ULTRE framework: a framework for Unbiased Learning to Rank Evaluation based on simulation of user behavior

Yurou Zhao[1], Jiaxin Mao[2] and Qingyao Ai[3]

[1]Renmin University of China, China
[2]Renmin University of China, China
[3]University of Utah, USA

Abstract
Unbiased learning to rank (ULTR) with biased user behavior data has received considerable attention in the IR community. However, how to properly evaluate and compare different ULTR approaches has not been systematically investigated and there is no shared task or benchmark that is specifically developed for ULTR. In this paper, we propose the Unbiased Learning to Rank Evaluation(ULTRE) framework. The proposed framework utilizes multiple click models in generating simulated click logs and supports the evaluation of both the offline, counterfactual and the online, bandit-based ULTR models. Our experiments show that the ULTRE framework are effective in click simulation and comparing different ULTR models. The ULTRE framework will be used in the Unbiased Learning to Rank Evaluation Task (ULTRE), a pilot task in NTCIR 16.

Keywords
Unbiased Learning to Rank, Evaluation, Click Model, Click Simulation

1. INTRODUCTION

Interest in Learning to Rank (LTR) approaches that learn from user interactions has increased recently as users' interaction with search systems can reflect their implicit relevance feedback for the search results. Though collecting user clicks is much less costly and more convenient than collecting expert annotations, user clicks contain different types of bias (such as position bias) and noise. Therefore,the unbiased learning to rank (ULTR) that aims at learning a ranking model from the noisy and biased user clicks has become a trending topic in IR. There are two main categories of algorithms for ULTR: 1) offline (counterfactual) LTR that learns an unbiased ranking model in an offline manner with batches of biased, historical click logs ([1, 2, 3]) 2) online ULTR which makes online interventions of ranking and extracting unbiased feedback or deriving unbiased gradient for modeling training ([4, 5]).

With a variety of models has been proposed for unbiased learning to rank, how to properly evaluate and compare different ULTR models still needs more research. Previous works on ULTR often use a simulation-based evaluation approach due to the lack of real search logs and online search systems. Such approach rely on predefined user behavior models and public available learning-to-rank datasets with item-level relevance judgments to simulate user clicks. Using the simulation-based approach, we can train ULTR models with the simulated clicks and then evaluate the models on test sets with expert annotations.

Though widely adopted, current evaluation approaches have some limitations. First, there are no standard evaluation settings or shared evaluation benchmarks for the ULTR community as existing studies on ULTR often rely on their own evaluation apparatus and adopt different assumptions in click simulation, making the experimental results reported in different papers incomparable. Second, most studies only use a single user behavior model to simulate clicks, which may not fully capture the diverse patterns of real user behavior. It may also introduce systematic biases into the comparison among ULTR models as the ULTR model that shares the same user behavior assumption with the click simulation model might be preferred by the evaluation ([6]).

To overcome the above limitations, we propose an unbiased learning to rank evaluation (ULTRE) framework. In this framework, we focus on extending and improving the click simulation phase in previous ULTR evaluation. Specifically, instead of using a single, over-simplified click model, we will use multiple user behavior models that trained and calibrated on real query log as several click simulators. Equipped with the click simulators, we further design two evaluation protocols for offline and online ULTR models, respectively.

In our empirical experiments, we implemented four different click simulators. After calibrations with real click logs, we incorporate them into our ULTRE framework. Then we compare several ULTR models under the framework to the verify the usefulness and effectiveness of our ULTRE framework.

We believe the ULTRE framework can serve as a shared benchmark and evaluation service for ULTR. It may also support an in-depth investigation of the simulation-based evaluation approach. The ULTRE framework will be used in the Unbiased Learning to Rank Evaluation Task (ULTRE), a pilot task in NTCIR 16.[1]

2. ULTRE FRAMEWORK

This section details the ULTRE framework, as shown in Figure 1. The whole process is made up of three stages: 1) generating simulated click logs 2) training the ULTR models with the simulated click logs of the training queries 3) evaluating the ULTR models with the relevance annotations in the validation and test set.

