ranking loss functions.

Table 3: Comparison of ULTR methods on pre-trained BERT embeddings and LTR features, displaying averaged results over five independent runs with standard deviation in parentheses. ↑ indicates the higher the better and ↓ the lower the better. Methods are grouped by loss, and significant differences are marked with ▲ or ▼ compared to the naive method in each group using a two-sided paired t-test with $\alpha = 0.01$ and Bonferroni correction.

Input features	Method	DCG@1 ↑	DCG@3 ↑	DCG@5 ↑	DCG@10 ↑	MRR@10 ↑	NLL ↓
	Random	1.472 (0.020)	3.148 (0.030)	4.349 (0.036)	6.693 (0.044)	0.577 (0.003)	0.944 (0.002)
	BM25 ($k_1 = 1.2, b = 0.75$)	2.211	4.656	6.377	9.544	0.716	-
Baidu BERT	Pointwise Naive	1.264 (0.030)	2.765 (0.059)	3.881 (0.068)	6.111 (0.079)	0.535 (0.007)	0.246 (0.005)
	Pointwise Two-Tower	1.266 (0.018)	2.769 (0.023)	3.893 (0.026)	6.114 (0.031)	0.533 (0.003)	0.228 (0.005)▼
	Pointwise RegressionEM	1.343 (0.034)▲	2.884 (0.074)▲	4.017 (0.091)▲	6.257 (0.107)▲	0.548 (0.005)▲	0.229 (0.005)▼
	Pointwise IPS	1.157 (0.011)▼	2.606 (0.012)▼	3.699 (0.013)▼	5.915 (0.014)▼	0.517 (0.002)▼	0.230 (0.005)▼
	Listwise Naive	1.362 (0.022)	2.940 (0.037)	4.096 (0.037)	6.388 (0.042)	0.549 (0.004)	-
	Listwise IPS	1.353 (0.034)	2.920 (0.060)	4.078 (0.063)	6.359 (0.071)▼	0.548 (0.005)	-
	Listwise DLA	1.345 (0.020)	2.917 (0.037)	4.083 (0.048)	6.378 (0.051)	0.549 (0.004)	-
	LambdaRank Naive	1.419 (0.017)	3.047 (0.039)	4.237 (0.054)	6.570 (0.065)	0.562 (0.005)	-
	LambdaRank PairD	1.410 (0.028)	3.025 (0.044)	4.215 (0.064)	6.550 (0.066)	0.559 (0.005)	-
Our BERT	Pointwise Naive	1.705 (0.013)	3.602 (0.022)	4.944 (0.023)	7.546 (0.031)	0.619 (0.002)	0.221 (0.005)
	Pointwise Two-Tower	1.656 (0.017)▼	3.556 (0.024)▼	4.948 (0.032)	7.639 (0.038)▲	0.615 (0.004)▼	0.213 (0.006)▼
	Pointwise RegressionEM	1.694 (0.031)	3.619 (0.056)	4.998 (0.081)▲	7.657 (0.111)▲	0.618 (0.004)	0.215 (0.005)
	Pointwise IPS	1.589 (0.012)▼	3.438 (0.012)▼	4.794 (0.013)▼	7.436 (0.035)▼	0.604 (0.001)▼	0.214 (0.005)
	Listwise Naive	1.798 (0.008)	3.768 (0.020)	5.167 (0.029)	7.844 (0.046)	0.631 (0.001)	-
	Listwise IPS	1.816 (0.017)	3.800 (0.027)▲	5.200 (0.031)▲	7.885 (0.043)▲	0.634 (0.002)▲	-
	Listwise DLA	1.816 (0.011)	3.806 (0.022)▲	5.210 (0.024)▲	7.890 (0.031)▲	0.634 (0.001)▲	-
	LambdaRank Naive	1.931 (0.016)	3.993 (0.029)	5.452 (0.040)	8.233 (0.055)	0.648 (0.003)	-
	LambdaRank PairD	1.932 (0.007)	3.985 (0.019)	5.438 (0.022)	8.216 (0.038)	0.646 (0.003)	-
LTR Features	Pointwise Naive	1.312 (0.034)	2.977 (0.066)	4.246 (0.090)	6.757 (0.133)	0.543 (0.008)	0.247 (0.005)
	Pointwise Two-Tower	1.333 (0.046)	2.986 (0.085)	4.241 (0.094)	6.698 (0.110)▼	0.553 (0.008)▲	0.228 (0.006)▼
	Pointwise RegressionEM	1.397 (0.068)▲	3.062 (0.133)▲	4.300 (0.166)▲	6.729 (0.181)	0.559 (0.014)▲	0.229 (0.005)▼
	Pointwise IPS	1.352 (0.044)▲	3.013 (0.095)	4.258 (0.126)	6.717 (0.154)	0.560 (0.011)▲	0.232 (0.005)▼
	Listwise Naive	1.599 (0.023)	3.485 (0.042)	4.845 (0.072)	7.443 (0.100)	0.596 (0.005)	-
	Listwise IPS	1.641 (0.020)▲	3.545 (0.012)▲	4.920 (0.020)▲	7.522 (0.026)▲	0.602 (0.002)▲	-
	Listwise DLA	1.621 (0.026)	3.525 (0.074)▲	4.891 (0.101)▲	7.512 (0.160)▲	0.599 (0.006)▲	-
	LambdaRank Naive	1.750 (0.035)	3.717 (0.068)	5.125 (0.083)	7.810 (0.111)	0.613 (0.007)	-
	LambdaRank PairD	1.723 (0.018)▼	3.683 (0.034)▼	5.089 (0.042)▼	7.761 (0.046)▼	0.608 (0.004)▼	-

