NTCIR 17 Conference: Proceedings of the 17th NTCIR Conference on Evaluation of Information Access Technologies, December 12-15, 2023, Tokyo, Japan
DOI: https://doi.org/10.20736/0002001336

CIR at the NTCIR-17 ULTRE-2 Task

Lulu Yu
CAS Key Lab of Network Data
Science and Technology, ICT, CAS
University of Chinese Academy of Sciences
Beijing, China
yululu23s@ict.ac.cn

Keping Bi
CAS Key Lab of Network Data
Science and Technology, ICT, CAS
University of Chinese Academy of Sciences
Beijing, China
bikeping@ict.ac.cn

Jiafeng Guo
CAS Key Lab of Network Data
Science and Technology, ICT, CAS
University of Chinese Academy of Sciences
Beijing, China
guojiafeng@ict.ac.cn

Xueqi Cheng
CAS Key Lab of Network Data
Science and Technology, ICT, CAS
University of Chinese Academy of Sciences
Beijing, China
cxq@ict.ac.cn

ABSTRACT

The **C**hinese academy of sciences **I**nformation **R**etrieval team (CIR) has participated in the NTCIR-17 ULTRE-2 task. This paper describes our approaches and reports our results on the ULTRE-2 task. We recognize the issue of false negatives in the Baidu search data in this competition is very severe, much more severe than position bias. Hence, we adopt the Dual Learning Algorithm (DLA) to address the position bias and use it as an auxiliary model to study how to alleviate the false negative issue. We approach the problem from two perspectives: 1) correcting the labels for non-clicked items by a relevance judgment model trained from DLA, and learn a new ranker that is initialized from DLA; 2) including random documents as true negatives and documents that have partial matching as hard negatives. Both methods can enhance the model performance and our best method has achieved nDCG@10 of 0.5355, which is 2.66% better than the best score from the organizer.

KEYWORDS

Unbiased Learning to Rank, Position Bias, Selection Bias, Negative Sampling

TEAM NAME

CIR

SUBTASKS

Effectiveness Evaluation Subtask (Chinese), Robustness Investigation Subtask (Chinese)

1 INTRODUCTION

Learning to Rank (LTR) is a crucial component in various real-world systems, especially search engines. While human annotations provide accurate relevance information, they are expensive to obtain. User clicks in the search logs can serve as an indication of document relevance and are cheap to collect. This makes them favorable for modern ranking models, which typically have numerous parameters, and require a substantial amount of training data. However, it is obvious that severe bias and noise exist in the click data e.g. position bias. Thus, extensive research on unbiased learning to rank (ULTR) [1, 2, 11] has been conducted to mitigate different biases.

There is a lack of public datasets that have various types of bias from real-world user interactions, so existing ULTR methods have been primarily studied on synthetic datasets and achieved promising results. For instance, Joachims et al. [7] introduced a randomization experiment to estimate position bias and used Inverse Propensity Weighting (IPW) to debias click signals. To eliminate the need for result randomization, Ai et al. [1] proposed the Dual Learning Algorithm (DLA) to jointly learn an unbiased ranker and an unbiased propensity model. However, it remains uncertain whether these methods would be effective when applied to real-world datasets. Given this observation, the ULTRE-2 task encourages participants to explore ULTR approaches to alleviate various types of biases in real user clicks during training, and achieve better ranking performance based on the search dataset collected from the search logs from the largest Chinese search engine, Baidu.

The biases in real-world search datasets are much more complex than the position bias in the simulated clicked data according to the position model. Therefore, the existing popular ULTR methods that handle position bias only do not work effectively on the Baidu search dataset as expected [12]. We attribute this to that UTLR methods mostly address the position bias on the clicked documents and treat non-clicked items as irrelevant while most non-clicked items in the first search result page (SERP) are relevant. The items are not clicked because they are not examined rather than not being relevant. Search engines typically provide high-quality first SERPs where most items are relevant to the query, especially for head queries whose click data is abundant. The false negative issues can be much more severe than the position bias in real-world search data. Hence, Instead of focusing on the position bias, in our solution, we mainly focus on how to address the false negative issue. We simply adopt the state-of-the-art ULTR method DLA [1] to alleviate position bias as it has shown superior performance compared to other methods [13], and use it as the foundation model to address the issue of false negatives.