Stage 1 Simulation of clicks (step 1-5)
This stage is the key for the evaluation because the quality of simulated click may impact the performance of the trained ULTR model. It contains the following steps:

- `Step 1`: Train and calibrate four user behavior models (PBM, UBM, DCM, MCM) with real query logs.
- `Step 2`: Construct different click simulators based on models obtained in step 1.

Causality in Search and Recommendation (CSR) and Simulation of Information Retrieval Evaluation (Sim4IR) workshops at SIGIR, July 15, 2021
EMAIL: 13516173041@163.com (A. 1); maojiaxin@gmail.com (A. 2); aiqy@cs.utah.edu (A. 3)

[1]http://research.nii.ac.jp/ntcir/ntcir-16/

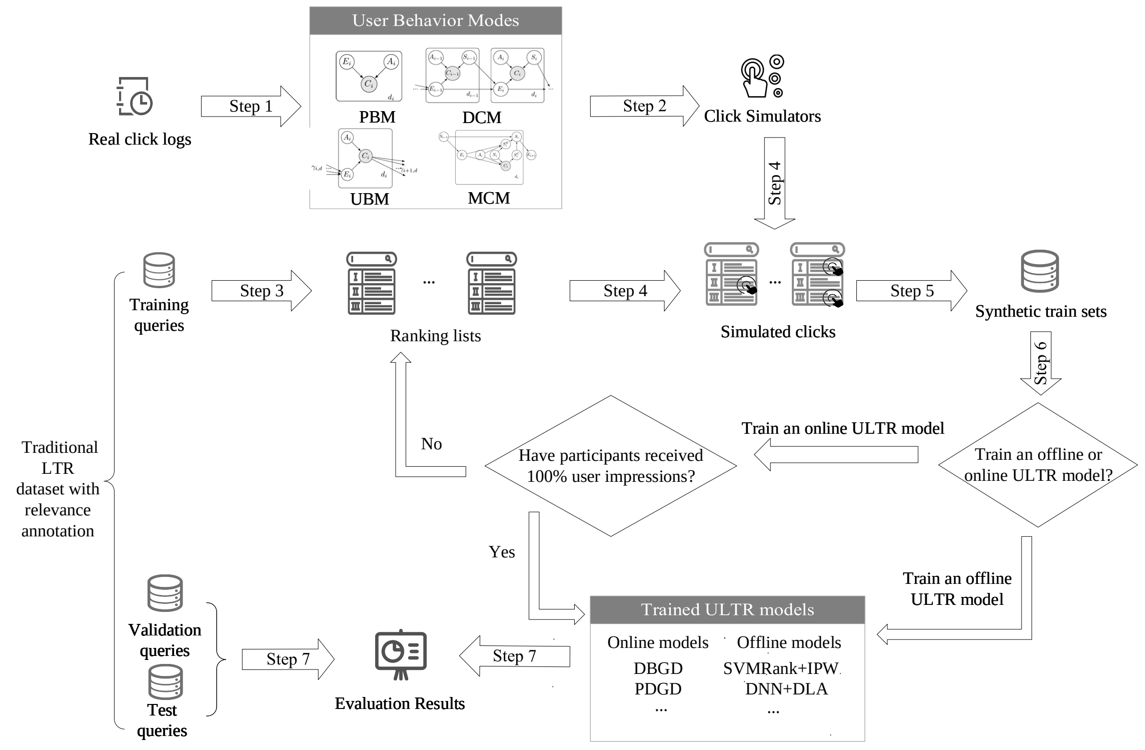


Figure 1: ULTRE framework

- `Step 3`: Collect the ranking lists for the training queries and the corresponding relevance annotations for the documents in the ranking lists. Depending on which class of ULTR models we want to train and evaluate, we will generate the ranking lists differently. Offline, counterfactual ULTR models, we will train a simple production ranker on a small proportion of the train set with relevance labels to generate ranking lists for all train queries. For the online ULTR models, the ranking lists will be generated by the online ULTR model that is being evaluated.
- `Step 4`: Use the click simulators defined in Step 1 and calibrated in Step 2 to generate simulated click logs for the ranking lists obtained in step 3.
- `Step 5`: Finally, collect the generated clicks and use them for the training of ULTR models. Because we construct four simulators, we will construct four synthetic train sets.