8 CONCLUSION

In this work, we carefully revisited and extended the experiments conducted by Zou et al. [65] on the recently released Baidu-ULTR dataset. As the largest publicly available dataset comprising both click logs from a major search engine and expert annotations, this dataset constitutes a rare opportunity to assess the progress of click-based learning-to-rank, and especially unbiased learning-to-rank.

Our main conclusion, however, is that while we have observed substantial improvements relating to the choice of query-document representations and ranking loss (e.g., pointwise or listwise), conventional unbiased learning-to-rank techniques do not bring clear improvements in ranking performance on the Baidu-ULTR dataset. Our findings confirm the work by Zou et al. [65], even though we used different dataset features and preprocessing, model implementations, and performed extensive hyperparameter tuning. Further, we observed a divergence between click-based and annotations-based objectives, as all click-based approaches were outperformed on annotation-based metrics by simple baselines such as BM25, even when BM25 was a model input. We believe these results call for more research into the conditions for success and failure of unbiased learning-to-rank and click-based approaches as a whole.

ACKNOWLEDGMENTS

We are grateful to Lixin Zou for his valuable insights and clarifications on the original Baidu work, as well as to Jiaxin Mao and Zechun Niu for sharing their learnings from NTCIR 17.

This research was supported by the Mercury Machine Learning Lab, created by TU Delft, the University of Amsterdam, and funded by Booking.com. Maarten de Rijke was supported by the Dutch Research Council (NWO), project nrs 024.004.022, NWA.1389.20.-183, and KICH3.LTP.20.006, and the European Union's Horizon Europe program under grant agreement No 101070212. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] Aman Agarwal, Xuanhui Wang, Cheng Li, Michael Bendersky, and Marc Najork. 2019. Addressing Trust Bias for Unbiased Learning-to-Rank. In *The World Wide Web Conference (WWW)*.

[2] 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 (WSDM)*.

[3] 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 (SIGIR)*.

[4] 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 (SIGIR)*.

[5] Qingyao Ai, Tao Yang, Huazheng Wang, and Jiaxin Mao. 2021. Unbiased Learning to Rank: Online or Offline? *ACM Transactions on Information Systems (TOIS)* 39, 2 (2021).

