LearnDCG: End-to-End Joint Optimization of Ranker and Loss in Neural Ranking

Abstract

Neural rankers are typically trained with surrogate objectives because target evaluation metrics such as Normalized Discounted Cumulative Gain (NDCG) are non-differentiable. Existing approaches rely on hand-designed surrogate losses or fixed smooth relaxations, which can be misaligned with the target metric or impose rigid inductive biases. More recent efforts to learn the loss itself (e.g., NeuralLoss) require multi-stage pretraining on synthetic data and may not match real-world relevance distributions. We introduce `LearnDCG`, a differentiable and learnable approximation of NDCG that unifies ranker and loss optimization within a single end-to-end pipeline. `LearnDCG` parameterizes the gain, discount, and temperature components of NDCG and learns them jointly with the ranker, enabling adaptation to dataset-specific relevance patterns while preserving the theoretical guarantees of differentiable ranking formulations. Experiments on MQ2007, MQ2008, WEB10K, and YLTR show that `LearnDCG` consistently outperforms classical surrogate losses, differentiable relaxations, and learnable alternatives. Beyond improved effectiveness, `LearnDCG` eliminates synthetic pretraining and reduces training complexity, achieving state-of-the-art results with improved efficiency. Gains are statistically significant across diverse architectures, from MLPs to transformer-based rankers, demonstrating the generality of the approach.

1 Introduction

From search engines to recommender systems, ranking tasks are central to the delivery of information to users and deeply integrated in customized user experiences [12, 34]. One key challenge faced by neural rankers is the non-differentiable nature of ranking performance metrics such as Normalized Discounted Cumulative Gain (NDCG) and Mean Reciprocal Rank (MRR) [4, 26]. Their discontinuity means they cannot be directly used in model optimization and backpropagation, leading to most approaches using surrogate losses [2, 20, 22] or differentiable approximations of ranking metrics [1, 6, 16, 21, 24]; however, the former are only loosely connected to the evaluation metrics that define ranking performance [10], while the latter fix the functional form of the loss *a priori* and cannot adapt to the distributional properties of different datasets [23]. Related efforts in diversified search have also explored differentiable reformulations of non-decomposable metrics for end-to-end optimization [31, 33], though these target diversity-specific objectives rather than relevance-based ranking losses.

More recently, researchers have begun to explore the possibility of learning the loss function itself. The *NeuralLoss* approach introduces a neural network surrogate that is pretrained to approximate target ranking metrics on metric-driven data [13]. By leveraging the approximation capacity of deep models [27] and enforcing properties such as permutation invariance through a Transformer-based architecture, NeuralLoss provides a continuous and differentiable objective that aligns optimization with non-differentiable evaluation criteria. This approach shows that the loss function itself can be learned, providing a systematic approach to align training objectives with the evaluation metrics that define ranking performance. Despite its novel approach, NeuralLoss has several shortcomings: **(1)** The loss model is trained entirely on synthetic data generated from a uniform distribution of relevance scores and labels, which may be considered to be an oversimplification of the complex structure of ranking problems [6]; **(2)** By relying on uniformly distributed training data, the approach fails to capture the strong imbalances characteristic of real-world ranking scenarios, where highly relevant documents are scarce compared to marginally relevant or irrelevant ones [7]; and, **(3)** The assumption of a uniform distribution also disregards domain-specific patterns in relevance score distributions that play a critical role in shaping ranking effectiveness, leading to potential misalignment between training loss and evaluation criterion [8, 10].

