On Optimizing Top-K Metrics for Neural Ranking Models

Rolf Jagerman
Google Research
jagerman@google.com

Zhen Qin
Google Research
zhenqin@google.com

Xuanhui Wang
Google Research
xuanhui@google.com

Michael Bendersky
Google Research
bemike@google.com

Marc Najork
Google Research
najork@google.com

ABSTRACT

Top-K metrics such as NDCG@K are frequently used to evaluate ranking performance. The traditional tree-based models such as LambdaMART, which are based on Gradient Boosted Decision Trees (GBDT), are designed to optimize NDCG@K using the LambdaRank losses. Recently, there is a good amount of research interest on *neural* ranking models for learning-to-rank tasks. These models are fundamentally different from the decision tree models and behave differently with respect to different loss functions. For example, the most popular ranking losses used in neural models are the Softmax loss and the GumbelApproxNDCG loss. These losses do not connect to top-K metrics such as NDCG@K naturally. It remains a question on how to effectively optimize NDCG@K for neural ranking models. In this paper, we follow the LambdaLoss framework and design novel and theoretically sound losses for NDCG@K metrics, while the original LambdaLoss paper can only do so using an unsound heuristic. We study the new losses on the LETOR benchmark datasets and show that the new losses work better than other losses for neural ranking models.

CCS CONCEPTS

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

KEYWORDS

Learning to Rank; Ranking Metric Optimization; LambdaLoss

ACM Reference Format:
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), July 11–15, 2022, Madrid, Spain.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3477495.3531849

1 INTRODUCTION

Ranking losses play a crucial role in Learning-to-Rank (LTR) techniques [14, 28]. Before neural ranking models, the most effective models are Gradient Boosted Decision Trees (GBDT). One such important approach is LambdaMART [5], which uses the LambdaRank loss and has long dominated the LTR field. Recently, neural ranking models have become popular for LTR tasks. These neural models are fundamentally different from the tree-based models and have different behaviors with respect to ranking losses. For example, popular ranking losses to optimize neural ranking models are the Softmax loss [2, 16, 20] and GumbelApproxNDCG loss [1, 3, 18], which are usually sub-optimal when applied to tree-based models [2, 5]. Thus it is worth studying ranking losses with respect to ranking metric optimization for neural ranking models.

Top-K metrics such as NDCG@K are more commonly used to evaluate ranking performance than their full-list counterparts, because they reflect the utility of a real-world ranking system better due to their affinity to user behaviors, in which users are more likely to focus on results at top ranks only. Thus optimizing top-K metrics is practically important. Although this has been studied in tree-based models, it remains a question on how to do this effectively for neural ranking models.

Throughout the paper, we focus on NDCG@K as a representative example for top-K metrics. Directly optimizing NDCG is challenging because ranking metrics are discontinuous and flat everywhere. Optimizing NDCG@K can be even more challenging because the truncation of the list and the top-K documents per query can change dynamically during the training procedure, making it hard to define a loss over a fixed set of documents.

Existing work on ranking metric optimization have mainly focused on NDCG without a cutoff (e.g., SoftNDCG [22], SmoothNDCG [7], ApproxNDCG [18]). However, these approaches do not generalize to NDCG@K directly. In contrast, the LambdaRank [6] loss can be extended to NDCG@K, which will be referred to as LambdaRank@K throughout this paper. Such a loss is based the delta change of NDCG@K when two documents are swapped. A natural question that arises now is whether we can use LambdaRank@K to optimize NDCG@K for *neural ranking models*. As we show in this paper, direct application of LambdaRank@K to neural ranking models is not effective. Furthermore, the recently proposed LambdaLoss [26] framework can also be extended to NDCG@K using a similar heuristic as what was used in LambdaRank@K. Unfortunately, such a heuristic is theoretically unsound and, as we show in this paper, is not an effective strategy for optimizing top-K metrics. Finally, other ranking losses used in neural models such as the Softmax loss are also not designed to optimize NDCG@K and thus are not very effective either.