We probe the issue from two perspectives. The first is selection bias that represents the bias towards the documents that have zero examination propensities. Usually, selection bias [3] is used to describe the case that some documents are not selected by the ranker on the result page and thus have no chance to be examined. In this task, documents at lower positions have similar issues, so

we approach the issue from the perspective of selection bias and aim to correct the labels of the non-clicked items. The second one is negative sampling. ULTR models are usually trained only using the results on the SERPs [1, 11]. However, we know that there are many false negatives in this data which would confuse the model. Introducing true negatives for model training can guide the model to learn to differentiate relevance.

Concretely, for selection bias, we employ the model trained with DLA as the relevance judgment model to generate labels for non-clicked items. Then, based on the updated labels, we train another model that is initiated from the judgment model to learn true relevance. We refer to this approach as a Dual Learning Algorithm with Label Correction (DLA-LC). For negative sampling, we sample random documents as true irrelevant documents and also involve other documents that are partially relevant to act as hard negatives. Our experimental results show that both approaches can boost the model performance.

Table 1: A summary of the notations used in this paper.

Notation	Meaning
q	one query
d	one candidate item
π_q	the candidate document list of query q
o, r, c	binary random variables indicating whether an item d is observed, perceived as relevant, and clicked by a user

2 RELATED WORK

To leverage biased user clicks for optimizing an unbiased ranking model in learning-to-rank systems, considerable research has been tailored for unbiased learning to rank (ULTR) to alleviate the bias present in user clicks. There are two streams of ULTR methodologies. One trend depends on click modeling [5, 6], which makes assumptions about user browsing behaviors and then models the examination probability. By maximizing the likelihood of the observed data, they can accurately infer relevance from user clicks. For example, Craswell et al. [6] designed a Position-Based Model that assumes users will click on a document if and only if they have examined it and considered it relevant. However, this method is based on statistics requiring multiple occurrences of the same query-document pair for reliable inference, which is challenging for long-tail queries and sparse query systems e.g. personal search. The other trend derives from counterfactual learning, which treats bias as a counterfactual factor and debiases user clicks via inverse propensity weighting [2]. The latest work is that Ai et al. [1] presents a Dual Learning Algorithm that treats training an unbiased ranking model and propensity model as a dual problem and optimizes each other together. Most ULTR approaches experiment on synthetic datasets and perform effectively, but whether they can maintain the performance on real user clicks still remains uncertain. In this competition, we propose a DLA-LC to make DLA more suitable for real-world datasets.

3 UNBIASED LEARNING

In this section, we first introduce our methods to address the position bias and false negative issues and we also describe one upper bound model training we use.

3.1 Alleviating Position Bias with IPW

To alleviate the position bias, a series of IPW-based methods [2] have been proposed and shown to be effective. Notably, recent work [9, 13] has shown that DLA [1] has achieved the state-of-the-art performance on the real-world dataset. DLA jointly learns a propensity model and ranking model to alleviate position bias that can dynamically learn the examination propensities of each position without the need for result randomization [7]. Thus, we first adopt DLA to eliminate the position bias. The loss function we utilize is as follows:

$$L_{ranking} = -\sum_{x\in\pi_q, c_q^x=1} \frac{P(o_q^1=1|\pi_q)}{P(o_q^x=1|\pi_q)} \cdot \log \frac{e^{f(x)}}{\sum_{z\in\pi_q} e^{f(z)}}, \quad (1)$$

$$L_{observation} = -\sum_{x\in\pi_q, c_q^x=1} \frac{P(r_q^1=1|\pi_q)}{P(r_q^x=1|\pi_q)} \cdot \log \frac{e^{g(x)}}{\sum_{z\in\pi_q} e^{g(z)}}, \quad (2)$$

where $f(x)$ and $g(x)$ denote the output of the ranking model and propensity model respectively and the superscript 1 represents the item in the first place. The probabilities of observation and relevance are computed as follows:

$$P(r_q^x=1|\pi_q) = \frac{e^{f(x)}}{\sum_{z\in\pi_q} e^{f(z)}}, \quad (3)$$

$$P(o_q^x=1|\pi_q) = \frac{e^{g(x)}}{\sum_{z\in\pi_q} e^{g(z)}}. \quad (4)$$

3.2 Alleviating Selection Bias with Label Correction

Although position bias can be alleviated by ULTR methods like DLA, it is important to acknowledge that other biases also exist in the click data. Specifically, position bias only accounts for clicked items as non-clicked items are all labeled as 0. However, not all non-clicked items are irrelevant. The documents are not clicked probably because users did not examine them or they are similar to the clicked items. This issue is especially severe when the ranker is effective and most results on the first search result page (SERP) are relevant. Labeling all non-clicked items as 0 could confuse the model in relevance modeling. So, it would be advantageous to adjust the documents labeled as 0 to reasonable non-zero values.

Therefore, we utilize the model trained with DLA as an auxiliary model to generate labels for non-clicked items. We attempt various reasonable strategies to transform the output of this auxiliary model into new labels as shown in Equation(5). We merely list two representative modes: directly regard the output after the sigmoid function as their labels or set the labels of those whose output is larger than the minimum output of clicked items as one. Here, l_{ij}, a_{ij} means the label, auxiliary model output of the i^{th} query's

j^{th} item with $c_{ij} = 0$,

$$l_{ij} = \begin{cases} sig : sigmoid(a_{ij}) \\ min : 1 if a_{ij} \geq min(a_{ik}), c_{ik} = 1 \end{cases} . \quad (5)$$

3.3 Negative Sampling

Considering a large proportion of non-click items are false negatives, we try to rebuild the ranking lists to reduce the number of false negatives by adding random and hard negatives and replacing original non-click candidate items with random negatives. In this approach, we do not consider position bias and instead treat the original clicked item as relevant, while considering the newly generated negatives as irrelevant.

With the help of Galago [1], we pre-generate substantial hard negatives for each query based on BM25 scores. To simulate a multi-level irrelevance among candidate items, these negatives are then sampled from a Gaussian distribution assuming that the BM25 scores follow this distribution. Besides the most relevant (with the largest score) and irrelevant (with the smallest score) hard negatives are also included. We adjust the number of hard negatives from 50 to 90.

We have devised two schemes: the **"click-only"** scheme replaces all non-clicked items with random negatives, while the **"last-click"** scheme replaces items after the last clicked item with random negatives. Moreover, we employ random negatives to ensure that the candidate list for each query is of equal length. The loss function we use is as follows. The π'_q indicates reconstructed ranking lists to a query q,

$$L_{click-only}/L_{last-click} = -\sum_{x \in \pi'_q} log \frac{e^{f(x)}}{\sum_{z \in \pi'_q} e^{f(z)}}. \quad (6)$$

3.4 GBDT Training

In addition to the above methods, we also train a Gradient Boosting Decision Tree (GBDT) model using the validation set with the help of Light Gradient Boosting Machine (LightGBM) [8] with LambdaRank objective. Concretely, 80% of the total validation set for training and 20% left for validating, split by query id column. By training directly with human annotation labels, which are unbiased, we aim to establish an upper bound for model performance given the same input features. Moreover, the scores obtained from the unbiased learning model can be aggregated with other signals in LightGBM. Consequently, we build two models using two settings of input features, with one including the best model score obtained from previous training with clicks.

4 EXPERIMENTS

We present a detailed explanation of our experimental setup and then summarize the representative results in this section.

4.1 Experimental Setup

Methods for comparison. We compare the performance of the following methods:

- **DLA:** The Dual Learning Algorithm [1] simultaneously learns an unbiased ranking model and a propensity model.
- **Scratch-DLA-LC (sig):** Retrain a DLA model using the modified click labels from the first mode (sig) in Equation5.
- **Aux-DLA-LC (sig):** Given a DLA model trained with click labels, continue training this DLA model utilizing the modified click labels from the first mode (sig) in Equation5.
- **Scratch-DLA-LC (min):** Retrain a DLA model using the modified click labels from the second mode (min) in Equation5.
- **Aux-DLA-LC (min):** Given a DLA model trained with click labels, continue training this DLA model utilizing the modified click labels from the second mode (min) in Equation5.
- **lgbBase:** This GBDT model uses human annotation labels, and has the same input features as the unbiased neural ranking models.
- **lgbAdd:** Except for the addition of the best model score to the input features, everything is the same as the lgbBase model.