Stage 2 Training of ULTR models (step 6)
After generating the synthetic training set in Stage 1, we can use different synthetic train sets to train the ULTR models. It is worth mentioning that if the model is an online one, step 3-5 in stage 1 and stage 2 will be repeated multiple times to simulate the online learning procedure.

Stage 3 Evaluation of ULTR models (step 7)
Finally, in the last stage, we evaluate the trained ULTR models on the validation and test queries. We can compute some relevance-based evaluation metrics, such as nDCG, MAP, and MRR, with the relevance annotations in the validation and test set, to evaluate the ranking performance of the trained models.

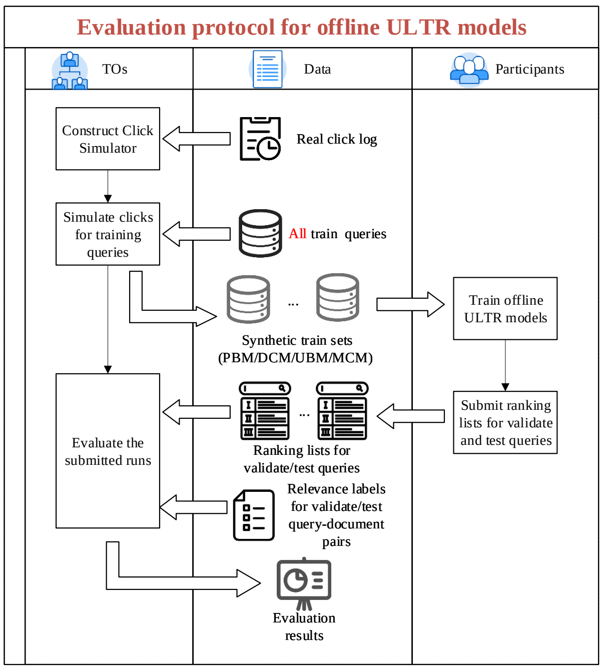
2.1. Evaluation protocol

Based on the ULTRE framework, we can provide a shared evaluation task and benchmark for the evaluation of different ULTR models. In this section, we develop evaluation protocols that describe how the task organizers of the shared ULTRE task (i.e. the TOs) interact with the participants of the shared task and work together to evaluate the ULTR models developed by the participants. Since there are two categories of ULTR models, we design two evaluation protocols, one for offline ULTR models, the other for online ones, respectively.

2.1.1. Evaluation protocol for offline ULTR models

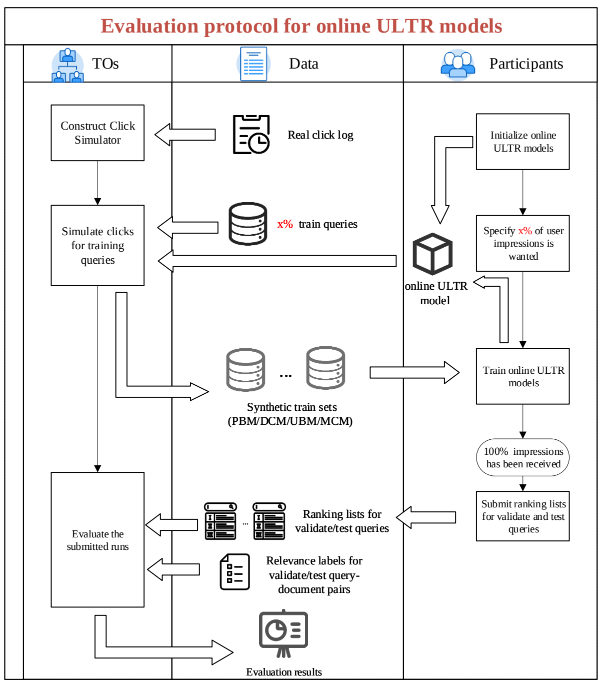
Figure 2 displays the steps in the evaluation protocol for offline ULTR models, and show what each role should do in each step. TOs represent the task organizers, and participants represent those who are willing to use the ULTRE framework to evaluate their ULTR models. The protocol consists three steps:

- `Step 1`: TOs construct click simulator based on real click log, and then simulate clicks for all queries in the train set. The participants then can use the simulated clicks to train their ULTR models. As four click simulator equipped with different user behavior models (PBM/UBM/DCM/MCM) will be used, TOs will produce four synthetic train sets for participants in this step.
- `Step 2`: Participants train their ULTR models on each synthetic train set respectively. Participants may have their preferred train set, as a result, they are allowed to only train the model on a single set. However, as each set is produced under some unique user behavior assumptions, the participants are strongly encouraged to train their models on all training sets. Such exploration can test the robustness of the ULTR model. After training the



Evaluation protocol for offline ULTR models
TOs
Data
Participants
Construct Click Simulator
Real click log
Simulate clicks for training queries
All train queries
...
Synthetic train sets (PBM/DCM/UBM/MCM)
Train offline ULTR models
Submit ranking lists for validate and test queries
Evaluate the submitted runs
Ranking lists for validate/test queries
Relevance labels for validate/test query-document pairs
Evaluation results

Figure 2: Evaluation protocol for offline ULTR models



Evaluation protocol for online ULTR models
TOs
Data
Participants
Construct Click Simulator
Real click log
Initialize online ULTR models
Simulate clicks for training queries
x% train queries
Specify x% of user impressions is wanted
online ULTR model
...
Synthetic train sets (PBM/DCM/UBM/MCM)
Train online ULTR models
100% impressions has been received
Submit ranking lists for validate and test queries
Evaluate the submitted runs
Ranking lists for validate/test queries
Relevance labels for validate/test query-document pairs
Evaluation results

Figure 3: Evaluation protocol for online ULTR models

ULTR models, the participants can submit the ranking lists (runs) for the validation and test queries. Each run submitted by the participants should only use the synthetic data generated by a single click simulator, so ideally, for each ULTR model, we expect the participant to submit four runs.

- `Step 3`: After receiving the runs submitted by participants, TOs evaluate the runs based on true relevance labels (i.e. expert annotation). Specifically, TOs will show the results on validation set on the leaderboard and release the official results in the final report.

2.1.2. Evaluation protocol for online ULTR models

As shown in Figure 3, the evaluation protocol for online ULTR models involves similar steps in the offline one. However, the main difference between them is that the participants can iteratively submit the ranking lists to the TOs to get simulated clicks and use them to update their ULTR models in an online process.

- `Step 1`: Participants submit the ranking lists for training queries generated by their own ULTR models and specify that they want to receive x% of user impressions.
- `Step 2`: TOs sample x% of all training queries according to the query frequency in the real log. Based on the ranking lists of those selected training queries submitted by the participant in step 1, TOs construct synthetic training sets following the same process in the step 1 of the evaluation protocol for offline ULTR models.
- `Step 3`: Participants update their models with the training data received in step 2.
- Repeat Step 1-Step 3 until participants receive 100% of impressions.
- `Step N`: Same as the final step in the evaluation protocol for offline ULTR models, TOs perform evaluation for the models on validation and test set.

2.2. Construct click simulators

This section provides more details about the process of constructing click simulator.

2.2.1. Choice of user behavior models

Compared with previous studies that only use a single click model, we use the following user behavior models:

- Position-Based Model (PBM)[7]: a click model that assumes the click probability of a search result only depends on its relevance and its ranking position.
- Dependent Click Model (DCM)[8]: a click model that is based on the cascade assumption that the user will sequentially examine the results list and find attractive results to click until she feels satisfied with the clicked result.
- User Browsing Model (UBM)[9]: a click model that assumes the examination probability on a search result depends on its ranking position and the distance to the last clicked result.
- Mobile Click Model (MCM)[10]: a click model that considers the click necessity bias (i.e.some vertical results can satisfy users' information need without a click) in user clicks.