In this paper, we follow the LambdaLoss framework and derive a novel and sound loss for NDCG@K, named LambdaLoss@K. Unlike LambdaRank@K which sets a zero weight for pairs where both are ranked beyond the rank K, the new LambdaLoss@K proposes a

SIGIR '22, July 11–15, 2022, Madrid, Spain

ACM ISBN 978-1-4503-8732-3/22/07.
https://doi.org/10.1145/3477495.3531849

novel correction multiplier to appropriately weigh those pairs. This correction multiplier is based on NDCGCost@K, a modification of the NDCGCost from LambdaLoss [26], which can be used to appropriately bound NDCG@K.

To validate the effectiveness of the new LambdaLoss@K, we conduct experiments on three LETOR bechmark datasets [17]. Our experiments show that our new loss can outperform the commonly used losses for neural ranking models. It also outperforms LambdaRank@K and the original LambdaLoss@K in [26] when optimizing NDCG@K. Furthermore, the LamdaLoss@K is better than the LambdaLoss in general, validating our theoretical results.

2 RELATED WORK

LTR is a widely studied field where numerous ranking losses have been studied. Several notable losses include Pairwise Logistic [4] (also called RankNet), ListNet [27], Softmax [2, 16]. Furthermore, ranking metric optimization has been tackled from various perspectives, for example the works on LambdaRank [5, 6], SoftNDCG [22], SmoothNDCG [7], ApproxNDCG [18] and GumbelApproxNDCG [3]. Finally, one of the most recent works, LambdaLoss [26], uses the ideas from LambdaRank to develop a theoretically sound framework for ranking metric optimization and is considered a strong loss for optimizing neural LTR models. Our work builds on the LambdaLoss work but extends it by considering a new way to incorporate cutoff-based metrics such as NDCG@K. This is a novel contribution over the original LambdaLoss paper which was only able to incorporate top-K metrics via a heuristic.

There are several existing works that has investigated top-K ranking metric optimization. First, Oosterhuis and de Rijke [15] consider the problem of unbiased LTR from interaction logs with top-K cutoffs. To be specific, they study item selection bias and uses a stochastic ranking policy to overcome this bias, meaning it is situated in between online and counterfactual LTR [9]. Our work is different in that we develop a novel loss for optimizing top-K ranking metrics from *relevance labels* and we do not consider the problem of item selection bias or unbiased LTR from interaction logs (e.g., [11, 24, 25]). Second, Lee et al. [12] looks at top-K metric optimization in the context of factorization methods for recommendation systems, which is different from our work since we consider the problem of neural LTR. Finally, we note that the LambdaRank and LambdaMART frameworks [5] are capable of optimizing top-K metrics. Such methods have enjoyed considerable success for decision-tree based models [13, 19]. However, as we will see in the results of this paper, applying the LambdaRank loss to *neural* LTR models does not work very well, a phenomenon that was also reported in [20].

3 RANKING METRICS

Ranking metrics are commonly used to evaluate Information Retrieval (IR) systems. Broadly speaking, ranking metrics express how well a ranking induced from item scores $s \in \mathbb{R}^n$ matches a ranking induced from relevance labels $y \in \mathbb{R}^n$. Let π denote the 1-based ranks of a set of item scores s after sorting them in descending order:

$$\pi(s_i \mid s) = \text{ 1-based rank of the } i^{\text{th}} \text{ score } s_i. \tag{1}$$

