Estimating the Hessian Matrix of Ranking Objectives for Stochastic Learning to Rank with Gradient Boosted Trees

Jingwei Kang
University of Amsterdam
Amsterdam, The Netherlands
j.kang@uva.nl

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

Harrie Oosterhuis
Radboud University
Nijmegen, The Netherlands
harrie.oosterhuis@ru.nl

ABSTRACT

Stochastic learning to rank (LTR) is a recent branch in the LTR field that concerns the optimization of probabilistic ranking models. Their probabilistic behavior enables certain ranking qualities that are impossible with deterministic models. For example, they can increase the diversity of displayed documents, increase fairness of exposure over documents, and better balance exploitation and exploration through randomization. A core difficulty in LTR is gradient estimation, for this reason, existing stochastic LTR methods have been limited to differentiable ranking models (e.g., neural networks). This is in stark contrast with the general field of LTR where Gradient Boosted Decision Trees (GBDTs) have long been considered the state-of-the-art.

In this work, we address this gap by introducing the first stochastic LTR method for GBDTs. Our main contribution is a novel estimator for the second-order derivatives, i.e., the Hessian matrix, which is a requirement for effective GBDTs. To efficiently compute both the first and second-order derivatives simultaneously, we incorporate our estimator into the existing PL-Rank framework, which was originally designed for first-order derivatives only. Our experimental results indicate that stochastic LTR without the Hessian has extremely poor performance, whilst the performance is competitive with the current state-of-the-art with our estimated Hessian. Thus, through the contribution of our novel Hessian estimation method, we have successfully introduced GBDTs to stochastic LTR.

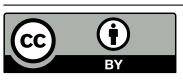
CCS CONCEPTS

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

KEYWORDS

Learning to Rank, XGBoost, Pytorch

ACM Reference Format:
Jingwei Kang, Maarten de Rijke, and Harrie Oosterhuis. 2024. Estimating the Hessian Matrix of Ranking Objectives for Stochastic Learning to Rank with Gradient Boosted Trees. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24), July 14–18, 2024, Washington, DC, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3626772.3657918



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

SIGIR '24, July 14–18, 2024, Washington, DC, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0431-4/24/07
https://doi.org/10.1145/3626772.3657918

1 INTRODUCTION

Learning to rank (LTR) is a core problem in the information retrieval (IR) field that concerns the optimization of ranking models [16]. Generally, ranking models use a scoring function that independently assigns a score to each document to be ranked, and subsequently, ranks them by ordering them based on their assigned scores. Traditionally, this ordering was done by deterministically sorting, i.e., the highest scored document is placed at the first rank, the second highest at the second rank, etc. The aim of LTR techniques is to optimize a ranking metric based on the produced rankings, for instance, discounted cumulative gain (DCG), precision, or recall [13, 15]. The difficulty with this approach is that these metrics, and the sorting process underlying the ranking, are not differentiable [4, 16, 18]. Therefore, LTR has always relied on methods for gradient estimation or approximation [19].

The existing families of LTR methods can be divided into three categories according to their optimization objective: (i) heuristic approximations of ranking metrics [14, 23]; (ii) proven lower bounds on ranking metrics [6, 26, 29, 30]; or (iii) the expected values of ranking metrics under stochastic ranking models [2, 11, 18, 25]. The latter often choose to optimize Plackett-Luce (PL) ranking models as their stochastic rankers [5, 17–19, 21, 27, 28]. The advantage of optimizing the expected values of ranking metrics is that they are fully differentiable, unlike their deterministic counterparts. However, this stochastic LTR approach comes with two important challenges: (i) If the goal is to produce a deterministic model then it may not be perfectly aligned with the stochastic LTR objective; and (ii) exact computation of an expected value requires iterating over all possible rankings, i.e., every permutation of documents, which is infeasible in practice [18].