2.2.2. Train and calibrate the user behavior models with real query logs

We train and calibrate all the user behavior models based on real query logs collected by Sogou.com, a commercial Chinese search engine, so the synthetic clicks are similar to the real user clicks.

Table 1
Statistics of dataset used in training user behavior model

	Training	Test
sessions	843,933	836,979
unique queries	569	642

We split the real logs evenly into training and test set, then strictly follow the training process of each user behavior model proposed in the original works[7, 9, 8, 10]. However, to make sure those models can work for all candidate documents, we assume that the attractiveness parameter α of each query-document pair only depends on its five-level relevance label (0-4).

2.2.3. Generating clicks with click simulators

Equipped with the trained user behavior models, the working process of click simulators on each query session can be summarized by the code provided in [11]. We add some necessary modifications to the click generating procedure and show it in Algorithm 1.

Algorithm 1 Generating synthetic clicks with a click simulator for a query session

```
Input: user behavior model $M$
       query session $s$ consisting of query $q$ and ranking list $d_1$,... $d_n$
       vector of relevance labels for documents $(r_{d_1}, ...r_{d_n})$
       vector of vertical types for documents $(v_{d_1}, ...v_{d_n})$
Output: vector of simulated clicks $(c_1, ...c_n)$
1: for $i = 0 \to n$ do
2:     Compute $p = P(C_i = 1|C_1 = c_1, ...C_{i-1} = c_{i-1})$ using previous clicks $c_1, ...c_{i-1}$,
       relevance label $r_{d_i}$,vertical type $v_{d_i}$ and parameters of M
3:     Generate random value $C_i$ from Bernoulli(p)
4: end for
```

3. EXPERIMENTS

We conduct a series of experiments to answer the following research questions: **RQ1** : How do different click simulators performs in predicting clicks and generating synthetic click logs? **RQ2**: Can we evaluate existing ULTR models with the ULTRE framework?

3.1. Examining click simulators (RQ1)

Experiment Set up
Datasets The dataset used in training user behavior models were sampled from real search log dataset released by Chinese commercial search engine Sogou.com. We divide the dataset into training and test sets with proportion 1:1. The statistics of the dataset are shown in Table 1.
Evaluation Metrics For click prediction task, we report the log-likelihood (LL) and perplexity (PPL) of each user behavior model. Higher values of log-likelihood and lower values of perplexity indicates better click prediction performance.

To measure the quality of generated samples from different click simulators, we compute Kullback-Leibler(KL) divergence between the distribution of real clicks and the distribution of simulated clicks and Reverse/Forward PPL.

Table 2
User behavior model performance on LL and PPL metrics.

	LL	PPL
DCM	-0.1848	1.2363
PBM	-0.1721	1.2059
UBM	-0.1513	1.2029
MCM	**-0.1503**	**1.1787**

Table 3
KL-divergence between click logs generated by different user behavior models and real log

	KL-divergence	
Model	Session-based	Rank-based
Baseline	0.1950	0.3884
DCM	0.1245	0.5325
PBM	0.2856	0.2212
UBM	**0.0771**	0.2173
MCM	0.0786	**0.1951**

The first metric is proposed by Malkevich et al.[11], it measures a local KL-divergence for every query and then calculate a weighted average of local divergences as follows:

$$KL - div = \frac{\sum_{q \in Q} KL - div(q).s_q}{\sum_{q \in Q} s_q}$$

where Q is the number of unique queries and s_q is the number of sessions observed for a particular query q. This metric can be calculated for two click distributions: the distribution over sessions which shows the percentage of sessions with a certain number of clicks and the distribution over ranks which shows how many times a certain rank was clicked. Lower values of the metrics correspond to better click simulation performance.