For simplicity we will write $\pi_i = \pi(s_i \mid s)$ where s can be inferred from the context. Using this formulation, we can write the definitions of DCG and NDCG [10]:

$$DCG(s, y) = \sum_{i=1}^{n} \frac{G(y_i)}{D(\pi_i)}, \tag{2}$$

$$NDCG(s, y) = \frac{DCG(s, y)}{DCG(y, y)}, \tag{3}$$

where G is a gain function such as $G(y_i) = 2^{y_i} - 1$ and D is a rank discount function such as $D(r) = \log_2(1 + r)$. Correspondingly, it is possible to define the top-K variants of these metrics:

$$DCG@K(s, y) = \sum_{i \,:\, \pi_i \leq K} \frac{G(y_i)}{D(\pi_i)}, \tag{4}$$

$$NDCG@K(s, y) = \frac{DCG@K(s, y)}{DCG@K(y, y)}, \tag{5}$$

Ranking metrics are traditionally considered hard to optimize for directly, as the rank function π makes ranking metrics discontinuous and flat everywhere. Instead, a common approach taken in LTR is to optimize a ranking loss, which acts as a surrogate for a ranking metric.

4 POPULAR RANKING LOSSES

Several popular ranking losses that are used today are pairwise losses such as pairwise logistic loss [4] (also called the RankNet loss) and the Softmax cross-entropy loss [16]:

$$RankNet(s, y) = \sum_{(i,j) \,:\, y_i > y_j} \log(1 + \exp(-\sigma(s_i - s_j))) \tag{6}$$

$$Softmax(s, y) = -\sum_{i=1}^{n} y_i \cdot \log \frac{\exp(s_i)}{\sum_{j=1}^{n} \exp(s_j)} \tag{7}$$

where σ is a hyper-parameter and we set it to 1 in this paper. Although these ranking losses permit optimization via gradient descent, they are only loosely connected to the ranking metrics that we are really interested in optimizing. To close this gap, existing work has looked at approximate metric optimization. For example, the work on ApproxNDCG [18] proposes a differentiable approximation to π:

$$\pi(s_i \mid s) \approx \tilde{\pi}(s_i \mid s) = 1 + \sum_{j=1}^{n} \text{sigmoid}(s_j - s_i). \tag{8}$$

Plugging $\tilde{\pi}$ into the definition of DCG and NDCG of Eqs. 2 and 3 gives rise to the approximate losses ApproxDCG and ApproxNDCG. Furthermore, recent work [3] has shown that to effectively optimize an approximate ranking loss, it is important to apply Gumbel sampling to the scores. This results in the GumbelApproxNDCG loss [3], which is defined as an application of ApproxNDCG on scores that are corrupted by Gumbel noise:

$$\text{GumbelApproxNDCG}(s, y) = \text{ApproxNDCG}(s + g, y) \tag{9}$$

$$g_i \sim \text{Gumbel}(\alpha, \beta), \tag{10}$$

where $\text{Gumbel}(\alpha, \beta)$ is the standard Gumbel distribution with typical parameters $\alpha = 0$, $\beta = 1$.

5 OPTIMIZING NDCG@K

Existing work, such as ApproxNDCG has focused on constructing an optimizable loss directly from the definition of a metric by replacing ranks with transformed scores. However, how to extend such work to top-K metrics is not immediately clear. An alternative approach is taken by LambdaRank [6], which is an extension of the RankNet loss as defined in Eq. 6 that scales the gradient of each pair of scores (s_i, s_j) by the difference in a ranking metric if those scores were swapped, also called the *lambda weight* $\Delta_{i,j}$. This approach works on any ranking metric, including top-K ranking metrics. For NDCG, this delta is

$$\Delta_{i,j} = \left|G(y_i) - G(y_j)\right| \cdot \left|\frac{1}{D(\pi_i)} - \frac{1}{D(\pi_j)}\right| = \left|G(y_i) - G(y_j)\right| \cdot \delta_{i,j}, \tag{11}$$

normalized by the ideal DCG, which is a constant for each query and is omitted from our formulation for clarity.