Several methods have been designed for the latter issue. Early listwise LTR work optimizes PL ranking models, and avoids the permutation problem by only maximizing the probability of a single optimal ranking [5, 28]. More recent stochastic LTR work uses sampling to approximate the gradient instead. Bruch et al. [2] proposed to sample rankings from PL models using the computationally efficient Gumbel-Softmax trick [12]. Singh and Joachims [25] applied a policy-gradient method to approximate the gradient based on sampled rankings. Subsequently, Oosterhuis [18] proposed the PL-Rank method that makes use of several mathematical properties of ranking metrics and PL ranking models to perform gradient estimation with high computational efficiency [19]. Stochastic LTR with PL-Rank converges significantly quicker and at higher levels of performance than standard policy-gradient approaches.

Besides these different high-level approaches, the current state of stochastic LTR has another interesting difference with deterministic LTR. To the best of our knowledge, all stochastic LTR

$$\frac{\partial^2}{\partial m(d)^2} R(\pi) = \mathbb{E}_y \Bigg[\Bigg(1 - \sum_{x=1}^{\text{rank}(d,y)} \pi(d \mid y_{1:x-1}) \Bigg) \Bigg(\sum_{k=\text{rank}(d,y)+1}^{K} \theta_k \rho_{y_k} \Bigg) + \sum_{k=1}^{\text{rank}(d,y)} \pi(d \mid y_{1:k-1}) \Bigg(\theta_k \rho_d - \sum_{x=k}^{K} \theta_x \rho_{y_x} \Bigg) \Bigg(\mathbb{1}[d \in y] + 1 - \pi(d \mid y_{1:k-1}) - \sum_{x=1}^{\text{rank}(d,y)} \pi(d \mid y_{1:x-1}) \Bigg) \Bigg]$$

Figure 1: The complete second-order derivative of the reward function $R(\pi)$ w.r.t. the scoring function $m(d)$.

works optimize neural networks (NNs) [3, 17–19, 21],[1] whilst in deterministic LTR, Gradient Boosted Decision Trees (GBDTs) have historically had the most prominent role [4, 8, 10, 24]. Accordingly, it is surprising that despite their enormous importance to the LTR field, GBDTs have been largely ignored for stochastic LTR.

This work aims to bridge this gap; We first point out that existing stochastic LTR work has been focused on estimating the first-order derivatives of ranking objectives. But GBDTs also require the second-order derivatives for their optimization, as a result, the existing methods are not applicable to GBDTs [17–19, 21]. We address this issue by proposing a novel estimator for these second-order derivatives, i.e., Hessian matrices, in a computationally efficient manner. Our method integrates very well with PL-Rank [18, 19], such that both first and second-order derivatives are computed efficiently. Our experimental results show that stochastic LTR via GBDTs without the Hessian leads to detrimental performance. Conversely, with our estimated Hessian, GBDTs are able to outperform NNs on several LTR benchmarks by considerable margins. Furthermore, we observe that stochastic LTR with NNs does not have stable convergence, and thus requires early-stopping, but we do not observe such behavior for GBDTs with our estimated Hessians. Thus, by contributing the first Hessian estimation method for stochastic LTR, we have closed an important gap with deterministic LTR, that enables substantial performance and stability improvements.

2 BACKGROUND: STOCHASTIC LTR

PL ranking models. Plackett-Luce ranking models provide a probability distribution over fixed-length permutations [17, 21]. Accordingly, they have often been deployed as models of stochastic ranking models [5, 17–19, 21, 27, 28]. We use y to denote a ranking $y = [y_1, y_2, \ldots, y_K]$ with a cutoff of K, and $y_{1:k} = [y_1, y_2, \ldots, y_k]$ the partial ranking up to rank k. Let π indicate a Plackett-Luce ranking model, with $m(d)$ representing the log score of document d, then the probability that d is chosen to be the k-th item in ranking y from the set of items D is:

$$\pi(d \mid y_{1:k-1}, D) = \frac{e^{m(d)} \mathbb{1}[d \notin y_{1:k-1}]}{\sum_{d' \in D \setminus y_{1:k-1}} e^{m(d')}}. \tag{1}$$

