Regular Article

Unbiased Pairwise Approach toward Learning-to-Rank: An Empirical Study

Son Thanh Le[1], Ha Manh Tran[2], Quang Duy Nguyen[1], Sinh Van Nguyen[1]

[1] School of Computer Science and Engineering, International University - Vietnam National University, Ho Chi Minh City, Vietnam
[2] Department of Information Technology, University of Foreign Languages - Information Technology, Ho Chi Minh City, Vietnam

Correspondence: Ha Manh Tran, hatm@huflit.edu.vn
Communication: received 23 August 2021, revised 19 September 2021, accepted 20 September 2021
Online publication: 23 October 2021, Digital Object Identifier: 10.21553/rev-jec.281
The associate editor coordinating the review of this article and recommending it for publication was Prof. Vo Nguyen Quoc Bao.

Abstract– **With the bloom of information technology in recent decades, people are constantly being exposed to a huge amount of information. Learning-to-rank comes out as one of the solutions to ease out the mentioned obstacle by trying to rearrange objects according to their degrees of importance or relevance. This solution usually applies machine learning techniques to construct ranking models in information retrieval systems. The aim of this study is to explore and experiment the existing learning-to-rank approaches with real-life logs data. The study also includes estimating and minimizing the bias noise found in the click-through data of the recorded logs. Evaluation results have presented the advantage and disadvantage of the experimented approaches in realistic settings.**

Keywords– **Unbiased Pairwise, Learning-to-Rank, Machine Learning, Click Logs.**

1 Introduction

Nowadays, as an online content provider, one usually must present curated lists of products to their customers whose tastes might be vastly different from each other. Naturally, an emerging personalizing problem aims to provide each customer a better experience. Ranked retrieval involves sorting the provided items in such an order corresponding to each individual taste and personal preference. Thus the information retrieval systems must try to deduce the importance of each presented document based on users' information need.

Machine learning techniques might be able to supply a new perspective on how to estimate and calculate this information need. By tracking activities of past interactions between users and other documents that him or her has chosen to view, we could see this as a measure of preference. After all, user should be considered the expert on his or her information needs.

Learning-to-rank, a kind of machine learning, makes use of large training datasets in order to help the learning algorithm capture the patterns. Thus, traditional source of data, such as expert's manual annotated data quickly become obsolete. Most modern systems nowadays make use of implicit user feed-backs under the form of click-through logs. It is abundant, cheap to collect, continuous and renewable data source. Yet, they are noisy and biased.

Many past researches have pointed out that to reliably utilize such kind of data, one must account for numerous possible kinds of bias including position bias [1, 2], presentation bias [3], and quality-of-context bias [4], etc. Among which, the position bias has a strong influence on users' clicks [5].

The term of position bias describes the tendency of users to interact with items on top of a list with higher probability than with items at a lower position in the list, regardless of the item's actual relevance. This phenomenon is either due to laziness [1, 3] or due to trust to the search site (trust bias) [4, 6, 7].

This means that the said bias would pose both as a reason and a challenge to ranking systems or recommendation systems. On one hand, as the viewers are more likely to click on those rank higher, re-ranking the order item by their relevance would essentially enhance the viewers' experience on the platform which result in elevating the product's attractiveness, in turn could bring about other business advantages. On the other hand, because of this tendency, naive interpretation of clicks log might not be an accurately indication of the item's absolute relevance within the presented set thus introducing noise in the data and could be misleading.

In this study, we have explored the approach of combining regression-based Expectation-Maximization and DeepPropDCG [7] into an uniform workflow, and then experimented its performance on a production dataset. The contribution of this study is therefore twofold:

- Providing a modular workflow for easy alteration and tuning and various representations of the same data.
- Validating the approach against the real-life click logs for learning its sensible insights of capturing

and generalizing the desired behavioural knowledge.

The rest of the paper is structured as follows: the next section provides some background of previous efforts to incorporate bias into learning to rank. Section 3 describes the theoretical work of the parts in the focused approach. Section 4 reports the exploratory experiments and acquired knowledge through their results before the paper is concluded in Section 5.

2 Background

2.1 Unbiased Learning To Rank

There have been much effort in dealing with aforementioned problem. One such approach is click modeling. The idea is to incorporate the bias into the model by trying to mimicking the users' behavior and use it as an assumption [8]. One of the most famous models is known as the Cascade model [9] that assumes sequential user behavior. In other words, a user is assumed to scan documents one by one from the top; the scanning process continues after obtaining an irrelevant document but stops after obtaining a relevant one.