Experimental implementation. According to [10], the ULTRE-2 organizers provide us with three types of features, traditional word matching features (e.g. TF-IDF, BM25), as well as a score feature and 768 dimensions' embedding features derived from the pre-trained BERT model. We primarily choose the first two types as the input features, i.e. 14 features in total.

Only results in the initial ranking lists have their associated features. This means that we do not have access to the features of new query-result pairs using the given feature data alone. To address this limitation, we extract another 24 heuristic-based features such as TF, TF-IDF, BM25, and language model with three types of smoothing methods for title, abstract, and the combined fields, the same as features in [12] to calculate the features for new query-negative pairs.

When training with click data, for each query, we fix the length of its ranking list as 10. As for negative sampling, at first, the reconstructed ranking list consists of the items we reserve in the initial ranking lists depending on different strategies mentioned in Section 3.3 and several random negatives for aligning the length, and on the basis of it, append additional random and hard negatives. We use the total validation set (referred to as total valid) for validating when training the neural network model while for the GBDT model, we use 20% of the validation set (referred to as 20% valid) to select the best model. We present the nDCG@10 on both the corresponding validation and test sets.

To implement our models, we utilize the ULTRA framework[2] as the basis. Our ranking model is a deep neural network (DNN) as well. In our experiment, we start by projecting the above input features to a higher dimension, specifically 64. These projected features are then fed into the DNN with hidden layer dimensions as 32,16 and 8. For optimization, we employ the AdamW optimizer and fine-tune the learning rates in the range of 2e-6 to 1e-5.

4.2 Experimental Results

Initially, we evaluate the nDCG@10 of every individual feature mentioned above on the total validation set. Among these features,

[1] https://lemur.sourceforge.io/documentation/Galago%20Documentation.html

[2] https://github.com/ULTR-Community/ULTRA_pytorch

Table 2: The model performance on both the hidden test set and the validation set. The total validation set (referred to as total valid) was used to evaluate the performance of neural network models, while 20% of the validation set (referred to as 20% valid) was used to assess the performance of GBDT models, which were trained using 80% of the validation set.

Model	nDCG@10			DCG@10
	test	total valid	20% valid	test
Scratch-DLA-LC (sig)	**0.5355**	**0.5019**	/	11.4538
Aux-DLA-LC (sig)	0.5326	0.5015	/	11.3898
Scratch-DLA-LC (min)	unk	0.4816	/	unk
Aux-DLA-LC (min)	unk	0.4947	/	unk
DLA	0.5247	0.4920	/	11.2031
lgbBase	0.5350	/	0.5003	11.4794
lgbBaseAdd	0.5333	/	0.5021	11.4616

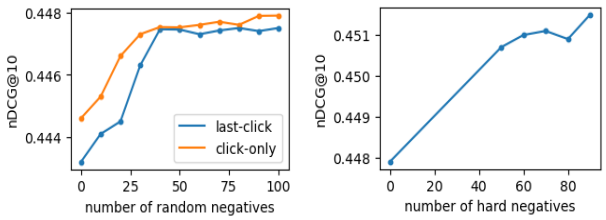

Figure 1: Performance curves of two schemes ("click-only" and "last-click") w.r.t. the number of random and hard negatives. (a) Performance curves of two schemes w.r.t. the number of random negatives. (b) The Performance curve of the "click-only" scheme w.r.t. the number of hard negatives.

the best nDCG@10 from given and self-generated traditional features are 0.4608 and 0.4625 respectively, and the pre-trained score stands out with an impressive nDCG@10 of 0.4767, demonstrating its superior significance in model training compared to the traditional features.

Nevertheless, it's a pity we did not have time to leverage the pre-trained model available on Github[3] to obtain the pre-trained score for each query-negative pair. As a result, we solely relied on traditional features as input, making it challenging to compare the effectiveness of model training under negative sampling with other models that include the pre-trained score in their input features. Despite this limitation, our experimental results showed an improvement in the model's performance when using traditional features as input. Consequently, we decided to showcase the nDCG@10 on the total valid to demonstrate the approach's effectiveness.