The probability of the overall ranking y is the product of the placement probabilities of each individual document:

$$\pi(y) = \prod_{k=1}^{K} \pi(y_k \mid y_{1:k-1}, D). \tag{2}$$

Ranking metrics. Our objective is to optimize a ranking metric that fits the form of a *DCG@K* metric [15]. This means that each document d has a relevance ρ_d and each rank has a weight θ_k, e.g., for standard DCG: $\theta_k = \frac{\mathbb{1}[k \leq K]}{\log_2(k+1)}$; the expected value of this metric under π is then:

$$R(\pi) = \sum_{y \in \pi} \pi(y) \sum_{k=1}^{K} \theta_k \rho_{y_k} = \mathbb{E}_y \Bigg[\sum_{k=1}^{K} \theta_k \rho_{y_k} \Bigg]. \tag{3}$$

PL-Rank. Oosterhuis [18] found that the gradient of $R(\pi)$ w.r.t. the general scoring function $m(d)$ can be expressed as:

$$\frac{\partial}{\partial m(d)} R(\pi) = \mathbb{E}_y \Bigg[\Bigg(\sum_{k=\text{rank}(d,y)+1}^{K} \theta_k \rho_{y_k} \Bigg) + \sum_{k=1}^{\text{rank}(d,y)} \pi(d \mid y_{1:k-1}) \Bigg(\theta_k \rho_d - \sum_{x=k}^{K} \theta_x \rho_{y_x} \Bigg) \Bigg]. \tag{4}$$

Subsequently, Oosterhuis [19] introduced the PL-Rank algorithm (specifically PL-Rank-3) to compute this gradient from a set of N sampled rankings, with high computational efficiency and sample efficiency. For each sampled ranking y, PL-Rank starts by pre-computing the following values (in linear time, $O(K)$):

$$\begin{aligned} PR_{y,i} &= \sum_{k=i}^{K} \theta_k \rho_{y_k}, \quad PR_{y,d} = PR_{y,\text{rank}(d,y)+1}, \\ RI_{y,i} &= \sum_{k=1}^{\min(i,K)} \frac{PR_{y,k}}{\sum_{d' \in D \setminus y_{1:k-1}} e^{m(d')}}, \quad RI_{y,d} = RI_{y,\text{rank}(d,y)}, \\ DR_{y,i} &= \sum_{k=1}^{\min(i,K)} \frac{\theta_k}{\sum_{d' \in D \setminus y_{1:k-1}} e^{m(d')}}, \quad DR_{y,d} = DR_{y,\text{rank}(d,y)}. \end{aligned} \tag{5}$$

With these values pre-computed, gradient computation for a single ranking can also be done in linear time since:

$$\begin{aligned} \frac{\partial}{\partial m(d)} R(\pi) &= \mathbb{E}_y \Big[PR_{y,d} + e^{m(d)} \Big(\rho_d DR_{y,d} - RI_{y,d} \Big) \Big] \\ &\approx \frac{1}{N} \sum_{i=1}^{N} \Big[PR_{y^{(i)},d} + e^{m(d)} \Big(\rho_d DR_{y^{(i)},d} - RI_{y^{(i)},d} \Big) \Big], \end{aligned} \tag{6}$$

where the second line shows the gradient estimator of PL-Rank. Because this has to be done for each of the N sampled rankings, and each ranking is created through (top-K) sorting, the complexity of the entire algorithm is $O(N \cdot (K + D))$. Thereby, PL-Rank efficiently approximates the first-order derivatives from sampled rankings.

3 METHOD: APPROXIMATING THE HESSIAN

The goal of this work is to make GBDTs work effectively for stochastic LTR. As discussed earlier, GBDTs require both first- and second-order derivatives, and to the best of our knowledge, the stochastic LTR field is currently missing an algorithm for efficiently estimating the second-order derivatives. Accordingly, as PL-Rank

[1] Technically, [27] is an exception to this observation.

has done for the first-order derivatives, we contribute a novel algorithm to computationally-efficiently estimate the second-order derivatives of ranking metrics w.r.t. PL ranking models.