To address these limitations, we introduce `LearnDCG`, a learnable and parametrized approximation of NDCG that is optimized *jointly with the ranker in an end-to-end pipeline*. This unified training setup ensures that the loss function adapts directly to the same data and gradients driving the ranker, eliminating the need for pretraining or multi-stage optimization as required by methods such as NeuralLoss [13]. Our formulation decomposes the differentiable NDCG surrogate into three key components: a gain function, which specifies the relevance value assigned to each document based on its ground-truth label; a discount function, which determines the decay in positional importance as documents appear lower in the ranking; and a temperature parameter, which regulates the smoothness of the soft-sorting operation that underlies differentiability. Standard differentiable formulations fix these components *a priori*, imposing rigid inductive biases that cannot adapt across datasets. `LearnDCG` instead treats them as learnable functions and trains them alongside the ranker. By embedding loss learning directly into the model's optimization, our end-to-end approach allows the system to discover gain curves, discount patterns, and temperature schedules that best align with the structure of the task and the distributional properties of the dataset. As such, `LearnDCG` possesses two unique characteristics, namely **(a)** it is jointly optimized with the ranker in an end-to-end manner; and, **(b)** it adaptively aligns with dataset-specific distributions without requiring synthetic pretraining.

We evaluate `LearnDCG` on four widely used benchmark datasets, MQ2007, MQ2008, WEB10K, and YLTR (we also report `OHSUMED` [9] dataset results on our Github; omitted here due to space constraints), covering both medium-scale and large-scale ranking scenarios. Across these datasets, we compare against classical surrogate losses, differentiable relaxations of NDCG, and learnable surrogate losses. Our experiments demonstrate that jointly training the loss and the ranker in a single end-to-end pipeline consistently yields higher effectiveness, with statistically significant improvements on standard evaluation metrics such as NDCG and MRR. In addition to effectiveness, `LearnDCG` eliminates the need for pretraining or staged optimization, thereby reducing training complexity

while improving efficiency. Finally, we show that these improvements are stable across diverse ranking architectures, indicating the robustness and generality of our approach.

2 Proposed Approach

Static Differentiable Loss. Our methodology begins with a standard differentiable formulation of NDCG, which reformulates the non-differentiable ranking metric into a smooth surrogate objective through position function approximations. Let $X = \{x_1, \ldots, x_n\}$ denote the set of documents associated with a query, the ground-truth relevance labels $r(x_i) \in \mathbb{Z}_{\geq 0}$, and $s_i = f(x_i; \theta)$ being the score assigned by a ranking function f with parameters θ. The canonical NDCG can be defined as:

$$\text{NDCG} = \frac{\sum_{i=1}^{n} \frac{g(r(x_i))}{d(\pi(x_i))}}{\max_{\pi^*} \sum_{i=1}^{n} \frac{g(r(x_i))}{d(\pi^*(x_i))}}, \tag{1}$$

where π^* denotes the ideal ranking permutation, $g(\cdot)$ is the gain function, $d(\cdot)$ is the discount function, and $\pi(x_i)$ is the discrete rank position of document x_i. The framework is built on three key elements: (i) reformulating the measure from position-based indexing to document-based indexing, (ii) approximating the non-differentiable position and truncation functions with smooth sigmoid-based surrogates, and (iii) optimizing the resulting continuous objective via gradient descent. Here, the key is to approximate the non-differentiable position $\pi(x_i)$ by a smooth surrogate $\hat{\pi}(x_i)$ defined as:

$$\hat{\pi}(x_i) = 1 + \sum_{j \neq i} \sigma(-\alpha(s_i - s_j)), \tag{2}$$

where $\sigma(z)$ is the sigmoid function and $\alpha > 0$ is a temperature parameter controlling the sharpness of the approximation. This can now yield differentiable approximations $\hat{\pi}(x_i) \approx \pi(x_i)$ with provably bounded error [25].

While this formulation achieves an accurate surrogate of NDCG with provable approximation bounds [25], the conventional choice of fixing the gain function as $g(y) = 2^y - 1$ and the discount as $d(r) = \log_2(1 + r)$ imposes rigid inductive biases. In practice, ranking tasks differ widely in their relevance grading schemes, slate lengths, and emphasis on top versus mid-ranked positions. For example, datasets with three-level relevance and short slates may benefit from smoother gains and flatter discounts, whereas five-level datasets require sharper gain scaling to capture fine-grained relevance distinctions.