Another different approach is to treat bias as a counterfactual effect and quantify it [5, 10]. The quantification relies on Inverse Propensity Weighting developed in the causal inference field [11]. The said weighting technique has been viewed as a commonly-used technique to address the sample bias and has been widely adopted for unbiased evaluation and learning [12–15]. This approach employs a Bernoulli variable to denote if the relevance of a specific document is observed, or the propensity. As stated before, this quantity could be affected by numerous factors. In this approach, with the main goal of tackling position bias, it is assumed to depend on the position which the document is displayed to the user. After that, its inverse could be used for weighting each learning example. The idea is if an observed click further down the result lists is more likely to contain creditable information than those of top positions where position bias is heavier. The approach is thus referred to as inverse propensity weighting.

2.2 Position Bias Model

To further formalize the user behaviour described in the previous section, we scrutinize the position bias model. It is a simple yet effective generative click model and has been shown to be as effective as other sophisticated ones [8]. Suppose, for a query q (which could be a context collection and/or contains user personalizing information), document d is displayed at position $k \in [1, K]$. Any observed documents are examined and click can be generated only after the user has examined the mentioned document. In this model, it is assumed that the user clicks the document if and only if they examine the document and the document is relevant. The examination decision only depends on the position k but not on q and d.

Formally, the position bias model assumes an observed click Bernoulli variable C standing for whether the user clicks document d, two hidden Bernoulli variables E and R signifying whether the user examines the document and the relevance, respectively. Aforementioned assumptions could be expressed mathematically as follows:

$$\begin{aligned} &O \leftrightarrow E \\ &C = 1 \leftrightarrow E = 1, R = 1 \\ &P(E = 1|q, d, k) = P(E = 1|k) \\ &P(R = 1|q, d, k) = P(R = 1|q, d). \end{aligned}$$

Thus, the examining probability, or the propensity, only depends on the display position k and the perceived relevance is the true relevance. Based on that, we could derive the following formula:

$$\begin{aligned} P(C = 1|q, d, k) &= P(E = 1, R = 1|q, d, k) \\ &= P(E = 1|k) \times P(R = 1|q, d) \\ &= \theta_k \times \gamma_{q,d}, \end{aligned}$$

where we make use of the shorthands:

$$\begin{aligned} \theta_k &= P(E = 1|k) \\ \gamma_{q,d} &= P(R = 1|q, d). \end{aligned}$$

Given a click log under the form of $\mathcal{L} = (c, q, d, k)$, the log likelihood of the data is calculated by:

$$\begin{aligned} &\log P(\mathcal{L}) \\ &= \sum_{(c,q,d,k)\in\mathcal{L}} c \log(\theta_k \times \gamma_{q,d}) + (1 - c) \log(1 - \theta_k \times \gamma_{q,d}). \end{aligned}$$

3 The Focused Approach

With the goal of finding an applicable approach for realistic production experiments, we try to explore and tune previously proposed methods into a single workflow that works well with the real-life collected data. The focused approach is a workflow with two steps: unbiasing and learning-to-rank that employ regression-based Expectation-Maximization (EM) and DeepPropDCG, respectively.

3.1 Propensity Estimation

It is clear now that one of the key components in unbiased learning to ranking is to estimate the unknown propensity of the current system/data. There have been many attempts. However, most of the ideas depend on delivering randomized results in order to calculate the needed propensity (e.g RandTopN [2], RandPair [5],...), which in term would cause a drop in the users' experience.

Recently, Wang et al. [2] have presented a different regression-based EM algorithm that does not need the result randomization intervention which vastly helps reduce the develop, deploy, and evaluate process. The work is based on the position bias model. In order to find the parameters that maximize the log-likelihood $\log P(\mathcal{L})$, the algorithm employ a technique based on EM method.

The Regression-based EM algorithm iterates over the Expectation and Maximization steps to update $\{\gamma_k\}$ and $\{\theta_{q,d}\}$. At iteration t+1, the Expectation step estimates the distribution of the hidden variable E and R given parameters from iteration t and the observed data $\mathcal{L}$. From this, marginals $P(E = 1|c, q, d, k)$ and $P(R = 1|c, q, d, k)$ can be derived.

In the Maximization step at t+1, the parameters $\theta_k^{(t+1)}$ and $\gamma_{q,d}^{(t+1)}$ are updated to their maximum likelihood values given the posterior probabilities from the Expectation step. In traditional EM, the estimation phase for $\gamma_{q,d}^{(t+1)}$ would be working with (q, d) identifiers to output the probability for each pair. Using the exact identifiers has been proved to be challenging in personal search due to the highly sparse and noisy nature of click data. So instead, we work with the feature vectors $x_{q,d}$ and use a regression function $f(x)$ to maximize the likelihood of a sample a binary relevance label $r \in \{0, 1\}$ according to $P(R = 1|c, q, d, k)$.

Since an actual click logs is employed to evaluate, there is no exact results to the propensity, furthermore, the method's theoretical validity has been proven in the original paper. Thus, within the scope of this study, we survey over the convergence and the qualitative outcomes of this step.