To apply LambdaRank to NDCG@K, a minor modification to $\delta_{i,j}$ is sufficient:

$$\delta_{i,j}@K = \left|\frac{\mathbb{1}[\pi_i \leq K]}{D(\pi_i)} - \frac{\mathbb{1}[\pi_j \leq K]}{D(\pi_j)}\right|. \tag{12}$$

Note that this formulation introduces sparsity: many $\delta_{i,j}@K$ end up being 0. It has been shown that LambdaRank@K works effectively for tree-based models [5]. Although the empirical effectiveness of LambdaRank is well documented [13, 19, 20], the method remains a heuristic and the underlying loss being optimized is unknown.

More recently, the LambdaLoss framework [26] was introduced and proposes a theoretically-sound framework for Lambda-based losses such as LambdaRank. In a sense, LambdaLoss is very similar to LambdaRank when optimizing NDCG. It makes a minor adjustment to $\delta_{i,j}$:

$$\delta_{i,j} = \left|\frac{1}{D(|\pi_i - \pi_j|)} - \frac{1}{D(|\pi_i - \pi_j| + 1)}\right|. \tag{13}$$

The derivation of LambdaLoss is based on the concept of NDCGCost, a cost version of the NDCG metric:

$$\text{NDCGCost}(s_i, y_i) = G(y_i)\left(1 - \frac{1}{D(\pi_i)}\right). \tag{14}$$

The original LambdaLoss paper provided a heuristic approach to use the same loss for top-K variant NDCG@K. The authors propose the following top-K variant of $\delta_{i,j}$:

$$\delta_{i,j}@K = \mathbb{1}[\pi_i \leq k \text{ or } \pi_j \leq k]\delta_{i,j}. \tag{15}$$

In other words, this uses the full $\delta_{i,j}$ when either item i or item j is in the top-K ranks, and sets it to 0 otherwise. This formulation is a heuristic providing no guarantees, and is only loosely connected to NDCG@K. In fact, as we will see in the experiments, this loss is not effective at optimizing NDCG@K.

Here is where we deviate from the standard LambdaLoss to formulate our novel contribution LambdaLoss@K. Instead of setting items beyond the top-K to 0, we weigh them, leading to the following definition of $\delta_{i,j}@K$:

$$\delta_{i,j}@K = \begin{cases} \delta_{i,j}\mu_{i,j} & \text{if } \pi_i > K \text{ or } \pi_j > K \\ \delta_{i,j} & \text{else} \end{cases}, \tag{16}$$

where $\mu_{i,j}$ is a correction multiplier defined as

$$\mu_{i,j} = \frac{1}{1 - \frac{1}{D(\max(\pi_i, \pi_j))}}. \tag{17}$$

Note that for NDCG without a cutoff (i.e. $K = \infty$), this formulation is identical to the original LambdaLoss.

To derive this new definition of $\delta_{i,j}@K$, we start with the definition of NDCGCost@K which follows from NDCG@K from Eq. 5:

$$\text{NDCGCost@}K(s_i, y_i) = \begin{cases} G(y_i)\left(1 - \frac{1}{D(\pi_i)}\right) & \text{if } \pi_i \leq K \\ G(y_i) & \text{else} \end{cases}. \tag{18}$$

For $\pi_i \leq K$, we get:

$$\text{NDCGCost@}K(s_i, y_i) = G(y_i)\left(1 - \frac{1}{D(\pi_i)}\right) \tag{19a}$$

$$= G(y_i)\sum_{j=1}^{n} \delta_{i,j}\,\mathbb{1}[s_i < s_j] \tag{19b}$$

$$= G(y_i)\left(\sum_{j:\pi_j \leq K} \delta_{i,j}\,\mathbb{1}[s_i < s_j] + \sum_{j:\pi_j > K} \delta_{i,j}\,\mathbb{1}[s_i < s_j]\right) \tag{19c}$$