[6] Jessa Bekker, Pieter Robberechts, and Jesse Davis. 2019. Beyond the Selected Completely at Random Assumption for Learning from Positive and Unlabeled Data. In *Machine Learning and Knowledge Discovery in Databases: European Conference (ECML PKDD)*.

[7] 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 (WWW)*.

[8] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. 2018. JAX: composable transformations of Python+NumPy programs. http://github.com/google/jax

[9] Sebastian Bruch, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2019. An Analysis of the Softmax Cross Entropy Loss for Learning-to-Rank with Binary Relevance. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR)*.

[10] Christopher J. C. Burges, Robert Ragno, and Quoc Viet Le. 2006. Learning to Rank with Nonsmooth Cost Functions. In *Proceedings of the 19th International Conference on Neural Information Processing Systems (NIPS)*.

[11] Olivier Chapelle and Yi Chang. 2011. Yahoo! Learning to Rank Challenge Overview. *Journal of Machine Learning Research (JMLR)* 14 (2011), 1–24.

[12] Olivier Chapelle and Ya Zhang. 2009. A Dynamic Bayesian Network Click Model for Web Search Ranking. In *The World Wide Web Conference (WWW)*.

[13] Jia Chen, Haitao Li, Weihang Su, Qingyao Ai, and Yiqun Liu. 2023. THUIR at WSDM Cup 2023 Task 1: Unbiased Learning to Rank. In *Proceedings of The Sixteen ACM International Conference on Web Search and Data Mining (WSDM)*.

[14] 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 (WSDM)*.

[15] Mouxiang Chen, Chenghao Liu, Zemin Liu, Zhuo Li, and Jianling Sun. 2024. Identifiability Matters: Revealing the Hidden Recoverable Condition in Unbiased Learning to Rank. arXiv:2309.15560 [cs.IR]

[16] Stanley F. Chen and Joshua Goodman. 1999. An Empirical Study of Smoothing Techniques for Language Modeling. *Computer Speech & Language* 13, 4 (1999), 359–394.

[17] Xiaoshu Chen, Xiangsheng Li, Kunliang Wei, Bin Hu, Lei Jiang, Zeqian Huang, and Zhanhui Kang. 2023. Multi-Feature Integration for Perception-Dependent Examination-Bias Estimation. In *Proceedings of The Sixteen ACM International Conference on Web Search and Data Mining (WSDM)*.

[18] Aleksandr Chuklin, Ilya Markov, and Maarten de Rijke. 2015. *Click Models for Web Search*. Morgan & Claypool. https://doi.org/10.2200/S00654ED1V01Y201507ICR043

[19] 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 (WSDM)*.

[20] Domenico Dato, Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Nicola Tonellotto, and Rossano Venturini. 2016. Fast Ranking with Additive Ensembles of Oblivious and Non-Oblivious Regression Trees. *ACM Transactions on Information Systems (TOIS)* 35, 2, Article 15 (2016).

[21] Domenico Dato, Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, and Nicola Tonellotto. 2022. The Istella22 Dataset: Bridging Traditional and Neural Learning to Rank Evaluation. In *The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.

[22] Romain Deffayet, Philipp Hager, Jean-Michel Renders, and Maarten de Rijke. 2023. An Offline Metric for the Debiasedness of Click Models. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.

[23] Romain Deffayet, Jean-Michel Renders, and Maarten De Rijke. 2023. Evaluating the Robustness of Click Models to Policy Distributional Shift. *ACM Transactions on Information Systems (TOIS)* 41, 4, Article 84 (2023).

[24] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)* 39, 1 (1977), 1–38.

[25] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL]

[26] Georges E. Dupret and Benjamin Piwowarski. 2008. A User Browsing Model to Predict Search Engine Click Data from Past Observations.. In *International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.

[27] Varun Godbole, George E. Dahl, Justin Gilmer, Christopher J. Shallue, and Zachary Nado. 2023. Deep Learning Tuning Playbook. http://github.com/google-research/tuning_playbook Version 1.0.

[28] 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 (RecSys)*.