Learnable Loss. As such, we propose `LearnDCG`, a learnable and parametrized formulation that retains the same position approximation structure [25] but replaces the fixed gain, discount, and temperature components with learnable counterparts. Rather than fixing $g(y) = 2^y - 1$ and $d(r) = \log_2(1 + r)$, we introduce learnable functions that adapt to dataset-specific characteristics. In practice, we parameterize these functions with a small set of scalar variables that are optimized jointly with the ranker, retaining the computational efficiency of the fixed formulation while allowing the loss to adapt dynamically to the data.

Instead of a fixed exponential base, we define a learnable exponential base $b_g > 1$ such that:

$$g(y) = b_g^y - 1, \tag{3}$$

where b_g is optimized jointly with the ranking model. This allows the loss to control the separation between successive relevance grades and enables the model to adaptively adjust the relative importance of different relevance levels. We generalize the logarithmic decay by introducing a parametric formulation:

$$d(r) = \frac{\ln(1 + r)}{\ln(b_d)}, \quad b_d > 1, \tag{4}$$

corresponding to a generalized logarithmic decay. The smoothness parameter α that governs the sigmoid approximation of pairwise comparisons is itself learned, which allows us to maintain adaptive control between soft and sharp rank approximations. Given the approximated positions $\hat{\pi}(x_i)$, the surrogate `LearnDCG` for a query is defined as

$$\widehat{\text{DCG}} = \sum_{i=1}^{n} \frac{g(r(x_i))}{d(\hat{\pi}(x_i))}, \tag{5}$$

The corresponding $\widehat{\text{NDCG}}$ is obtained by normalizing $\widehat{\text{DCG}}$ with the ideal discounted cumulative gain computed using the same parameterized gain and discount functions under the ideal ranking permutation. The `LearnDCG` loss is then defined over a batch of B queries as

$$\mathcal{L}_{\text{LearnDCG}} = -\frac{1}{B} \sum_{q=1}^{B} \widehat{\text{NDCG}}^{(q)}. \tag{6}$$

To ensure validity of the learned parameters and maintain numerical stability, we reparameterize all learnable scalars through the softplus transformation:

$$\tilde{\phi} = \log(\exp(\phi) + 1), \quad \phi \in \{\theta_g, \theta_d, \theta_\alpha\}, \tag{7}$$

so that $b_g = 1 + \text{softplus}(\theta_g)$, $b_d = 1 + \text{softplus}(\theta_d)$, and $\alpha = \text{softplus}(\theta_\alpha)$. This guarantees positivity and domain constraints during gradient-based optimization. The additional parameters introduce negligible computational overhead compared to the fixed formulation, since the position approximation structure remains unchanged. The parameters of the ranker and the loss function variables $\{b_g, b_d, \alpha\}$ are *updated jointly via a single backpropagation pass*, enabling **fully end-to-end training** without requiring pretraining of surrogate losses on synthetic data.

Our proposed `LearnDCG` inherits the approximation guarantees of the standard differentiable formulation, since positions are approximated with the same sigmoid relaxation. Specifically, if the pairwise score margin $\delta = \min_{i \neq j} |s_i - s_j|$ is sufficiently large, then the document position error $|\hat{\pi}(x_i) - \pi(x_i)|$ is bounded via

$$|\hat{\pi}(x_i) - \pi(x_i)| < \frac{n - 1}{\exp(\alpha\delta) + 1}, \tag{8}$$

which ensures the error vanishes to zero if $\delta \to \infty$. Therefore, `LearnDCG` preserves differentiability while extending the space of gain and discount functions from fixed to learnable, allowing the surrogate to adapt its curvature and scale to the empirical relevance distribution of each dataset.

3 Experiments

Datasets. We evaluate our proposed approach on four publicly available learning-to-rank (LTR) benchmark datasets: `MQ2007`, `MQ2008`, `WEB10K`, and `YLTR` (and include `OHSUMED` on Github). Both `MQ2007` and `MQ2008` [19] originate from the Million Query Track of TREC

Table 1: Performance comparison across ranking models.