Our first step is to discover a formulation of the second-order derivatives. Luckily, the first-order derivatives in Eq. 4 provides a useful starting point for the derivation. Due to space limitations, we omit the intermediate steps of the derivation and only display the result in Figure 1. However, we believe that with the same starting point (Eq. 4) and knowledge of the result (Figure 1), one can reproduce a derivation with moderate effort.

The full formula in Figure 1 does not immediately reveal an efficient manner of computing it. Inspired by the PL-Rank approach, we aim to reformulate the second-order derivatives in terms that can be computed with minimal computational complexity. Conveniently, we can reuse the pre-computed terms for PL-Rank in Eq. 5 for the second-order derivatives. In addition, we introduce three more terms that can be pre-computed; the first is the cumulative sum of all reciprocal scoring denominators:

$$DN_{y,i} = \sum_{k=1}^{\min(i,K)} \frac{1}{\sum_{d' \in D \setminus y_{1:k-1}} e^{m(d')}}, \quad DN_{y,d} = DN_{y,\text{rank}(d,y)}. \tag{7}$$

The other two terms are variations on RI and DR where the denominators have been squared:

$$\begin{aligned} RS_{y,i} &= \sum_{k=1}^{\min(i,K)} \frac{PR_{y,k}}{(\sum_{d' \in D \setminus y_{1:k-1}} e^{m(d')})^2}, \quad RS_{y,d} = RS_{y,\text{rank}(d,y)}, \\ DS_{y,i} &= \sum_{k=1}^{\min(i,K)} \frac{\theta_k}{(\sum_{d' \in D \setminus y_{1:k-1}} e^{m(d')})^2}, \quad DS_{y,d} = DS_{y,\text{rank}(d,y)}. \end{aligned} \tag{8}$$

Importantly, these three terms can be computed with linear complexity, $O(K)$, for a single ranking. Furthermore, we introduce two functions, one that has to be multiplied with $e^{m(d)}$ and the other with $e^{m(d)^2}$:

$$\begin{aligned} X_1(y,d) &= \Big(1 + \mathbb{1}[d \in y]\Big)\Big(\rho_d DR_{y,d} - RI_{y,d}\Big) - DN_{y,d} PR_{y,d}, \\ X_2(y,d) &= \Big(RS_{y,d} - \rho_d DS_{y,d}\Big) - DN_{y,d}\Big(\rho_d DR_{y,d} - RI_{y,d}\Big), \end{aligned} \tag{9}$$

where $\mathbb{1}[d \in y]$ indicates whether d is ranked in the top-K in y. Since the previous terms can be pre-computed and stored in $O(K)$, subsequently, the computation of X_1 and X_2 is constant per document, therefore, computing them for *all* documents gives a complexity of $O(K + D)$. Finally, the hessian can be expressed in these terms and estimated from N sampled rankings accordingly:

$$\begin{aligned} \frac{\partial^2}{\partial m^2(d)} R(\pi) &= \mathbb{E}_y \Big[PR_{y,d} + e^{m(d)} X_1(y,d) + e^{m(d)^2} X_2(y,d) \Big], \\ &\approx \frac{1}{N} \sum_{i=1}^{N} \Big[PR_{y^{(i)},d} + e^{m(d)} X_1(y^{(i)},d) + e^{m(d)^2} X_2(y^{(i)},d) \Big]. \end{aligned} \tag{10}$$

Again, we see that after pre-computing the terms, the operation per document and ranking is constant. As a result, computing the second-order derivative for every document, i.e., the Hessian, with our algorithm has the same complexity as the underlying sorting procedure used to sample rankings: $O(N \cdot (K + D))$.