$$= G(y_i)\left(\sum_{j:\pi_j \leq K} \delta_{i,j}\,\mathbb{1}[s_i < s_j] + \sum_{j:\pi_j > K} \delta_{i,j}\mu_{i,j}\,\mathbb{1}[s_i < s_j]\right), \tag{19d}$$

where Eq. 19b follows from the definition of $\delta_{i,j}$ of LambdaLoss and Eq. 19d holds because $\mathbb{1}[s_i < s_j]$ is always zero in the second sum. Next, for $\pi_i > K$, we get:

$$\text{NDCGCost@}K(s_i, y_i) = G(y_i)\left(1 - \frac{1}{D(\pi_i)}\right)\frac{1}{\left(1 - \frac{1}{D(\pi_i)}\right)} \tag{20a}$$

$$= G(y_i)\sum_{j=1}^{n} \frac{1}{\left(1 - \frac{1}{D(\pi_i)}\right)}\delta_{i,j}\,\mathbb{1}[s_i < s_j] \tag{20b}$$

$$= G(y_i)\left(\sum_{j:\pi_j \leq \pi_i}^{n} \mu_{i,j}\delta_{i,j}\,\mathbb{1}[s_i < s_j] + \sum_{j:\pi_j > \pi_i}^{n} \mu_{i,j}\delta_{i,j}\,\mathbb{1}[s_i < s_j]\right), \tag{20c}$$

where we follow the similar reasoning as with Eq. 19. Overall, the result is that for both cases $\pi_i > K$ and $\pi_j > K$ we always obtain $\mu_{i,j}$ as an extra multiplier for $\delta_{i,j}$, resulting in Eq. 16. Note that $\delta_{i,j}@K$ is symmetric and can be bounded using the same method that was used in the original LambdaLoss paper.

6 EXPERIMENTS

To validate the effectiveness of the proposed LambdaLoss@K, we experiment using a standard supervised LTR setup with 3 public benchmark datasets: Web30K [17], Yahoo [17], and Istella [8]. Each data set contains a collection of queries and corresponding documents where query-document pairs are represented as feature vectors. Furthermore each query-document pair has a corresponding relevance label $\in \{0, 1, 2, 3, 4\}$, where higher values indicate higher relevance of the document to the query.

For each dataset we train neural LTR models using TF-Ranking [16]. For all input features, we applied a log1p transformation as in [20].

Table 1: Comparison of several baseline losses against the novel LambdaLoss@K. Bold numbers indicate the best performance. $^{\blacktriangle}$ and $^{\blacktriangledown}$ indicate statistically significant (t-test, $p < 0.05$) differences compared to LambdaLoss.

	WEB30K			Istella			Yahoo		
	NDCG@1	NDCG@5	NDCG	NDCG@1	NDCG@5	NDCG	NDCG@1	NDCG@5	NDCG
Pairwise	45.31$^{\blacktriangledown}$	45.38$^{\blacktriangledown}$	70.67$^{\blacktriangledown}$	67.76$^{\blacktriangledown}$	64.77$^{\blacktriangledown}$	79.95$^{\blacktriangledown}$	66.47$^{\blacktriangledown}$	69.84$^{\blacktriangledown}$	82.66$^{\blacktriangledown}$
Softmax	47.33	46.69$^{\blacktriangledown}$	71.45	69.45$^{\blacktriangledown}$	66.70$^{\blacktriangledown}$	81.02$^{\blacktriangledown}$	67.52	70.62	83.15
Gumbel	48.20	46.85	71.31$^{\blacktriangledown}$	71.27	66.93$^{\blacktriangledown}$	80.89$^{\blacktriangledown}$	67.75	70.30$^{\blacktriangledown}$	82.91$^{\blacktriangledown}$
LambdaLoss	47.74	47.11	71.55	71.17	67.62	81.51	67.92	70.78	83.17
LambdaLoss@1	**48.50**$^{\blacktriangle}$	**47.40**$^{\blacktriangle}$	**71.65**	**71.72**$^{\blacktriangle}$	**67.90**$^{\blacktriangle}$	**81.74**$^{\blacktriangle}$	**68.06**	70.71	83.13
LambdaLoss@5	47.34	47.09	71.57	71.19	67.61	81.56	67.94	**70.82**	**83.17**

Table 2: Comparing LambdaRank@K and LambdaLoss@K with respect to NDCG@5. Significant differences compared to LambdaLoss@5 V1 are indicated the same as Table 1.