The second metric was first used in [12] to test the distributional converge of click models. Reverse PPL is the PPL of a surrogate model (an intermediary to evaluate the similarity between the generated samples and the real data samples) that is trained on generated samples and evaluate on real data. Forward PPL is the PPL of a surrogate model that is trained on real data and evaluated on generated samples.

Performance on predicting clicks
The results for the click prediction task on test set are presented in Table 2, from which we can observe that MCM performs the best among all models, similar to the observation in [10]. However, the others also have a relatively good performance, as their values of metrics are all close to the ideal values (0 for LL and 1 for PPL).

Quality of generated click logs
This section measures the similarity between the real log and generated click logs on test set.

Table 3 summarizes the simulation performance of the four user behavior models and baseline model (always simulate a click on the first position) in terms of the KL-divergence of the click distribution over sessions (session-based KL) and over ranks (rank-based KL), from which we can obtain following observations:
(1) UBM generates the best samples in terms of session-based KL-divergence, while MCM performs the best in terms of rank-based KL-divergence. Considering the value of session-based KL-divergence of MCM only slightly higher than the one of UBM, it's fair to say that the click logs generated by MCM are the most similar to the real logs.
(2) The samples of DCM are better than the samples generated by the baseline model in terms of the session-based KL-divergence, however the performance in terms of the rank-based

Table 4
Reverse/Forward PPL of Surrogate DCM/PBM/UBM/MCM models based on different synthetic datasets generated from target user behavior models(DCM/PBM/UBM/MCM)

	Surrogate DCM		Surrogate PBM		Surrogate UBM		Surrogate MCM	
Data	Reverse PPL	Forward PPL	Reverse PPL	Forward PPL	Reverse PPL	Forward PPL	Reverse PPL	Forward PPL
Real data	1.2363	1.2363	1.2059	1.2059	1.2029	1.2029	1.1787	1.1787
DCM samples	-	-	1.2374	1.3688	1.2350	1.3625	1.2191	1.3137
PBM samples	1.2824	1.2272	-	-	1.2061	1.1880	1.2053	1.2152
UBM samples	1.2409	1.2317	1.2055	1.1953	-	-	1.1802	1.1764
MCM samples	1.2388	1.2248	1.2061	1.1841	1.2031	1.1831	-	-

KL-divergence is rather low. A possible reason for that is DCM does not use rank-based examination parameter as the other three models. On the contrast, the samples of PBM are better regarding the rank-based KL-divergence and worse regarding session-based KL-divergence. Such observations may caused by the simple rank-based assumption used in PBM.

Table 4 shows the results of Reverse/Forward PPL of surrogate DCM/PBM/UBM/MCM models based on different synthetic datasets generated from target models (DCM/PBM/UBM/MCM).

To conduct an adequate and fair experiment, all models take the role of the surrogate model. For example, when we choose DCM as the surrogate model, the generated samples of the other three models (PBM/UBM/MCM)can be compared.

From Table 4 we can obtain the following observations:
(1) Samples generated by UBM and MCM achieves better performance than the ones of DCM and PBM, for the reason that when the surrogate model is UBM and MCM, MCM-samples and UBM-samples outperforms DCM-samples and PBM-samples respectively.
(2) The comparison between UBM-samples and MCM-samples is a little bit complex. Since when surrogate model is DCM, MCM-samples are better in terms of both Reverse PPL and Forward PPL, however when surrogate model is PBM, the better samples are different regarding different metric. As a result, we cannot conclude whether the samples generated by UBM or MCM are the most similar to the real logs.