Our Hessian estimator integrates well with the existing PL-Rank algorithm since it reuses all of its pre-computed terms. Similarly, the rankings sampled for estimating the first-order derivatives can also be reused for estimating the Hessian. Therefore, we argue that it can be seen as a natural extension of the PL-Rank that makes it relevant to GBDTs. Accordingly, just as PL-Rank, our Hessian estimator can also be used for optimizing the second-order derivatives of other types of ranking metrics, i.e., exposure-fairness [18]. The remainder of this paper evaluates whether our Hessian estimator improves GBDTs optimization for relevance ranking metrics.

Table 1: NDCG@K reached by three stochastic LTR methods and LambdaMART. Results are means over five runs, with standard deviations shown in brackets. Best performance by a stochastic method shown in bold.

	Method	$K = 5$	$K = 10$
Yahoo!	Neural Network	0.7566 (0.0006)	0.7867 (0.0003)
	GBDT w/. Hessian	**0.7614 (0.0012)**	**0.7905 (0.0009)**
	GBDT w/o. Hessian	0.7233 (0.0004)	0.7620 (0.0012)
	LambdaMART	0.7796	0.8035
MSLR	Neural Network	0.4810 (0.0023)	0.4895 (0.0017)
	GBDT w/. Hessian	**0.4844 (0.0072)**	**0.4941 (0.0054)**
	GBDT w/o. Hessian	0.4313 (0.0032)	0.4429 (0.0033)
	LambdaMART	0.4889 (0.0014)	0.4955 (0.0033)
Istella	Neural Network	**0.6117 (0.0016)**	**0.6533 (0.0011)**
	GBDT w/. Hessian	0.5942 (0.0023)	0.6376 (0.0019)
	GBDT w/o. Hessian	0.5734 (0.0057)	0.5777 (0.0034)
	LambdaMART	0.6554	0.7018

4 EXPERIMENTAL SETUP

Our experiments answer the following two research questions:

RQ1 Do our estimated Hessians provide an increase in NDCG performance for stochastic LTR with GBDTs?

RQ2 Do GBDTs with our estimated Hessians reach higher levels of NDCG performance than NNs for stochastic LTR?

Our experiments compare the follow three models: (i) NNs optimized with PL-Rank [18, 19]; (ii) GBDTs optimized with gradients from PL-Rank and Hessians from our novel estimator (Eq. 10); and (iii) GBDTs optimized gradients from PL-Rank but without a Hessian (set to a value of 1 for all documents). Finally, to investigate the gap between stochastic and deterministic LTR, we also report the performance of the XGBoost built-in LambdaMART.

Our experiments use three publicly-available LTR datasets: Yahoo! Webscope-Set1 [7], MSLR-Web30K [22], and Istella [9]. We consider two ranking lengths $K = 5$ and $K = 10$ and optimize DCG@K; our evaluation metric is normalized DCG@K (NDCG@K); each reported result is the mean of five independent runs. The NNs were optimized with a PyTorch [20] implementation of PL-Rank (Eq. 6), the GBDTs were implemented with XGBoost [8]. For hyperparameter tuning, we used Optuna [1]; for a fair comparison, each method was given exactly 12 hours of hyperparameter tuning on the validation set, per setting. Our experimental implementation is publicly available at https://github.com/jkang98/2024-SIGIR-XGBoost-PL-Rank.

5 RESULTS

5.1 Importance of Hessian estimation

We first consider **RQ1**: *whether our estimated Hessians provide an increase in NDCG performance for stochastic LTR with GBDTs.* Table 1