3.2 DeepPropDCG

DeepPropDCG is a pairwise learning-to-rank approach proposed by Agarwa et al. [7]. It aims to learn a scoring function $f(x)$, whose results naturally form a ranking system by sorting the scores of candidate documents:

$$S_f(\mathbf{d}) = \text{argsort}\{f(x)|x \in \mathbf{x}\}.$$

The $rank(y|S_f(\mathbf{d}))$ of a result is thus a discontinuous step function of the score. Instead of directly using this in loss, we could substitute it with a (sub-)differentiable upperbound:

$$\begin{aligned} rank(y|S_f(\mathbf{d})) - 1 &= \sum_{\substack{x' \in \mathbf{x} \\ x' \neq x}} \mathbb{1}_{f(x')-f(x)>0} \\ &\leq \sum_{\substack{x' \in \mathbf{x} \\ x' \neq x}} \max(1 - (f(x) - f(x')), 0). \end{aligned}$$

Suppose, for each query q, we have a set of candidate documents $\mathbf{d}$ and, a set of features $\mathbf{x}$. After the ranking process, the system S returns a ranking $\mathbf{y}$. A wide range of popular and well known additive ranking performance metrics can be expressed by the formula:

$$\Delta(\mathbf{y}|q, \{r\}) = \sum_{y \in \mathbf{y}} \lambda(rank(y|\mathbf{y})) \times r_{q,y},$$

where $\lambda()$ could be any weighting function depending on the rank of the score y in $\mathbf{y}$, or $rank(y|\mathbf{y})$. For the sake of simplicity, the relevances are binary, $r_{q,y} \in \{0, 1\}$. Such metrics can be corrected for unwanted bias [2, 16] by weighting each result document with the inverse of the propensity:

$$\hat{\Delta}(\mathbf{y}|q_i, \{r\}) = \sum_{y \in \mathbf{y}} \frac{\lambda(rank(y|\mathbf{y}))}{p_{i,y}}$$

with the propensities $p_{i,y} = P(O_y = 1)$ are estimated beforehand. From that, we can derive the loss, or the risk, of the whole system S as:

$$\hat{R}(S) = \frac{1}{n} \sum_{i=1}^{n} \sum_{y \in \mathbf{y}_i} \frac{\lambda(rank(y|\mathbf{y}_i))}{p_{i,y}}.$$

By rearranging terms and applying the *rank* function, the risk becomes:

$$\hat{R}(S) \leq \frac{1}{n} \sum_{i=1}^{n} \frac{1}{p_i} \lambda\Big(\sum_{\substack{x' \in \mathbf{x_i} \\ x' \neq x_i}} \max(1 - (f(x_i) - f(x')), 0)\Big).$$

With the goal of optimizing nDCG (DCG) metric, we plug in the corresponding function and get the following formula:

$$\frac{1}{n} \sum_{i=1}^{n} \frac{-1}{p_i} \log^{-1}(2 + \sum_{\substack{x' \in \mathbf{x_i} \\ x' \neq x_i}} \max(1 - (f(x_i) - f(x')), 0)).$$

This could be considered as our minimization objective. Since the function ties together sublosses from pairs of document, stochastic gradient descent (SGD) is not directly feasible at the level of individual documents. Instead, SGD is performed at the level of document pair.

3.3 Evaluation Metric

Given the current scope which regards situation where each query can have multiple relevant documents, the evaluation metric used in this paper is a variant of nDCG. Due to click bias, the standard nDCG is not suitable for offline evaluation. We thus employ a weighted version of it, namely PSnDCG (in which PS is for Propensity Scored), which can correct the introduced bias [7, 17, 18]. The metric is defined as

$$\text{PSDCG} = \sum_{i=1}^{p} \frac{rel_i}{p_i \log_2(i+1)},$$

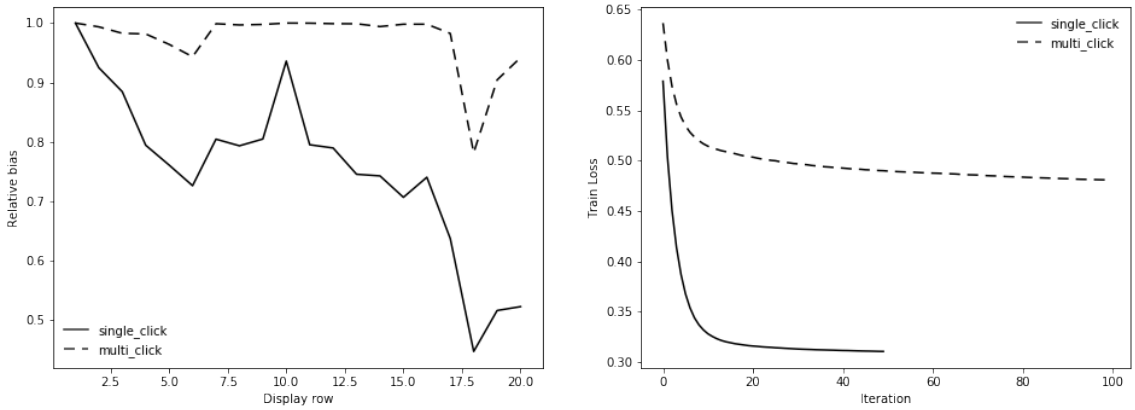
$$\text{PSnDCG} = \frac{\text{PSDCG}}{\text{PSInDCG}},$$

where p_l is the propensity θ estimated from the algorithm and PSInDCG is the PSDCG score of the optimal ranking - PS Ideal DCG.