3.2. Evaluating ULTR models with the ULTRE framework (RQ2)

To answer RQ2, we evaluate several offline ULTR models with the ULTRE framework.
Dataset and Simulation Setup
The dataset used to evaluate ULTR models is based on Sogou-SRR[2][13], a public dataset for relevance estimation and ranking in Web search. We select 1,211 unique queries with at least 10 successfully crawled results, 1,011 for training, 100 for validation and 100 for testing. In addition, we use a stratified sampling approach to ensure that the frequency of queries in each dataset is consistent with the one in the real logs. As mentioned in Section 3, we need a production ranker to produce ranking lists for training queries, so we trained a lambdaMART model with 1% data randomly sampled from the original training set (with 5-level relevance annotations). After that, we follow the click-simulation process in the ULTRE framework. Table 5 shows the details of the dataset we constructed.
Model Setup and Evaluation
We chose three offline ULTR models as our candidate models, which are IPW[1] (named PBM-IPS in [6]), CM-IPS[6] and DLA[2].

[2]http://www.thuir.cn/data-srr/

Table 5
Statistics of ULTRE dataset

	Training	Validation	Test
Unique queries	1,011	100	100
Session	144,675	100	100
Label	clicked(1) or not(0)	5-level relevance annotations(0-4)	5-level relevance annotations(0-4)

Table 6
Comparison of offline ULTR models on ULTRE data

	PBM	DCM	UBM	MCM
production ranker (baseline)	0.7815			
Full-info (skyline)	0.8182			
PBM-IPS(IPW)	0.8017	0.7826	0.8064	0.7647
CM-IPS	0.7894	0.7932	0.8050	0.7778
DLA	**0.8119**	**0.8173**$^+$	**0.8107**	**0.7932**$^+$

Significant improvements or degradations with respect to PBM-IPS are indicated with +/- in the paired samples t-test with $p \leq 0.05$. The best performance is highlighted in boldface.

To conduct a fair comparison, we followed the settings in [14], using a multiple-layer perceptron network (MLP) with three hidden layers (with 512,256,128 neurons) as the ranking model for all candidate models and set the batch size to 256. We trained each candidate for 10k steps, and chose the ranking model in the iteration that has the best performance on the validation set. Such experiment was repeated 10 times to ensure the reliability of final results. nDCG@5 was used to evaluate the performance of each candidate model.
Evaluation results
Table 6 shows nDCG@5 for three different offline ULTR models trained on different synthetic train sets. The baseline we used is the performance of production ranker and the skyline is the performance of a lambdaMART model trained on the whole train set with human annotations instead of biased clicks. From the results, we can see that:
(1) PBM-IPS trained on PBM-based training set performs the best compared to the model trained on other sets while CM-IPS trained on DCM-simulated training set performs the best compared to the model trained on other sets. That observation coincides with the conclusion in [6] that when the used behavior models used in click simulation and the correction method of bias are consistent, the results are better than the case in which they don't agree.
(2) Compared to PBM-IPS and CM-IPS, DLA performs the best on all synthetic train sets, which indicates that DLA is more robust and more adaptive to the change of user behavior assumption used in the click simulation. That advantage can be attributed to the the unification of learning propensity weights (used to correct bias in click data) and leaning ranking models proposed in DLA. Such learning paradigm can help DLA model adjust its propensity weights automatically to the difference between different synthetic training sets, while PBM-IPS and CM-IPS model

cannot.

The above observations demonstrate the usefulness and effectiveness of the ULTRE framework. By using the ULTRE framework, besides evaluating the performance of one particular model like many previous works have already done, we can conduct a fair and thorough comparison between different ULTR models. In addition, we have the chance to investigate the following questions: 1) to what extent the evaluation results will be influenced by the user simulation model and the mismatch between the assumptions of the simulation model and ranking model 2) which ULTR model can adapt to different environments defined by different simulation models and achieves a robust improvement in ranking performance.

4. Conclusion and Future work

In this paper, we introduce the ULTRE framework that aims to improve the simulation approach used in previous ULTR evaluation. Our experiments show that ULTRE framework can provide simulated-based training sets of both quality and diversity. More importantly, it enables us to conduct a thorough and relatively objective comparison of different ULTR models. We further design two evaluation protocols of using this framework as a shared evaluation service for both the offline and online ULTR models.