	WEB30K	Istella	Yahoo
LambdaRank	46.87	67.55$^{\blacktriangle}$	70.08
LambdaRank@1	45.35$^{\blacktriangledown}$	65.79$^{\blacktriangledown}$	69.46$^{\blacktriangledown}$
LambdaRank@5	46.68	67.82$^{\blacktriangle}$	70.38
LambdaLoss@5 V1	46.76	66.86	70.34
LambdaLoss	47.11	67.62$^{\blacktriangle}$	70.78$^{\blacktriangle}$
LambdaLoss@1	**47.40**$^{\blacktriangle}$	**67.90**$^{\blacktriangle}$	70.71$^{\blacktriangle}$
LambdaLoss@5	47.08	67.61$^{\blacktriangle}$	**70.82**$^{\blacktriangle}$

The architecture of the model is a basic feedforward neural network with hidden layers [1024, 512, 256], where dropout is set to 0.5, 0.5, and 0.8 and batch normalization are used with batch normalization momentum as 0.999. These parameters are tuned for the baseline models. We only vary the learning rate $\in \{0.01, 0.1, 1, 10\}$ for different runs.

To evaluate the trained ranking models we use the test split of each dataset. We report our comparison on NDCG@1, NDCG@5, and NDCG metrics.

6.1 Comparison of Ranking Losses

We compare the pairwise RankNet loss, the Softmax loss and the GumbelApproxNDCG loss with our newly proposed LambdaLoss@K losses: LambdaLoss@1 LambdaLoss@5. Note that without top-K truncation, the LambdaLoss is the same as the original paper [26]. Our main results are shown in Table 1. From this table, we can make the following observations:

First, let us take a look at the three baselines: the Pairwise RankNet loss, the Softmax loss and the GumbelApproxNDCG loss. Our findings here show that when evaluating using NDCG@1 or NDCG@5, the GumbelApproxNDCG loss is a strong contender. This matches our intuition since, unlike the RankNet or Softmax loss, GumbelApproxNDCG actually specifically optimizes for NDCG, so it should come as no surprise that it outperforms the other baseline losses there. Interestingly, the Softmax loss performs better across datasets on NDCG without cutoff. Unlike GumbelApproxNDCG, the Softmax loss does not specifically optimize for NDCG so this finding is somewhat unexpected. One possible explanation for this phenomenon is that, like most LTR losses, the GumbelApproxNDCG loss is non-convex and thus easily gets stuck in local optima. This problem of non-convexity for the GumbelApprox losses has also been documented in the literature [3, 21, 23].

Second, let us direct our attention to the various versions of the LambdaLoss. Note that the LambdaLoss is the original loss and the LambdaLoss@1 and LambdaLoss@5 are our novel losses. What is clear from these results is that LambdaLoss@1 is a very strong loss and in fact outperforms all other losses on the NDCG metrics for both the Web30K and Istella datasets. Furthermore, we see that LambdaLoss@5 is also a strong loss, and in fact achieves the best performance for the NDCG@5 metric on the Yahoo dataset. This is an encouraging finding as it confirms our theoretical intuition: optimizing for a specific metric can result in better performance on that metric. Another particularly interesting thing to note here is that the original LambdaLoss does not perform the best on the NDCG metric without cutoff, despite optimizing for that metric. In fact, either the LambdaLoss@1 and LambdaLoss@5 always outperforms the LambdaLoss on this metric. This indicates that performing top-K truncation during optimization may be beneficial, even when evaluating on metrics that have no cutoff. Overall, we demonstrate that targeting ranking metrics such as NDCG@K through optimizing LambdaLoss at various cutoffs can lead to strong performance improvements across various metrics.