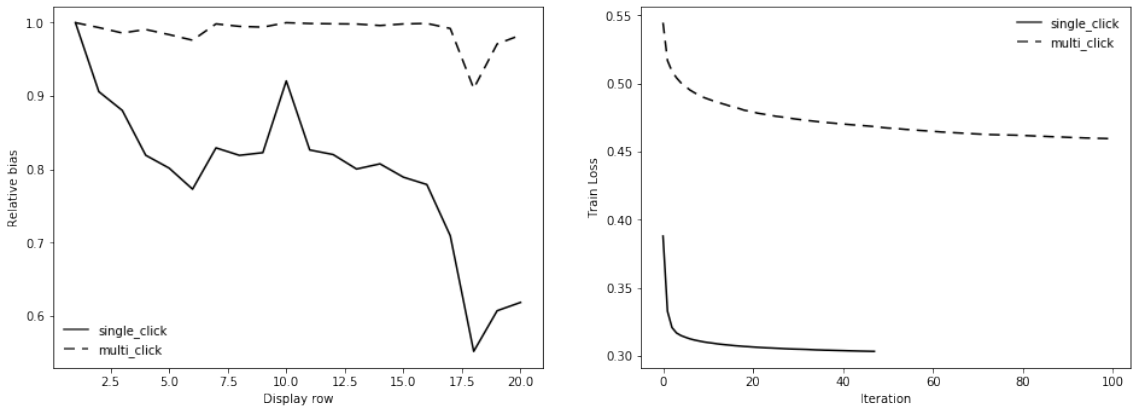
4 Empirical Evaluation

4.1 Datasets

The datasets used in this experiment are processed from click-through logs from where users are presented with a collection of documents manually composed and handpicked by the firm's editors following a specific subject or theme. The raw logs hold the user identifier, the collection identifier, the clicked document identifier, and some other information for each record for each



(a) GBDT_1



(b) GBDT_5

Figure 1. Normalized estimated bias and training loss of models trained with single-click versus multi-click datasets.

time the user clicks on an document presented within a collection, as reported in Table I. After collected, all the logs are segregated into sessions by grouping by (user, collection) so that every click event within a session is not 60 minutes away from one another. Each query is supposed to represent a chain of actions of a user going down a collection in a default ranking, clicking on documents.

Each record has total 401 features. These are generated from the available ranking features of the current system which are the latent representations of users and items generated by basic collaborative filtering method at a point before the time used for generating the supervised featured. Each item and its collection are associated with one or many "genre" tags describing the categories to which the object belongs. Moreover, higher business-level classifications, namely "audience", are also assigned to users. The "audience" labels depend on the genres with which the user has made interaction and some other business metrics. It is worth mentioning that, the collections are manually composed and curated by the service's editors following various themes and factors which eventually act as contexts information.

The supervised features play the role of representing the (q, d) to be learnt. We try to produce features which could capture numerous abstraction levels:

- User's features describes the tastes and preferences over many scopes of time for personalization.
- Document's features describes the characteristics.
- Collection's features describes the context in which the user and item interact.

Table I
Some Basic Statistics of the Datasets

# of users	105000
# of documents	5700
# of queries	290000

We take a sample of its processed logs from a 5-week period. The data of the first 4 weeks is used for training and the data of the last week is used for evaluating. Each query contains 2 clicked items on average.

4.2 Bias Estimation and Unbiasing

4.2.1 Meta-tuning on Query Representation: The training dataset comes in two versions:

- a multi-click version as described above where each query contains multiple clicked items
- a single-click version where queries are duplicated and modified so that each new query only contains one clicked item.

The estimated bias and training loss across multiple models using two training dataset versions are presented in Figure 1. The bias estimations are normalized so that the propensity at position 0^{th} is 1.0. As for the

Table II
Final Training Loss

Model	Training loss
GBDT_1	0.311
GBDT_3	0.305
GBDT_5	0.303

function $f(x)$, to account for the non-linearity, following the study of Wang et al. [2], we choose to compare between GBDTs of various settings. More specifically, with the GDBT method we used depth-3 trees and set the shrinkage to 0.2 with distinct number of boosting round per iteration r ranging from 1 to 10, denoted by "GDBT_{r}". All models were equipped with early stopping with the tolerance of 10^{-4} for 5 iterations.