Ranking Model	Fixed	Learned
MQ2007		
MLP [11]	54.47	**58.63**
AttSets [29]	55.21	**58.18**
Attention MIL [32]	55.96	**58.41**
Context-Aware Ranker [14]	53.92	**57.71**
MQ2008		
MLP [11]	72.27	**77.42**
AttSets [29]	74.24	**77.45**
Attention MIL [32]	75.01	**77.98**
Context-Aware Ranker [14]	75.21	**78.03**
WEB10K		
MLP [11]	35.49	**43.20**
AttSets [29]	41.22	**46.71**
Attention MIL [32]	42.02	**46.27**
Context-Aware Ranker [14]	37.64	**45.53**
YLTR		
MLP [11]	70.41	**73.04**
AttSets [29]	75.31	**77.63**
Attention MIL [32]	75.16	**77.15**
Context-Aware Ranker [14]	70.92	**75.82**

All improvements are statistically significant at $p < 0.05$ using paired t-test.

Table 2: Performance across datasets on NDCG/MRR @ 5/10.

Configuration		Metrics			
Dataset	**Loss**	**NDCG @5**	**NDCG @10**	**MRR @5**	**MRR @10**
MQ2007	ApproxNDCG	50.08	53.92	48.79	50.34
	LambdaRank	49.8	52.87	51.81	54.22
	LambdaLoss	53.28	56.26	49.86	50.49
	NeuralNDCG	54.32	54.95	54.39	56.01
	ListMLE	47.65	50.71	52.71	54.29
	ListNet	54.23	57.52	59.37	60.71
	RankNet	46.12	47.36	55.46	57.01
	RMSE	53.79	56.93	59.14	60.55
	NeuralLoss	54.36	57.37	54.86	56.46
	LearnDCG	**56.46**	**59.71**	**59.66**	**62.14**
MQ2008	ApproxNDCG	69.54	75.21	68.37	69.50
	LambdaRank	68.59	76.08	68.78	71.06
	LambdaLoss	71.64	76.48	70.41	72.01
	NeuralNDCG	71.82	76.87	67.96	69.40
	ListMLE	68.95	74.38	67.47	68.63
	ListNet	69.46	74.94	68.64	69.79
	RankNet	68.07	73.37	64.65	68.83
	RMSE	69.75	75.31	68.84	69.88
	NeuralLoss	72.25	77.07	70.14	71.11
	LearnDCG	**75.13**	**79.03**	**73.35**	**74.28**
WEB10K	ApproxNDCG	21.54	27.64	48.20	53.58
	LambdaRank	34.24	37.78	61.40	62.05
	LambdaLoss	38.21	40.63	60.87	68.80
	NeuralNDCG	39.13	42.02	66.51	69.92
	ListMLE	39.04	41.22	61.95	69.75
	ListNet	40.08	42.49	65.96	68.21
	RankNet	28.35	29.80	46.07	50.69
	RMSE	32.58	37.43	58.15	65.57
	NeuralLoss	22.83	25.94	46.54	50.16
	LearnDCG	**43.32**	**45.53**	**71.45**	**74.70**
YLTR	ApproxNDCG	65.33	70.92	62.55	63.82
	LambdaRank	69.74	73.45	55.75	59.71
	LambdaLoss	70.07	74.08	58.32	65.85
	NeuralNDCG	65.36	73.14	62.74	65.66
	ListMLE	49.24	58.04	36.79	39.31
	ListNet	70.91	75.75	68.19	69.33
	RankNet	48.96	57.84	36.67	39.24
	RMSE	62.89	69.93	52.91	54.68
	NeuralLoss	67.66	73.09	62.27	63.55
	LearnDCG	**72.06**	**76.82**	**68.88**	**69.91**

All improvements are statistically significant at $p < 0.05$ using paired t-test.

2007 and TREC 2008, respectively. MQ2007 contains 1,692 queries with 69,623 labeled query-document pairs, while MQ2008 includes 784 queries with 15,211 labeled query-document pairs. Both datasets use three-grade relevance judgments. WEB10K (MSLR-WEB10K) [17] is a large-scale benchmark derived from a commercial search engine containing 10,000 queries with approximately 1.2 million query-document pairs and five-level relevance labels. YLTR (Yahoo Learning to Rank Set-1) [5] is a large-scale web search benchmark containing 29,921 queries and 709,877 query-document pairs with five-level relevance labels. For the first three datasets, we adopt the standard five-fold cross-validation partitioning. For YLTR, we apply standard train/validation/test split over Set-1.