[29] Shashank Gupta, Philipp Hager, Jin Huang, Ali Vardasbi, and Harrie Oosterhuis. 2024. Unbiased Learning to Rank: On Recent Advances and Practical Applications. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining (WSDM)*.

[30] Shashank Gupta, Harrie Oosterhuis, and Maarten de Rijke. 2023. Safe Deployment for Counterfactual Learning to Rank with Exposure-Based Risk Minimization. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.

[31] 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 (WWW)*.

[32] Rolf Jagerman, Xuanhui Wang, Honglei Zhuang, Zhen Qin, Michael Bendersky, and Marc Najork. 2022. Rax: Composable Learning-to-Rank Using JAX. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*.

[33] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. 2005. Accurately Interpreting Clickthrough Data as Implicit Feedback. In *International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.

[34] 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 (WSDM)*.

[35] Haitao Li, Jia Chen, Weihang Su, Qingyao Ai, and Yiqun Liu. 2023. Towards Better Web Search Performance: Pre-training, Fine-tuning and Learning to Rank. In *Proceedings of The Sixteen ACM International Conference on Web Search and Data Mining (WSDM)*.

[36] Xiangsheng Li, Xiaoshu Chen, Kunliang Wei, Bin Hu, Lei Jiang, Zeqian Huang, and Zhanhui Kang. 2023. Pretraining De-Biased Language Model with Large-scale Click Logs for Document Ranking. In *Proceedings of The Sixteen ACM International Conference on Web Search and Data Mining (WSDM)*.

[37] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. (2019). arXiv:1711.05101 [cs.LG]

[38] Zechun Niu, Jiaxin Mao, Qingyao Ai, Lixin Zou, Shuaiqiang Wang, and Dawei Yin. 2023. Overview of the NTCIR-17 Unbiased Learning to Rank Evaluation 2 (ULTRE-2) Task. In *The 17th Round of NII Testbeds and Community for Information Access Research (NTCIR)*.

[39] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-Stage Document Ranking with BERT. arXiv:1910.14424 [cs.IR]

[40] Harrie Oosterhuis. 2022. Reaching the End of Unbiasedness: Uncovering Implicit Limitations of Click-Based Learning to Rank. In *Proceedings of the 2022 ACM SIGIR International Conference on Theory of Information Retrieval (SIGIR)*.

[41] Harrie Oosterhuis and Maarten de Rijke. 2021. Unifying Online and Counterfactual Learning to Rank: A Novel Counterfactual Estimator That Effectively Utilizes Online Interventions. In *International Conference on Web Search and Data Mining (WSDM)*.

[42] Rama Kumar Pasumarthi, Sebastian Bruch, Xuanhui Wang, Cheng Li, Michael Bendersky, Marc Najork, Jan Pfeifer, Nadav Golbandi, Rohan Anil, and Stephan Wolf. 2019. TF-Ranking: Scalable TensorFlow Library for Learning-to-Rank. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*.

[43] Ronak Pradeep, Rodrigo Nogueira, and Jimmy Lin. 2021. The Expando-Mono-Duo Design Pattern for Text Ranking with Pretrained Sequence-to-Sequence Models. arXiv:2101.05667 [cs.IR]

[44] Tao Qin and Tie-Yan Liu. 2013. Introducing LETOR 4.0 Datasets. arXiv:1306.2597 [cs.IR]

[45] Zhen Qin, Le Yan, Honglei Zhuang, Yi Tay, Rama Kumar Pasumarthi, Xuanhui Wang, Mike Bendersky, and Marc Najork. 2021. Are Neural Rankers still Outperformed by Gradient Boosted Decision Trees?. In *International Conference on Learning Representations (ICLR)*.

[46] 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 (WWW)*.