On the first look, there exist consistent performance gaps in term of both training loss and estimation gaps in term of normalized bias estimation values across different run regardless of the architectures and hyper-parameters used. This fact ensures that we are able to sufficiently draw conclusions about the datasets difference from below observations since other possible variant causes (e.g the nature of the algorithms used by models, random initialization states, etc.) have already been minimized and/or eliminated.

The right column of the figure shows that models using the multi-click set take twice as long to converge compared to their counterparts. Not only that, the final loss are also significantly higher than that of the single-click ones. It is also worth noting that although low training loss values do not suggest low performance but such high losses could indicate the inability to learn. Moreover, looking at the left side of the figure, multi-click models also failed to produce sensible bias estimation as the values are relatively the same across the position axis. This implies the absence of position bias, which goes against previous studies and assumptions about human behavior [1, 4]. On the contrary, models which use the single-click version in the training process show much more sensible estimation of position bias as there are noticeable downward trends along the position axis as expected. In addition, the converged rates are much faster (almost $\times 2$) and at much lower loss. From this point on wards, unless explicitly stated, the training process would always be done on the single-click version.

4.2.2 Data Fitting: Regarding estimated bias values shown in Figure 2, all models are able to generate a sensible downward trend along the displayed position, which stay consistent to previous studies. At a glance, noises and fluctuation are presented from around the 18^{th} position. This can be explained by the fact that not many queries have result spanning more than 15 items. Interestingly, there is a noticeable rise which occurs at position 11^{th} throughout all trials. This unexpected behavior could be the nature of the used data and could be tuned when in production by utilizing large datasets.

Regarding the training loss, although the learning curve is visually different from one another, as reported

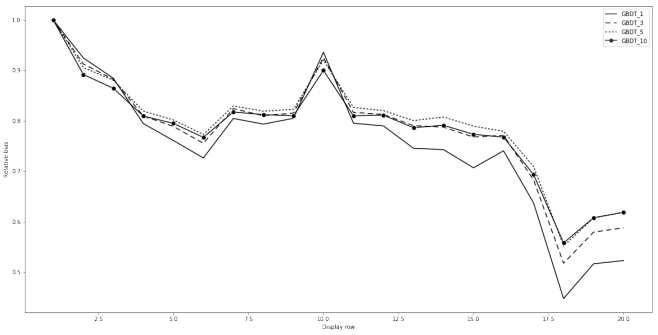


Figure 2. Normalized estimated bias

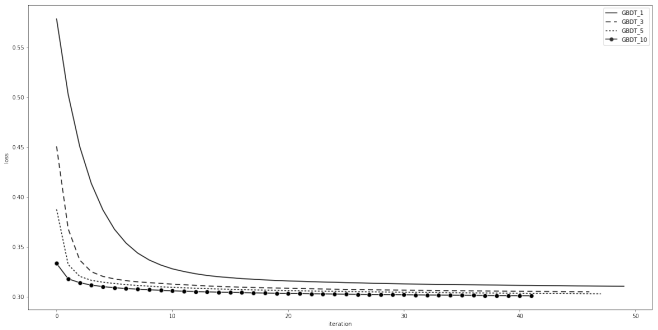
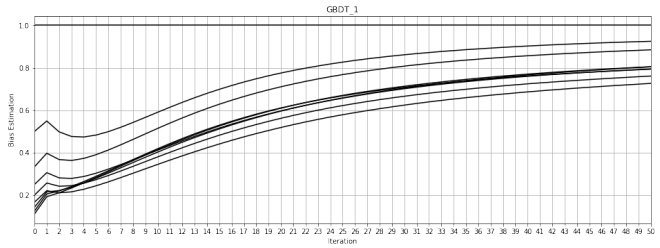


Figure 3. Training loss

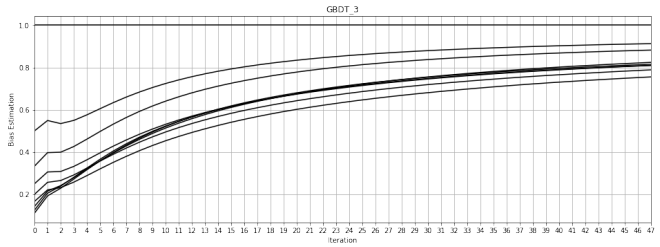
in Table II, the final losses are relatively similar across models. Recall that all the runs employs early stopping with the tolerance of 5 iterations, this means each run has various length corresponding to the point which the model's learning capability has already worn off. This could either means the residual error has become too small. Figure 3 shows at the few first iteration. In general, the higher number of trees used in an iteration, the faster convergence is achieved.