The nDCG@10 results are shown in Table 2. *unk* denotes we did not submit it to the leaderboard. Besides nDCG@10, we include DCG@10 to compare the results with [4] as the pre-trained score derived from the pre-trained model in it. In [4], the best score achieved 10.25, while in [10] the model without debiasing achieved 11.26. This verifies the necessity of introducing some proper high-quality traditional features into the model.

[3]https://github.com/lixsh6/Tencent_wsdm_cup2023/tree/main/pytorch_unbias

4.2.1 Results on Label Correction and GBDT Training. From Table 2, we can draw the following conclusions: (1) using already trained DLA as the auxiliary model, combined with both variants can significantly enhance the performance of the model trained from this auxiliary model. And between the two variants, the sigmoid function yields better results, (2) when comparing models with the same input features, the lgbBase model still outperforms the simple DLA model trained with clicks. However, it is not as effective as our proposed Scratch-DLA-LC (sig) model, (3) the performance of the neural network models on the validation set appears to be consistent with their performance on the test set. Thus we could determine the best model based on nDCG@10 on the validation. Nonetheless, there exists inconsistency in the performance of the GBDT model, suggesting potential overfitting issues.

4.2.2 Results on Negative Sampling. Results of the negative sampling technique on the total validation set are shown in Figure **??**. Comparing the two strategies mentioned in Section 3.3, namely the click-only strategy and the last-click strategy, it is observed that the click-only strategy performs slightly better. Therefore, we choose the click-only strategy as the basis strategy and focus on increasing the number of hard negative samples. From Figure 1(b), we can observe that by adjusting the number of negative samples, we find that including both random and hard negatives is beneficial, although the improvement is gradual.

5 CONCLUSIONS

In this paper, we present our solution for the NTCIR-17 ULTRE-2 task, focusing on the false negative issue. We propose two approaches to tackle this issue. Firstly, we explore the aspect of selection bias and introduce several methods of label correction. Through experiments, we find that our approach, DLA-LC, outperforms the basic DLA model. Additionally, we investigate the use of negative sampling to reconstruct the given ranking lists and reduce the number of false negatives. The nDCG@10 on the validation set indicates that this approach is effective in improving performance. Although we were unable to leverage the important pre-trained score in this experiment, we plan to explore this further in future research to determine the extent of improvement it can bring.

REFERENCES

[1] 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 & development in information retrieval*. 385–394.

[2] 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), 1–29.

[3] Yinqiong Cai, Jiafeng Guo, Yixing Fan, Qingyao Ai, Ruqing Zhang, and Xueqi Cheng. 2022. Hard Negatives or False Negatives: Correcting Pooling Bias in Training Neural Ranking Models. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 118–127.

[4] 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. *arXiv preprint arXiv:2302.13756* (2023).

[5] Aleksandr Chuklin, Ilya Markov, and Maarten De Rijke. 2022. *Click models for web search*. Springer Nature.

[6] 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.

[7] 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.

[8] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems* 30 (2017).

[9] Dan Luo, Lixin Zou, Qingyao Ai, Zhiyu Chen, Dawei Yin, and Brian D Davison. 2023. Model-based unbiased learning to rank. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*. 895–903.

[10] Zechun Niu, Jiaxin Mao, Qingyao Ai, Lixin Zou, Shuaiqiang Wang, and Dawei Yin. 2023. Overview of the NTCIR-17 Unbiased Learning to RankEvaluation 2 (ULTRE-2) Task. In *Proceedings of the NTCIR-17 Conference*.

[11] 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*. 115–124.

[12] Lulu Yu, Yiting Wang, Xiaojie Sun, Keping Bi, and Jiafeng Guo. 2023. Feature-Enhanced Network with Hybrid Debiasing Strategies for Unbiased Learning to Rank. *arXiv preprint arXiv:2302.07530* (2023).

[13] 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. *Advances in Neural Information Processing Systems* 35 (2022), 1127–1139.