[47] Stephen Robertson. 2004. Understanding Inverse Document Frequency: On Theoretical Arguments for IDF. *Journal of documentation* 60, 5 (2004), 503–520.
[48] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at TREC-3. In *Proceedings of The Third Text REtrieval Conference, TREC (NIST Special Publication, Vol. 500-225)*. 109–126.
[49] Yuta Saito, Suguru Yaginuma, Yuta Nishino, Hayato Sakata, and Kazuhide Nakata. 2020. Unbiased Recommender Learning from Missing-Not-At-Random Implicit Feedback. In *International Conference on Web Search and Data Mining (WSDM)*.
[50] Mark Sanderson et al. 2010. Test Collection Based Evaluation of Information Retrieval. *Foundations and Trends in Information Retrieval* 4 (2010), 247–375.
[51] Fatemeh Sarvi, Ali Vardasbi, Mohammad Aliannejadi, Sebastian Schelter, and Maarten de Rijke. 2023. On the Impact of Outlier Bias on User Clicks. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.
[52] Xiaojie Sun, Lulu Yu, Yiting Wang, Keping Bi, and Jiafeng Guo. 2023. Ensemble Ranking Model with Multiple Pretraining Strategies for Web Search. In *Proceedings of The Sixteen ACM International Conference on Web Search and Data Mining (WSDM)*.
[53] Anh Tran, Tao Yang, and Qingyao Ai. 2021. ULTRA: An Unbiased Learning To Rank Algorithm Toolbox. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management (CIKM)*.
[54] Ali Vardasbi, Maarten de Rijke, and Ilya Markov. 2020. Cascade Model-Based Propensity Estimation for Counterfactual Learning to Rank. In *International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.
[55] Ali Vardasbi, Harrie Oosterhuis, and Maarten de Rijke. 2020. When Inverse Propensity Scoring Does Not Work: Affine Corrections for Unbiased Learning to Rank. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM)*.
[56] 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)*.
[57] 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 (WSDM)*.
[58] Le Yan, Zhen Qin, Honglei Zhuang, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2022. Revisiting Two-tower Models for Unbiased Learning to Rank. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.
[59] Lulu Yu, Keping Bi, Jiafeng Guo, and Xueqi Cheng. 2023. CIR at the NTCIR-17 ULTRE-2 Task. In *The 17th Round of NII Testbeds and Community for Information Access Research (NTCIR)*.
[60] Lulu Yu, Yiting Wang, Xiaojie Sun, Keping Bi, and Jiafeng Guo. 2023. Feature-Enhanced Network with Hybrid Debiasing Strategies for Unbiased Learning to Rank. In *Proceedings of The Sixteen ACM International Conference on Web Search and Data Mining (WSDM)*.
[61] Yunan Zhang, Le Yan, Zhen Qin, Honglei Zhuang, Jiaming Shen, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2023. Towards Disentangling Relevance and Bias in Unbiased Learning to Rank. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*.
[62] Yurou Zhao, Zechun Niu, Feng Wang, Jiaxin Mao, Qingyao Ai, Yang Tao, Junqi Zhang, and Yiqun Liu. 2022. Overview of the NTCIR-16 Unbiased Learning to Rank Evaluation (ULTRE) Task. In *The 16th Round of NII Testbeds and Community for Information Access Research (NTCIR)*.
[63] 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 (RecSys)*.
[64] 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 (WebConf)*.
[65] Lixin Zou, Haitao Mao, Xiaokai Chu, Jiliang Tang, Wenwen Ye, Shuaiqiang Wang, and Dawei Yin. 2022. A Large Scale Search Dataset for Unbiased Learning to Rank. In *Advances in Neural Information Processing Systems (NeurIPS)*.
[66] Lixin Zou, Haitao Mao, Xiaokai Chu, Wenwen Ye, Changying Hao, Shuaiqiang Wang, Dawei Yin, Jiliang Tang, Aixin Sun, and Ce Zhang. 2023. Pre-training for Web Search. https://aistudio.baidu.com/competition/detail/536/0/introduction
[67] Lixin Zou, Haitao Mao, Xiaokai Chu, Wenwen Ye, Changying Hao, Shuaiqiang Wang, Dawei Yin, Jiliang Tang, Aixin Sun, and Ce Zhang. 2023. Unbiased Learning for Web Search. https://aistudio.baidu.com/competition/detail/534/0/introduction