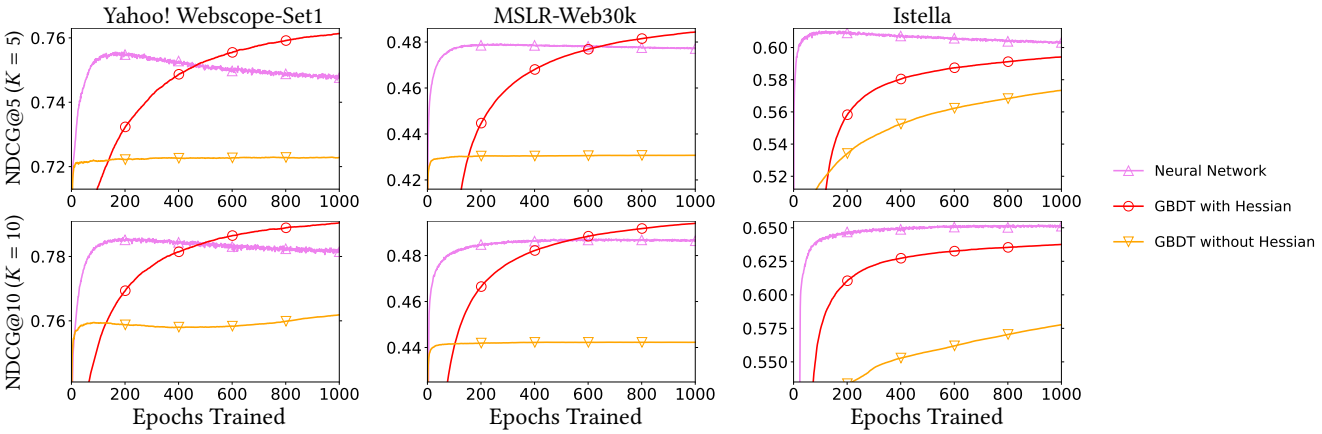


Figure 2: Learning curves of stochastic LTR methods in terms of NDCG@K over a thousand epochs.

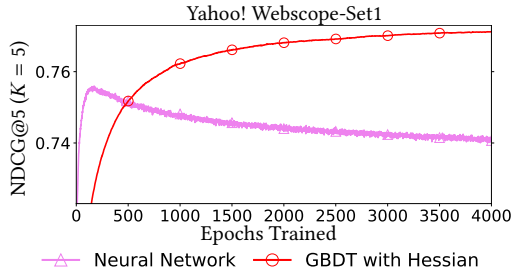


Figure 3: Convergence analysis on the Yahoo! dataset.

displays the NDCG@5/10 reached by GBDTs with and without an estimated Hessian. We see that, across all datasets and both ranking lengths, GBDTs without a Hessian reach a considerably lower NDCG than with our estimated Hessian. The differences in performance range from 0.02 to 0.06, where the lower value of 0.02 is already a very large difference for the NDCG metric. Additionally, Figure 2 displays the learning curves of both methods. Very clearly, GBDTs without Hessian consistently converge at suboptimal performance, with the exception on Istella dataset, this convergence takes place very quickly. In contrast, with our estimated Hessians, GBDTs reach much higher performance and continue improving over many more epochs.

We answer **RQ1** accordingly: Our estimated Hessian enables stochastic GBDT models to reach considerably higher NDCG than without a Hessian. This demonstrates both the importance of Hessians for stochastic LTR and the effectiveness of our novel estimator.

5.2 Comparison of GBDT and NN models

Next, we address **RQ2**: *whether GBDTs with our estimated Hessian reach higher levels of NDCG performance than NNs.* Table 1 reveals that this is clearly the case on Yahoo! and MSLR, but on Istella NNs have higher performance. In Figure 2, we see the learning curves of both methods. Surprisingly, NNs appear to learn in fewer epochs but their performance degrades over time. In contrast, GBDTs require more epochs but do not display such unstable convergence. To further analyze convergence stability, we performed an additional experiment over 4000 epochs, the results of which can be seen in Figure 3. There, NN performance continues to degrade while the GBDTs continue to learn. Thereby, our results suggest that GBDTs provide far stabler optimization that can reach higher performance on the Yahoo! and MSLR datasets. It is unclear why similar results are not achieved on the Istella dataset; it is possible that more epochs are required for GBDTs to reach NN levels of performance there, but we could not confirm this.