Albeit there are some differences in term of converge rate, the estimate procedure of our Regression-based EM is an iterative process in which the bias values is repeatedly recalculated and updated after every iteration. This essentially means apart from the training loss, which indicates how well each model fits to the data, there is also the convergence of the bias estimation values that must be taken into account. We theorize that a too fast learning paces may be harmful to the overall algorithm as there is not enough time for the bias values to adjust and stabilize. In such situation, the learnt models can be of misleading results. Further investigation into the raised problem is discussed the next section.

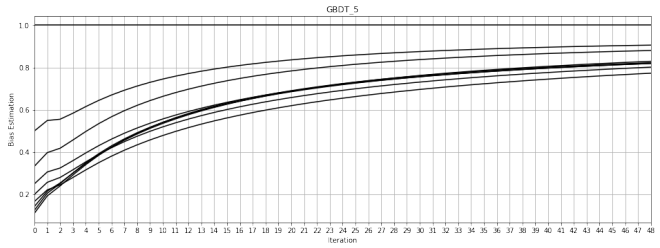
4.2.3 Bias Estimation Convergence: As addressed in the previous section, EM-algorithm consists of two interleaved steps in which the two unknowns, namely the bias and the ranking function, are learnt and updated at the same time. In other words, the propensity values are also learnable parameters that must be re-evaluated overtime to, ideally, converge. Naturally, the convergence states of the said values are also of valuable insights on how well the training process has taken place. As in Figure 4, it can be observed that the propensity values gradually converges from their



(a) GBDT_1



(b) GBDT_3



(c) GBDT_5

Figure 4. Bias estimation values of top 10 positions over time.

Table III
Bias Estimation Derivative Sums

Model	Last iteration's derivative sum	Last 3 iterations' derivative sum
GBDT_1	0.062	0.194
GBDT_3	0.060	0.185
GBDT_5	**0.055**	**0.172**

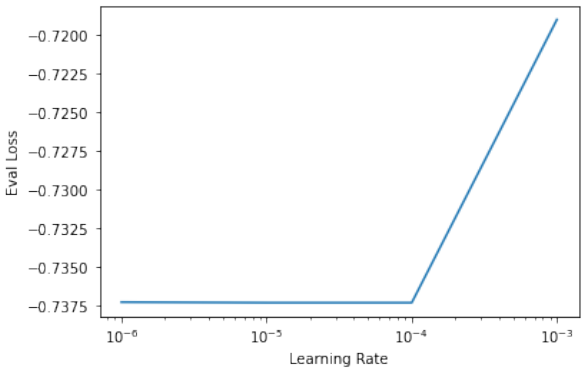


Figure 5. Performance by number of learning rates.

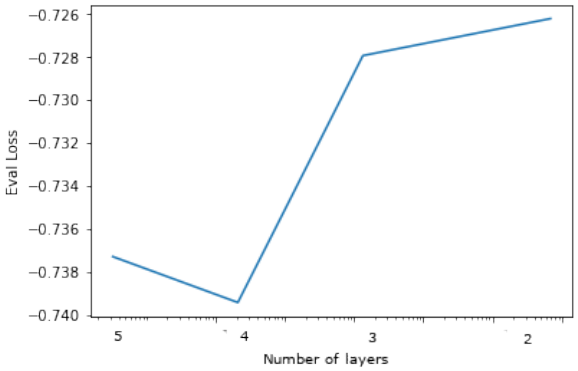


Figure 6. Performance by number of parameters.

Table IV
MLPs Layer Configurations

Number of layers	Layer configurations
2	[256, 128]
3	[256,128,64]
4	[256,128,64,64]
5	[256,256,128,128,64]

initialization after some iteration at different rates in different models used. The need to quantitatively measure the convergence of value, specifically $\{\theta_k\}$, thus emerges. To quantify such the concept, we choose to use the derivative of the last iteration and the summation of the derivatives of the last three iterations, the closer the metric value to zero, the better the convergence. The derivatives is computed using second order accurate central differences in the interior points, and second order accurate one-side backward differences at the boundary. Given that, let T be the number or training iteration, the metric could be formally defined as

$$\sum_{t'=0}^{t} \sum_{k=1}^{K} f'(\theta_k^{(T-t')})$$

with the values of t are chosen to be 0 and 2 corresponding to the last iteration's derivative sum and the last 3 iterations' derivatives sum, respectively.

The results are shown in Table III. Even though the chosen metric is arbitrary and does not yield any valuable insight when being on its own, we could compare the values in a relative manner to draw sufficient conclusions. It is clearly shown that the sweet spot is 5 trees used per iteration.

4.3 Learning to Rank

4.3.1 Hyper-parameters Tuning: Figure 5 shows the average performances of various different learning rate choices across multiple runs. As we could clearly see, 10^{-3} is shown to be too large, effectively worsen the performance of our model with respect to other finer values. From the Figure, we could also draw the conclusion that learning rates with values ranging from 10^{-4} to 10^{-6} yield relatively the same performance.