Baselines. We compare against nine baselines spanning three categories. *Surrogate losses*: RMSE (pointwise), RankNet [3] (pairwise), and the listwise methods ListMLE [30] and ListNet [4]. *Differentiable metric approximations*: LambdaRank [28], LambdaLoss [28], ApproxNDCG [18], and NeuralNDCG [15]. *Learnable surrogate losses*: NeuralLoss [13], which approximates NDCG with a pretrained neural network in a multi-stage configuration. We additionally compare LearnDCG against its own fixed-parameter variant (i.e., non-learned gain, discount, and temperature) to isolate the contribution of joint optimization. We use the same transformer-based ranker architecture in AllRank [14] across all experiments for fairness. We report NDCG and MRR values at cutoffs 5 and 10 as our evaluation metrics.

Research Questions. To systematically evaluate the effectiveness and generality of LearnDCG, we assess our work through five research questions: (**RQ1**) *Effectiveness*: Does jointly learning the loss and the ranker in an end-to-end manner yield consistent performance improvements over the fixed-parameter formulation? **(RQ2)** *Comparison with Learnable Alternatives:* How does LearnDCG compare to existing learnable surrogate losses such as NeuralLoss, which rely on multi-stage pretraining on synthetic data? **(RQ3)** *Comparison with State of the Art*: How does LearnDCG compare against a range of state-of-the-art surrogate losses used in learn to rank methods? **(RQ4)** *Efficiency:* Does the single end-to-end training pipeline of LearnDCG reduce training complexity and computational costs compared to multi-stage approaches? **(RQ5)** *Generalizability:* Are the performance gains of LearnDCG consistent across different neural ranking architectures, ranging from simple MLPs to transformer-based rankers?

Code and Data. All our code and results are publicly available at https://anonymous.4open.science/r/LearnDCG-D44E.

4 Findings

RQ1 (Effectiveness). We first compare LearnDCG against its fixed-parameter counterpart to isolate the contribution of making gain, discount, and temperature learnable. Table 1 reports NDCG@10 for both variants across four ranker architectures and four datasets,

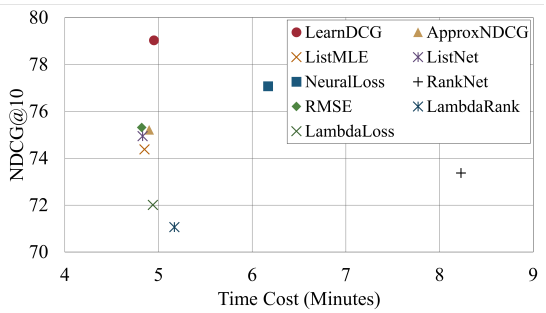

Figure 1: Total training time vs. NDCG@10 **on MQ2008. Training details include RTX A6000 GPU with 48 GB of memory, a learning rate of 0.001, and a batch size of 32.**

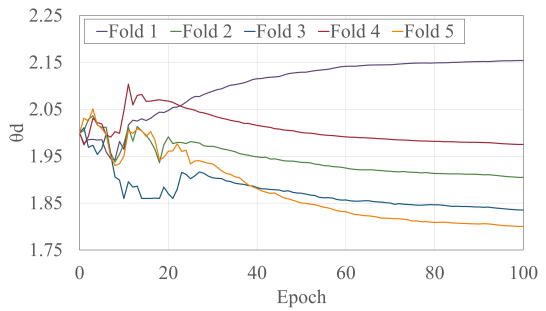

Figure 2: Evolution of the learned discount base θ_d across training epochs for all five folds of WEB10K.

together with significance tests using paired t-test. LearnDCG consistently and significantly outperforms the fixed formulation in every configuration ($p < 0.05$ in all cases, and $p \leq 0.003$ on WEB10K). The largest gains appear on WEB10K, where for example the MLP improves from 35.49 to 43.20, indicating that the fixed parameterization is particularly ill-suited for datasets with highly skewed relevance distributions. Figure 2 shows the progression of the learned discount parameter θ_d during training, which converges to different values across the five folds of WEB10K, ranging from approximately 1.80 to 2.15, further illustrating that no single fixed setting is universally optimal and that end-to-end optimization allows the loss to adapt to fold-specific data characteristics. Due to space constraints, we only show θ_d for WEB10K here. Further results for other parameters and datasets can be found in our repository.