Therefore, we answer **RQ2** as follows: our estimated Hessian enables GBDTs to reach performance comparable to NNs, on the Yahoo! and MSLR datasets this lead to substantial improvements in NDCG, whilst on Istella, NN performance was not reached. Additionally, we found that NNs do not converge stably and depend on early stopping for good performance. Conversely, GBDTs have very stable convergence and no performance decreases were observed.

Finally, to evaluate the gap between stochastic and deterministic LTR, we also compare with the built-in LambdaMART implementation of XGBoost in Table 1. Unsurprisingly, built-in LambdaMART still outperforms our stochastic methods; this implementation benefits from almost a decade of implementation optimization. Furthermore, while the GBDTs are optimizing distributions over rankings, the deterministic LambdaMART has a simpler task by focusing on individual rankings. Nonetheless, on the Yahoo! and MSLR datasets, GBDTs with estimated Hessian get much closer to LambdaMART performance than the NNs. In particular, the differences on MSLR between our GBDTs and LambdaMART are marginal. Thereby, we believe our GBDTs provide an important step in bridging the gap between stochastic LTR and state-of-the-art deterministic LTR.

6 CONCLUSION

In this work, we introduced the first stochastic LTR method for effectively optimizing GBDTs. We proposed a novel estimator for the second-order derivatives (i.e., the Hessian) of stochastic ranking objectives w.r.t. PL ranking models, and showed how it can be computed with minimal computational complexity. Our experimental results reveal that GBDTs have extremely poor performance when optimized without Hessian. Conversely, with our estimated Hessian, GBDTs are able to provide substantial performance gains over NNs. Furthermore, unlike NNs, GBDTs have very stable convergence that enable the performance gains. Our work brings stochastic LTR significantly closer to state-of-the-art deterministic LTR.

Acknowledgements. This work is partially supported by the Dutch Research Council (NWO), grant number VI.Veni.222.269 and project numbers 024.004.022, NWA.1389.20.183, KICH3.LTP.20.006.

REFERENCES

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Anchorage, AK, USA) *(KDD '19)*. Association for Computing Machinery, New York, NY, USA, 2623–2631.

[2] Sebastian Bruch, Shuguang Han, Michael Bendersky, and Marc Najork. 2020. A Stochastic Treatment of Learning to Rank Scoring Functions. In *Proceedings of the 13th International Conference on Web Search and Data Mining* (Houston, TX, USA) *(WSDM '20)*. Association for Computing Machinery, New York, NY, USA, 61–69.

[3] Alexander Buchholz, Jan Malte Lichtenberg, Giuseppe Di Benedetto, Yannik Stein, Vito Bellini, and Matteo Ruffini. 2022. Low-variance estimation in the Plackett-Luce model via quasi-Monte Carlo sampling. In *SIGIR 2022 Workshop on Reaching Efficiency in Neural Information Retrieval*.

[4] Chris J.C. Burges. 2010. *From RankNet to LambdaRank to LambdaMART: An Overview*. Technical Report MSR-TR-2010-82.

[5] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning* (Corvalis, Oregon, USA) *(ICML '07)*. Association for Computing Machinery, New York, NY, USA, 129–136.

[6] Soumen Chakrabarti, Rajiv Khanna, Uma Sawant, and Chiru Bhattacharyya. 2008. Structured Learning for Non-Smooth Ranking Losses. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Las Vegas, Nevada, USA) *(KDD '08)*. Association for Computing Machinery, New York, NY, USA, 88–96.

[7] Olivier Chapelle and Yi Chang. 2011. Yahoo! Learning to Rank Challenge Overview. In *Proceedings of the Learning to Rank Challenge (Proceedings of Machine Learning Research, Vol. 14)*, Olivier Chapelle, Yi Chang, and Tie-Yan Liu (Eds.). PMLR, Haifa, Israel, 1–24.

[8] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. Association for Computing Machinery, New York, NY, USA, 785–794.