6.2 Top-K Losses

We then compare the LambdaRank@K, the original LambdaLoss@K (based on Eq. 15, denoted as V1 in the table) and the newly proposed LambdaLoss@K on the 3 data sets based on the NDCG@5 metrics. The results are summarized in Table 2. From this table, we can see that the LambdaRank@K is not effective for neural ranking models. They are worse than the non-truncated version LambdaRank. In contrast, LambdaLoss@K is not only better than LambdaRank, but also more effective than their non-trucated version.

6.3 Parameter Study

The only parameter we vary is the learning rate. On the Web30K data, we show the impact of learning rates on the 4 methods. For brevity, we omit the results on the other 2 datasets as their behaviors are similar. From this figure, we can see that the learning rate plays a big role for the performance. We also see that LambdaLoss@K

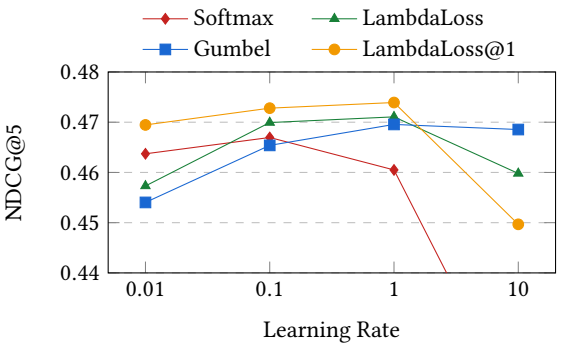

Figure 1: The impact of learning rates on different losses measured by NDCG@5 on WEB30K.

is better than other baseline on a large range of the learning rate values in Figure 1.

7 CONCLUSION

In this paper, we introduce a novel and theoretically sound LambdaLoss to optimize top-K metrics, called LambdaLoss@K. The new loss is able to work well with neural ranking models, in contrast to the LambdaRank loss for NDCG@K. We also show that such a loss is better than the commonly used Softmax and Gumbel-ApproxNDCG losses on the LETOR benchmark datasets. This is an encouraging result, as it suggests that a simple modification to an existing loss framework such as LambdaLoss can produce state-of-the-art results for metrics that are highly practically relevant. Meanwhile, we found that the LambdaLoss@K can work better on the full NDCG than LambdaLoss, a novel finding that inspires us to try various top-K ranking loss surrogates, even for ranking problems that are evaluated on untruncated NDCG. A possible direction for future work is extending the LambdaLoss@K to incorporate other Top-K ranking metrics besides NDCG@K. Our work has shown the efficacy of LambdaLoss@K for NDCG@K, but it is not immediately clear how to extend this to ranking metrics such as Recall@K or Precision@K, which we leave as future work. Furthermore, the model being optimized is a simple feed-forward neural network. How the proposed losses performs under different architectures is left as future work.

REFERENCES

[1] 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*. 61–69.

[2] Sebastian Bruch, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2019. An Analysis of the Softmax Cross Entropy Loss for Learning-to-Rank with Binary Relevance. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR '19)*. 75–78.

[3] Sebastian Bruch, Masrour Zoghi, Michael Bendersky, and Marc Najork. 2019. Revisiting approximate metric optimization in the age of deep neural networks. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1241–1244.

[4] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*. 89–96.

[5] Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning* 11, 23-581 (2010), 81.

[6] Christopher J. C. Burges, Robert Ragno, and Quoc Viet Le. 2006. Learning to Rank with Nonsmooth Cost Functions. In *Proceedings of the 19th International Conference on Neural Information Processing Systems (NIPS'06)*. 193–200.