RQ2 (Comparison with Learnable Alternatives). We compare LearnDCG against NeuralLoss [13], which parameterizes the loss through a neural surrogate pretrained on synthetic data. LearnDCG consistently outperforms NeuralLoss across all datasets and metrics. On MQ2007, LearnDCG achieves 59.66 MRR@5 versus 54.86 for NeuralLoss; on WEB10K the gap is particularly large (43.32 vs. 22.83 NDCG@5). NeuralLoss also struggles on WEB10K and YLTR, likely due to distributional mismatch between its uniform synthetic pretraining data and the skewed relevance distributions of real-world benchmarks. LearnDCG avoids this failure mode entirely by learning the loss jointly with the ranker on the target data.

RQ3 (Comparison with State of the Art). Table 2 compares LearnDCG against nine baselines spanning surrogate losses, differentiable metric approximations, and learnable alternatives. LearnDCG achieves the highest score on every metric across all four datasets. On WEB10K, LearnDCG attains 43.32 NDCG@5, outperforming the next-best method (ListNet, 40.08) and strong differentiable baselines such as NeuralNDCG (39.13) and LambdaLoss (38.21). On MQ2008, LearnDCG reaches 75.13 NDCG@5 compared to 72.25 for NeuralLoss and 71.82 for NeuralNDCG. On YLTR, LearnDCG yields 72.06 NDCG@5 versus 70.91 for ListNet. While individual baselines are competitive on specific datasets, e.g., ListNet performs well on YLTR and LambdaLoss on MQ2008, none matches LearnDCG's consistent effectiveness across all evaluation conditions.

RQ4 (Efficiency). This research question asks whether integrating the loss and the ranker into a single end-to-end pipeline reduces training complexity and computational costs compared to multi-stage approaches. Figure 1 provides a joint view of effectiveness (NDCG@10) against training cost for different loss functions. The most desirable region of the plot is the top-left corner, which represents methods that achieve higher effectiveness with lower computational cost. LearnDCG is positioned closest to this region, combining the highest NDCG@10 score with one of the lowest training costs, highlighting both its effectiveness and efficiency. In contrast, NeuralLoss, which requires multi-stage pretraining, appears further to the top-right, indicating strong but less efficient performance due to additional overhead. Classical surrogate losses such as RMSE, ListNet, and ListMLE cluster near the middle-bottom region, offering moderate efficiency but clearly lower effectiveness. RankNet is the least desirable, positioned in the bottom-right corner, reflecting both lower effectiveness and higher cost. The figure shows that LearnDCG achieves the best balance of efficiency and effectiveness by situating itself near the desirable top-left region.

RQ5 (Generalizability). Table 1 demonstrates that LearnDCG improves performance across all ranker architectures and all four datasets, with the largest gains on WEB10K (e.g., MLP improves from 35.49 to 43.20, $p < 0.001$). On MQ2008, the Context-Aware Ranker improves from 75.21 to 78.03 and Attention MIL from 75.01 to 77.98 (both $p < 0.05$). All improvements are statistically significant. These results confirm that LearnDCG generalizes well across architectures, with particularly strong benefits on larger datasets where fixed parameterizations are most limiting.

5 Concluding Remarks

This paper presented LearnDCG, a learnable surrogate loss that integrates directly with neural rankers in an end-to-end training pipeline. By treating the gain, discount, and temperature components of NDCG as learnable parameters, our approach eliminates the rigid biases of fixed surrogates and adapts dynamically to the relevance distributions of different datasets. Unlike prior learnable methods such as NeuralLoss, LearnDCG requires no synthetic pretraining or staged optimization, reducing computational overhead while maintaining theoretical guarantees of differentiability and bounded approximation error. Through experiments on four standard datasets, we demonstrated that LearnDCG consistently outperforms nine baselines including surrogate losses, differentiable metric approximations, and learnable alternatives. The improvements are robust across evaluation metrics, datasets of varying size and granularity, and neural architectures ranging from MLPs to transformer-based rankers.