[9] Domenico Dato, Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Nicola Tonellotto, and Rossano Venturini. 2016. Fast Ranking with Additive Ensembles of Oblivious and Non-Oblivious Regression Trees. *ACM Trans. Inf. Syst.* 35, 2, Article 15 (dec 2016), 31 pages.

[10] Domenico Dato, Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, and Nicola Tonellotto. 2022. The Istella22 Dataset: Bridging Traditional and Neural Learning to Rank Evaluation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 3099–3107.

[11] Fernando Diaz, Bhaskar Mitra, Michael D. Ekstrand, Asia J. Biega, and Ben Carterette. 2020. Evaluating Stochastic Rankings with Expected Exposure. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management* (Virtual Event, Ireland) *(CIKM '20)*. Association for Computing Machinery, New York, NY, USA, 275–284.

[12] Emil Julius Gumbel. 1954. Statistical Theory of Extreme Values and Some Practical Applications. *National Bureau of Standards Applied Mathematics Series. 33* (1954).

[13] Donna Harman. 2011. *Information Retrieval Evaluation*. Morgan & Claypool Publishers.

[14] Rolf Jagerman, Zhen Qin, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2022. On Optimizing Top-K Metrics for Neural Ranking Models. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 2303–2307.

[15] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.* 20, 4 (oct 2002), 422–446.

[16] Tie-Yan Liu. 2009. Learning to Rank for Information Retrieval. *Foundations and Trends® in Information Retrieval* 3, 3 (2009), 225–331.

[17] R. Duncan Luce. 1959. *Individual Choice Behavior: A Theoretical Analysis*. Wiley.

[18] Harrie Oosterhuis. 2021. Computationally Efficient Optimization of Plackett-Luce Ranking Models for Relevance and Fairness. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, Canada) *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 1023–1032.

[19] Harrie Oosterhuis. 2022. Learning-to-Rank at the Speed of Sampling: Plackett-Luce Gradient Estimation with Minimal Computational Complexity. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Madrid, Spain) *(SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 2266–2271.

[20] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc., Red Hook, NY, USA, 12.

[21] R. L. Plackett. 1975. The Analysis of Permutations. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 24, 2 (1975), 193–202.

[22] Tao Qin and Tie-Yan Liu. 2013. Introducing LETOR 4.0 Datasets. *CoRR* abs/1306.2597 (2013).

[23] Tao Qin, Tie-Yan Liu, and Hang Li. 2010. A General Approximation Framework for Direct Optimization of Information Retrieval Measures. *Information retrieval* 13 (2010), 375–397.

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

[25] Ashudeep Singh and Thorsten Joachims. 2019. *Policy learning for fairness in ranking*. Curran Associates Inc., Red Hook, NY, USA.

[26] Michael Taylor, John Guiver, Stephen Robertson, and Tom Minka. 2008. SoftRank: optimizing non-smooth rank metrics. In *Proceedings of the 2008 International Conference on Web Search and Data Mining* (Palo Alto, California, USA) *(WSDM '08)*. Association for Computing Machinery, New York, NY, USA, 77–86.

[27] Aleksei Ustimenko and Liudmila Prokhorenkova. 2020. StochasticRank: Global Optimization of Scale-Free Discrete Functions. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 9669–9679.

[28] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th International Conference on Machine Learning* (Helsinki, Finland) *(ICML '08)*. Association for Computing Machinery, New York, NY, USA, 1192–1199.

[29] Jun Xu, Tie-Yan Liu, Min Lu, Hang Li, and Wei-Ying Ma. 2008. Directly Optimizing Evaluation Measures in Learning to Rank. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Singapore, Singapore) *(SIGIR '08)*. Association for Computing Machinery, New York, NY, USA, 107–114.

[30] Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. 2007. A Support Vector Method for Optimizing Average Precision. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Amsterdam, The Netherlands) *(SIGIR '07)*. Association for Computing Machinery, New York, NY, USA, 271–278.