[7] Olivier Chapelle and Mingrui Wu. 2010. Gradient descent optimization of smoothed information retrieval metrics. *Information retrieval* 13, 3 (2010), 216–235.

[8] Domenico Dato, Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Nicola Tonellotto, and Rossano Venturini. 2016. Fast ranking with additive ensembles of oblivious and non-oblivious regression trees. *ACM Transactions on Information Systems (TOIS)* 35, 2 (2016), 1–31.

[9] Rolf Jagerman, Harrie Oosterhuis, and Maarten de Rijke. 2019. To model or to intervene: A comparison of counterfactual and online learning to rank from user interactions. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*. 15–24.

[10] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002), 422–446.

[11] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased learning-to-rank with biased feedback. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. 781–789.

[12] Hyunsung Lee, Sangwoo Cho, Yeongjae Jang, Jaekwang Kim, and Honguk Woo. 2021. Differentiable ranking metric using relaxed sorting for top-k recommendation. *IEEE Access* 9 (2021), 114649–114658.

[13] Pan Li, Zhen Qin, Xuanhui Wang, and Donald Metzler. 2019. Combining decision trees and neural networks for learning-to-rank in personal search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2032–2040.

[14] Tie-Yan Liu. 2009. Learning to Rank for Information Retrieval. *Found. Trends Inf. Retr.* 3, 3 (mar 2009), 225–331.

[15] Harrie Oosterhuis and Maarten de Rijke. 2020. Policy-aware unbiased learning to rank for top-k rankings. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 489–498.

[16] Rama Kumar Pasumarthi, Sebastian Bruch, Xuanhui Wang, Cheng Li, Michael Bendersky, Marc Najork, Jan Pfeifer, Nadav Golbandi, Rohan Anil, and Stephan Wolf. 2019. Tf-ranking: Scalable tensorflow library for learning-to-rank. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2970–2978.

[17] Tao Qin and Tie-Yan Liu. 2013. Introducing LETOR 4.0 Datasets. *CoRR* abs/1306.2597 (2013). http://arxiv.org/abs/1306.2597

[18] Tao Qin, Tie-Yan Liu, and Hang Li. 2010. A general approximation framework for direct optimization of information retrieval measures. *Information retrieval* 13, 4 (2010), 375–397.

[19] Zhen Qin, Suming J. Chen, Donald Metzler, Yongwoo Noh, Jingzheng Qin, and Xuanhui Wang. 2020. Attribute-Based Propensity for Unbiased Learning in Recommender Systems: Algorithm and Case Studies. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2359–2367.

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

[21] Pradeep Ravikumar, Ambuj Tewari, and Eunho Yang. 2011. On NDCG consistency of listwise ranking methods. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, 618–626.

[22] 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*. 77–86.

[23] Hamed Valizadegan, Rong Jin, Ruofei Zhang, and Jianchang Mao. 2009. Learning to rank by optimizing ndcg measure. *Advances in neural information processing systems* 22 (2009).

[24] Xuanhui Wang, Michael Bendersky, Donald Metzler, and Marc Najork. 2016. Learning to Rank with Selection Bias in Personal Search. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '16)*. 115–124.

[25] Xuanhui Wang, Nadav Golbandi, Michael Bendersky, Donald Metzler, and Marc Najork. 2018. Position bias estimation for unbiased learning to rank in personal search. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 610–618.

[26] Xuanhui Wang, Cheng Li, Nadav Golbandi, Michael Bendersky, and Marc Najork. 2018. The lambdaloss framework for ranking metric optimization. In *Proceedings of the 27th ACM international conference on information and knowledge management*. 1313–1322.

[27] 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*. 1192–1199.

[28] Le Yan, Zhen Qin, Rama Kumar Pasumarthi, Xuanhui Wang, and Mike Bendersky. 2021. Diversification-Aware Learning to Rank using Distributed Representation. In *Proceedings of the Web Conference 2021 (WWW '21)*. 127–136.