According to Figure 5, the learning rate choice, overall, does not make significant difference once going pass the 10^{-4} value. 10^{-3} is shown to be too large. Figure 6 shows the performance at various $f(x)$'s complexity depicted by the models' number of layers. It seems to reach its maximum performance at an intermediate value.

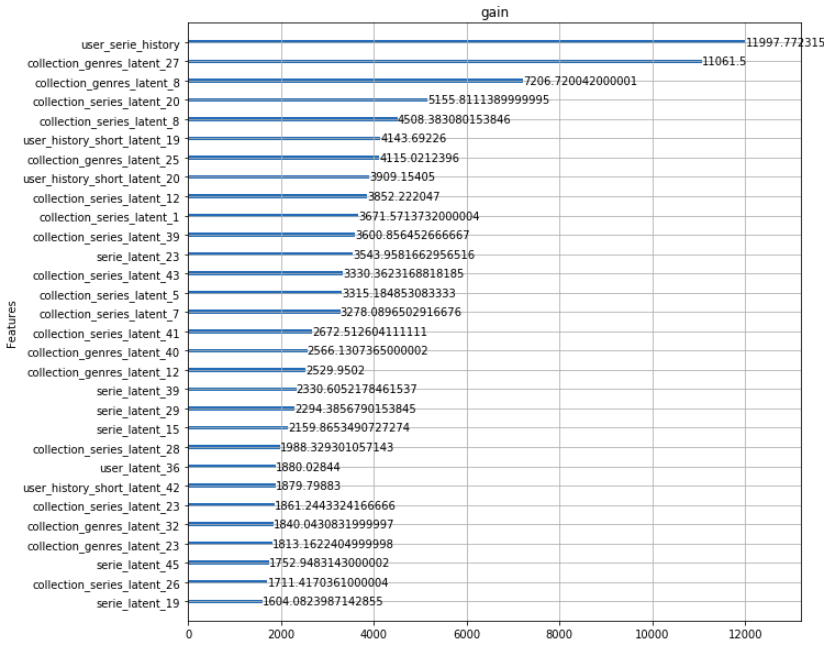


Figure 7. Important features of GBDT_5

Table V
PSnDCG Evaluation

Model	PSnDCG
DeepPropDCG	0.834
LambdaMART	0.767
GBDT	0.727
Default Order	0.578

4.3.2 Ranking Capability: The evaluation results provide a comprehensive look upon models against one another. Along with the PSnDCG-graded models, we also include the nDCG score of the default order by the authors of the ranking algorithm LambdaMART [19]. Noted that in the case of the default order, no personalization is presented here, every user is given the same ordering.

4.4 Qualitative Insights

MLPs architectures are well known for their power in various fields, and also in learning-to-rank specifically. But one of their drawbacks is the black-box nature. On the contrary, we could utilize the GBDT learnt in bias estimation step, taking advantage of its high interpretability to gain extra insights into the problem.

The average gain of splits which use a particular feature is plotted in Figure 7, the top 30 features are shown. With the top feature being the indicator whether the user has interacted with the item before, this can be explained by the fact that users often use the lists as a way to navigate to the items that they are consuming. Apart from that special case, the other top features shows extensive usage of collection's features which should contain information on the overall theme of the collection. Beside that, we could also observe that the personal tastes and preferences are well captured as the user's latent and their history are also in use.

5 Conclusion

We have presented the workflow of meta-tuning and producing sufficient data and the effort of capturing and evaluating position bias thoroughly provided in an in-house offline manner. The explored approach within the scope of this study apparently appears to yield improvement over the previously uniform manual rankings and successfully capture the biased behaviour of the users. However, the study still contains some drawbacks and the worthiest mentions are the lack of online A/B evaluation and explicit relevance feedback. The future work focuses on the following improvements regarding both implementation and evaluation: (i) employing further hyper-parameters tuning and more advanced and specialized architecture one may achieve better performance; (ii) applying online A/B testing; and (iii) gathering explicit relevance feedback from human judges to formulate more precise evaluations.

Acknowledgment

This research activity is funded by Ho Chi Minh City University of Foreign Languages–Information Technology under the grant number H2021-05. We would like to thank the NAB Studio data team and the team leader, Mr. Hoang Gia Vu, for the company's collected dataset and continuous support.

References

[1] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay, "Accurately interpreting clickthrough data as implicit feedback," in *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '05. New York, NY, USA: ACM, 2005, pp. 154–161.