References

[1] Mathieu Blondel, Olivier Teboul, Quentin Berthet, and Josip Djolonga. 2020. Fast Differentiable Sorting and Ranking. arXiv:2002.08871 [stat.ML] https://arxiv.org/abs/2002.08871

[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* (Santa Clara, CA, USA) *(ICTIR '19)*. Association for Computing Machinery, New York, NY, USA, 75–78. doi:10.1145/3341981.3344221

[3] 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* (Bonn, Germany) *(ICML '05)*. Association for Computing Machinery, New York, NY, USA, 89–96. doi:10.1145/1102351.1102363

[4] 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. doi:10.1145/1273496.1273513

[5] Olivier Chapelle and Yi Chang. 2010. Yahoo! Learning to Rank Challenge Overview. In *Proceedings of the 2010 International Conference on Yahoo! Learning to Rank Challenge - Volume 14* (Haifa, Israel) *(YLRC'10)*. JMLR.org, 1–24.

[6] Martin Engilberge, Louis Chevallier, Patrick Pérez, and Matthieu Cord. 2019. SoDeep: A Sorting Deep Net to Learn Ranking Loss Surrogates. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 10784–10793. doi:10.1109/CVPR.2019.01105

[7] Elham Ghanbari and Azadeh Shakery. 2022. A Learning to rank framework based on cross-lingual loss function for cross-lingual information retrieval. *Applied Intelligence* 52, 3 (Feb. 2022), 3156–3174. doi:10.1007/s10489-021-02592-z

[8] Aditya Grover, Eric Wang, Aaron Zweig, and Stefano Ermon. 2019. Stochastic Optimization of Sorting Networks via Continuous Relaxations. arXiv:1903.08850 [stat.ML] https://arxiv.org/abs/1903.08850

[9] William Hersh, Chris Buckley, T. J. Leone, and David Hickam. 1994. OHSUMED: an interactive retrieval evaluation and new large test collection for research. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Dublin, Ireland) *(SIGIR '94)*. Springer-Verlag, Berlin, Heidelberg, 192–201.

[10] Sanaz Keshvari, Faezeh Ensan, and Hadi Sadoghi Yazdi. 2022. ListMAP: Listwise learning to rank as maximum a posteriori estimation. *Information Processing Management* 59, 4 (2022), 102962. doi:10.1016/j.ipm.2022.102962

[11] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*. PMLR, 3744–3753.

[12] Pan Li and Alexander Tuzhilin. 2023. Dual Metric Learning for Effective and Efficient Cross-Domain Recommendations. *IEEE Transactions on Knowledge and Data Engineering* 35, 1 (2023), 321–334. doi:10.1109/TKDE.2021.3074395

[13] Chen Liu, Cailan Jiang, and Lixin Zhou. 2025. NeuralLoss: A Learnable Pretrained Surrogate Loss for Learning to Rank. *IEEE Transactions on Knowledge and Data Engineering* 37, 7 (2025), 4179–4192. doi:10.1109/TKDE.2025.3562450

[14] Przemyslaw Pobrotyn, Tomasz Bartczak, Mikolaj Synowiec, Radoslaw Bialobrzeski, and Jaroslaw Bojar. 2020. Context-Aware Learning to Rank with Self-Attention. *ArXiv* abs/2005.10084 (2020).

[15] Przemysław Pobrotyn and Radosław Białobrzeski. 2021. Neuralndcg: Direct optimisation of a ranking metric via differentiable relaxation of sorting. *arXiv preprint arXiv:2102.07831* (2021).

[16] Sebastian Prillo and Julian Martin Eisenschlos. 2020. SoftSort: A Continuous Relaxation for the argsort Operator. arXiv:2006.16038 [cs.LG] https://arxiv.org/abs/2006.16038

[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] Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. 2010. LETOR: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval* 13, 4 (2010), 346–374.