Our work includes an initial implementation for ULTRE framework and there are still some ongoing works for the final deployment. For example, we plan to adopt neural user behavior models such as Context-aware Click Simulator (CCS)[15] for the click simulation, since the user behavior models we used in this work are all based on the probabilistic graphic models (PGMs) and neural models may have a better click prediction performance. Moreover, the implementation of online service and comparison between online ULTR models under the ULTRE framework will be needed as we only present the comparison results of offline ULTR models in this paper. We plan to use the ULTRE framework in the Unbiased Learning to Rank Evaluation Task (ULTRE), a pilot task in NTCIR 16.

References

[1] T. Joachims, A. Swaminathan, T. Schnabel, Unbiased learning-to-rank with biased feedback, in: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 781–789.

[2] Q. Ai, K. Bi, C. Luo, J. Guo, W. B. Croft, Unbiased learning to rank with unbiased propensity estimation, in: The 41st International ACM SIGIR Conference on Research amp; Development in Information Retrieval, SIGIR '18, Association for Computing Machinery, New York, NY, USA, 2018, p. 385–394.

[3] Z. Hu, Y. Wang, Q. Peng, H. Li, Unbiased lambdamart: an unbiased pairwise learning-to-rank algorithm, in: The World Wide Web Conference, 2019, pp. 2830–2836.

[4] H. Wang, R. Langley, S. Kim, E. McCord-Snook, H. Wang, Efficient exploration of gradient space for online learning to rank, in: The 41st International ACM SIGIR Conference on Research amp; Development in Information Retrieval, SIGIR '18, Association for Computing Machinery, New York, NY, USA, 2018, p. 145–154.

[5] H. Oosterhuis, M. de Rijke, Differentiable unbiased online learning to rank, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM '18, Association for Computing Machinery, New York, NY, USA, 2018, p. 1293–1302.

[6] A. Vardasbi, M. de Rijke, I. Markov, Cascade model-based propensity estimation for counterfactual learning to rank, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 2089–2092.

[7] N. Craswell, O. Zoeter, M. Taylor, B. Ramsey, An experimental comparison of click position-bias models, in: Proceedings of the 2008 International Conference on Web Search and Data Mining, WSDM '08, Association for Computing Machinery, 2008, p. 87–94.

[8] F. Guo, C. Liu, Y. M. Wang, Efficient multiple-click models in web search, in: Proceedings of the Second ACM International Conference on Web Search and Data Mining, WSDM '09, Association for Computing Machinery, New York, NY, USA, 2009, p. 124–131.

[9] G. E. Dupret, B. Piwowarski, A user browsing model to predict search engine click data from past observations., in: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '08, Association for Computing Machinery, New York, NY, USA, 2008, p. 331–338.

[10] J. Mao, C. Luo, M. Zhang, S. Ma, Constructing click models for mobile search, in: The 41st International ACM SIGIR Conference on Research amp; Development in Information Retrieval, SIGIR '18, Association for Computing Machinery, New York, NY, USA, 2018, p. 775–784.

[11] S. Malkevich, I. Markov, E. Michailova, M. de Rijke, Evaluating and analyzing click simulation in web search, in: Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval, ICTIR '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 281–284.

[12] X. Dai, J. Lin, W. Zhang, S. Li, W. Liu, R. Tang, X. He, J. Hao, J. Wang, Y. Yu, An adversarial imitation click model for information retrieval, arXiv preprint arXiv2104.06077 (2021).

[13] J. Zhang, Y. Liu, S. Ma, Q. Tian, Relevance estimation with multiple information sources on search engine result pages, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, 2018, pp. 627–636.

[14] Q. Ai, T. Yang, H. Wang, J. Mao, Unbiased learning to rank: Online or offline?, ACM Trans. Inf. Syst. 39 (2021).

[15] J. Zhang, J. Mao, Y. Liu, R. Zhang, M. Zhang, S. Ma, J. Xu, Q. Tian, Context-aware ranking by constructing a virtual environment for reinforcement learning, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 1603–1612.