[2] X. Wang, N. Golbandi, M. Bendersky, D. Metzler, and M. Najork, "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 '18)*. New York, NY, USA: ACM, 2018, pp. 610–618.
[3] Y. Yue, R. Patel, and H. Roehrig, "Beyond position bias: Examining result attractiveness as a source of presentation bias in clickthrough data," in *Proceedings of the 19th International Conference on World Wide Web (WWW '10)*. New York, NY, USA: ACM, 2010, pp. 1011–1018.
[4] T. Joachims, L. Granka, B. Pan, H. Hembrooke, F. Radlinski, and G. Gay, "Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search," *ACM Transactions on Information Systems*, vol. 25, no. 2, pp. 7–es, Apr. 2007.
[5] T. Joachims, A. Swaminathan, and 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)*. New York, NY, USA: ACM, 2017, pp. 781–789.
[6] M. O'Brien and M. T. Keane, "Modeling result-list searching in the world wide web: The role of relevance topologies and trust bias," in *Proceedings of the 28th annual meeting of the cognitive science society*, vol. 28. eScholarship–University of California, 2006, pp. 1881–1886.
[7] A. Agarwal, X. Wang, C. Li, M. Bendersky, and M. Najork, "Addressing trust bias for unbiased learning-to-rank," in *The World Wide Web Conference (WWW '19)*. New York, NY, USA: ACM, 2019, pp. 4–14.
[8] A. Chuklin, I. Markov, and M. Rijke, "Click models for web search," *Synthesis lectures on information concepts, retrieval, and services*, vol. 7, no. 3, pp. 1–115, 2015.
[9] N. Craswell, O. Zoeter, M. Taylor, and 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)*. New York, NY, USA: ACM, 2008, pp. 87–94.
[10] X. Wang, M. Bendersky, D. Metzler, and M. Najork, "Learning to rank with selection bias in personal search," in *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '16)*. New York, NY, USA: ACM, 2016, pp. 115–124.
[11] P. R. Rosenbaum and D. B. Rubin, "The central role of the propensity score in observational studies for causal effects," *Biometrika*, vol. 70, no. 1, pp. 41–55, 1983.
[12] A. Agarwal, S. Basu, T. Schnabel, and T. Joachims, "Effective evaluation using logged bandit feedback from multiple loggers," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '17)*. New York, NY, USA: ACM, 2017, pp. 687–696.
[13] M. Dudík, J. Langford, and L. Li, "Doubly robust policy evaluation and learning," in *Proceedings of the 28th International Conference on International Conference on Machine Learning (ICML'11)*. Madison, WI, USA: Omnipress, 2011, pp. 1097–1104.
[14] L. Li, S. Chen, J. Kleban, and A. Gupta, "Counterfactual estimation and optimization of click metrics in search engines: A case study," in *Proceedings of the 24th International Conference on World Wide Web (WWW '15 Companion)*. New York, NY, USA: ACM, 2015, pp. 929–934.
[15] L. Li, W. Chu, J. Langford, and X. Wang, "Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms," in *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining (WSDM '11)*. New York, NY, USA: ACM, 2011, pp. 297–306.
[16] A. Agarwal, K. Takatsu, I. Zaitsev, and T. Joachims, "A general framework for counterfactual learning-to-rank," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19)*. New York, NY, USA: ACM, 2019, pp. 5–14.
[17] H. Jain, Y. Prabhu, and M. Varma, "Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. New York, NY, USA: ACM, 2016, pp. 935–944.
[18] K. Bhatia, K. Dahiya, H. Jain, Y. Prabhu, and M. Varma, "The extreme classification repository: Multi-label datasets and code," *http://manikvarma.org/ downloads/XC/XMLRepository.html*, 2016.
[19] C. Burges, "From ranknet to lambdarank to lambdamart: An overview," Microsoft Research, Tech. Rep., 2010, mSR-TR-2010-82.

Son Thanh Le received his master degree in computer science in 2006 from Korea Advanced Institute of Science and Technology (KAIST), Korea. He is working as lecturer at School of Computer Science and Engineering, International University–HCMC Vietnam National University. His current research interests focus on data science, web technology, information retrieval, mobile computing and computer networks.

Ha Manh Tran is associate professor of computer science at HCMC University of Foreign Languages–Information Technology. He obtained his master degree in computer science in 2004 from the University of Birmingham, United Kingdom and his doctoral degree in computer science in 2009 from Jacobs University Bremen, Germany. His research interests include communication networks, distributed systems, network management, information retrieval and machine learning.

Quang Duy Nguyen received received his master degree in computer science in 2018 at International University–HCMC Vietnam National University. He is working as research associate at School of Computer Science and Engineering, International University–HCMC Vietnam National University. His current research interests focus on data science, web technology, information retrieval and machine learning.

Sinh Van Nguyen is working at the School of Computer Science and Engineering, International University–HCMC Vietnam National University. He received the doctoral degree of computer science in 2013 from Aix-Marseille University, France and the master degree of computer science in 2008 from Asian Institute of Technology (AIT), Thailand. His research interests include computer graphics, images processing, geometric modeling, AR & VR applications, smart web.