[20] Tao Qin, Xu-Dong Zhang, Ming-Feng Tsai, De-Sheng Wang, Tie-Yan Liu, and Hang Li. 2008. Query-level loss functions for information retrieval. *Inf. Process. Manage.* 44, 2 (March 2008), 838–855. doi:10.1016/j.ipm.2007.07.016

[21] Zi-Hao Qiu, Quanqi Hu, Yongjian Zhong, Lijun Zhang, and Tianbao Yang. 2023. Large-scale Stochastic Optimization of NDCG Surrogates for Deep Learning with Provable Convergence. arXiv:2202.12183 [cs.LG] https://arxiv.org/abs/2202.12183

[22] Razieh Rahimi, Ali Montazeralghaem, and James Allan. 2019. Listwise Neural Ranking Models. In *ICTIR*. 101–104. https://doi.org/10.1145/3341981.3344245

[23] Jerome Revaud, Jon Almazan, Rafael Rezende, and Cesar De Souza. 2019. Learning With Average Precision: Training Image Retrieval With a Listwise Loss. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 5106–5115. doi:10.1109/ICCV.2019.00521

[24] Michal Rolínek, Vít Musil, Anselm Paulus, Marin Vlastelica, Claudio Michaelis, and Georg Martius. 2020. Optimizing Rank-Based Metrics With Blackbox Differentiation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 7617–7627. doi:10.1109/CVPR42600.2020.00764

[25] Thibaut Thonet, Yagmur Gizem Cinar, Éric Gaussier, Minghan Li, and Jean-Michel Renders. 2021. SmoothI: Smooth Rank Indicators for Differentiable IR Metrics. *arXiv preprint arXiv:2105.00942* (2021).

[26] Thibaut Thonet, Yagmur Gizem Cinar, Eric Gaussier, Minghan Li, and Jean-Michel Renders. 2022. Listwise Learning to Rank Based on Approximate Rank Indicators. *Proceedings of the AAAI Conference on Artificial Intelligence* 36, 8 (Jun. 2022), 8494–8502. doi:10.1609/aaai.v36i8.20826

[27] Edward Wagstaff, Fabian B. Fuchs, Martin Engelcke, Michael A. Osborne, and Ingmar Posner. 2021. Universal Approximation of Functions on Sets. arXiv:2107.01959 [cs.LG] https://arxiv.org/abs/2107.01959

[28] 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* (Torino, Italy) *(CIKM '18)*. Association for Computing Machinery, New York, NY, USA, 1313–1322. doi:10.1145/3269206.3271784

[29] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, and Tie-Yan Liu. 2013. A theoretical analysis of NDCG type ranking measures. In *Conference on learning theory*. PMLR, 25–54.

[30] 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. doi:10.1145/1390156.1390306

[31] Le Yan, Zhen Qin, Rama Kumar Pasumarthi, Xuanhui Wang, and Michael Bendersky. 2021. Diversification-Aware Learning to Rank using Distributed Representation. In *Proceedings of the Web Conference 2021* (Ljubljana, Slovenia) *(WWW '21)*. Association for Computing Machinery, New York, NY, USA, 127–136. doi:10.1145/3442381.3449831

[32] Bo Yang, Sen Wang, Andrew Markham, and Niki Trigoni. 2020. Robust attentional aggregation of deep feature sets for multi-view 3D reconstruction. *International Journal of Computer Vision* 128, 1 (2020), 53–73.

[33] Hai-Tao Yu. 2022. Optimize What You Evaluate With: Search Result Diversification Based on Metric Optimization. *Proceedings of the AAAI Conference on Artificial Intelligence* 36, 9 (Jun. 2022), 10399–10407. doi:10.1609/aaai.v36i9.21282

[34] Meike Zehlike, Ke Yang, and Julia Stoyanovich. 2022. Fairness in Ranking, Part II: Learning-to-Rank and Recommender Systems. *ACM Comput. Surv.* 55, 6, Article 117 (Dec. 2022), 41 pages. doi:10.1145/3533380