Is Tranformer the Best Option in LTR Tasks?

Yuri Sapronov [1 2]

Abstract

In ranking tasks, gradient boosting has long been the preferred approach over neural networks, with increasingly complex loss functions emerging each year to directly optimize specific metrics. However, we combined one of the earliest loss functions with a lightweight transformer and achieved results that significantly outperform gradient boosting. Furthermore, our method substantially improves upon previous attempts to apply transformers to ranking problems. While transformers can achieve very high metric scores, this often requires a large number of epochs. We address this issue by enhancing the training approach for our neural network.

1. Introduction

Ranking is a fundamental problem in information retrieval, recommendation systems, and various machine learning applications. The task involves sorting a set of items in descending order of relevance or utility with respect to a given query or context. Traditional approaches to ranking rely heavily on gradient boosting methods, such as LambdaMART [1] and XGBoost [2]. These methods have gained popularity due to their interpretability, scalability, and strong performance on a wide range of datasets.

In recent years, however, neural networks have begun to challenge gradient boosting by demonstrating their potential to model complex interactions between features. Approaches like RankNet [3], ListNet [4], and DeepRank [5] have introduced new perspectives to learning-to-rank tasks by incorporating neural architectures. Despite these advancements, gradient boosting remains the dominant paradigm, largely due to its ability to directly optimize ranking-specific metrics such as Normalized Discounted Cumulative Gain (NDCG) and Mean Reciprocal Rank (MRR).

1.1. Problem Definition

The ranking problem can be formally defined as follows: Given a query $q \in \mathcal{Q}$ and a set of associated documents $D = \{d_1, d_2, \ldots, d_n\}$, the goal is to assign a relevance score $s(d_i, q)$ to each document $d_i \in D$ such that the resulting ranked list maximizes a predefined evaluation metric.

Commonly used metrics for evaluating ranking algorithms include:

1. **Normalized Discounted Cumulative Gain (NDCG):**

$$\text{NDCG@k} = \frac{1}{\text{IDCG@k}} \sum_{i=1}^{k} \frac{2^{r_i} - 1}{\log_2(i+1)},$$

   where r_i is the relevance score of the document at position i, and IDCG@k is the ideal DCG, computed by sorting documents by their true relevance and applying the same formula.

2. **Mean Reciprocal Rank (MRR):**

$$\text{MRR} = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{\text{rank}_q},$$

   where rank_q is the position of the first relevant document for query q.

3. **Precision at k (P@k):**

$$\text{P@k} = \frac{1}{k} \sum_{i=1}^{k} \mathbb{I}(r_i > 0),$$

   where $\mathbb{I}$ is the indicator function.

These metrics are designed to reflect different aspects of ranking quality. For instance, NDCG emphasizes the importance of ranking highly relevant documents at the top of the list, while MRR focuses on the position of the first relevant document.

[*]Equal contribution [1]Moscow Institute of Physics and Technology, Dolgoprudny, Russia [2]Higher School of Economics University, Moscow, Russia. Correspondence to: Yuri Sapronov <sapronov.iuf1@phystech.edu>, Alexander Beznosikov <first2.last2@www.uk>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

1.2. Challenges and Opportunities

One of the main challenges in ranking is the discrepancy between the loss functions used during training and the evaluation metrics. Most neural ranking models are trained using surrogate losses, such as cross-entropy or pairwise hinge loss, which may not directly correlate with ranking-specific metrics like NDCG. To address this gap, recent works [6, 7] have proposed differentiable approximations of ranking metrics. These approaches enable end-to-end optimization of models with respect to NDCG or other metrics, though at the cost of increased computational complexity.

Transformers [8], originally introduced for natural language processing, have shown promise in ranking tasks due to their ability to model sequential dependencies and contextual information. However, applying transformers to ranking problems presents unique challenges, such as handling variable-length input sequences and computational overhead for large datasets.

2. Experimental Setup

Our primary objective was to develop a model whose output is invariant to the order in which documents are presented. To achieve this, we removed components like positional encodings and autoregressive masks, ensuring that the model processes documents in a permutation-invariant manner. Conceptually, our model operates as an encoder.

The best-performing configuration consisted of 2 transformer blocks with 4 attention heads per block. This lightweight setup struck a balance between computational efficiency and performance.

During training, we ensured that all documents for a query fit into the model, even for queries with up to 800 documents. We used a large batch size of 128 to stabilize training; however, smaller batch sizes can be used in cases of computational constraints.

For the loss function, we adopted the approach from ListNet, where the idea is to model the permutation probability of documents based on their relevance scores. Specifically, for , we compute the probability of a document being ranked first. Unlike the traditional softmax approach, we defined the target probability of a document being ranked first as the document's label divided by the sum of all labels within the query. Surprisingly, this adjustment significantly improved results, yielding up to a 3

The table presents the best results we achieved for the specified loss-function and model pairs. Below, we also provide a comparison with one of the best gradient boosting methods and a transformer previously applied to the ranking task.

References

Arik, S. Ö. and Pfister, T. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 6679–6687, 2021.

Burges, C., Ragno, R., and Le, Q. Learning to rank with non-smooth cost functions. *Advances in Neural Information Processing Systems*, 19, 2006.

Burges, C. J. C. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81, 2010.

Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., and Li, H. Learning to rank: From pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning*, pp. 129–136, 2007.

Chen, T. and Guestrin, C. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, 2016.

Luo, T., Wang, D., Liu, R., and Pan, Y. Stochastic top-k listnet. *arXiv preprint arXiv:1511.00271*, 2015.

Lyu, L., Roy, N., Oosterhuis, H., and Anand, A. Is interpretable machine learning effective at feature selection for neural learning-to-rank? In *European Conference on Information Retrieval*, pp. 384–402. Springer, 2024.

Pobrotyn, P., Bartczak, T., Synowiec, M., Białobrzeski, R., and Bojar, J. Context-aware learning to rank with self-attention. *arXiv preprint arXiv:2005.10084*, 2020.

Qin, T., Liu, T.-Y., and Li, H. A general approximation framework for direct optimization of information retrieval measures. *Information Retrieval*, 13:375–397, 2010.

Swezey, R., Grover, A., Charron, B., and Ermon, S. Pirank: Scalable learning to rank via differentiable sorting. *Advances in Neural Information Processing Systems*, 34: 21644–21654, 2021.

Taylor, M., Guiver, J., Robertson, S., and Minka, T. Softrank: Optimizing non-smooth rank metrics. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pp. 77–86, 2008.

Ustimenko, A. and Prokhorenkova, L. Stochasticrank: Global optimization of scale-free discrete functions. In *International Conference on Machine Learning*, pp. 9669–9679. PMLR, 2020.

Vaswani, A. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Method + Architecture	Web10k			Other Dataset		
	NDCG@5	NDCG@10	NDCG@all	NDCG@5	NDCG@10	NDCG@all
ListNet + Simple NN	53.5	53.8	77.6	–	–	–
SoftNDCG + TabNet	40.9	45.1	73.2	–	–	–
ListNet + TabNet	47.0	51.01	76.08	–	–	–
SoftNDCG + Simple NN	24.24	27.21	63.49	–	–	–
ListNet + our Transformer	**79.5**	**79.7**	**91.7**	**–**	**–**	**–**

Table 1. Performance comparison of methods on two datasets. The second dataset results are currently unavailable.

Method + Architecture	Web10k			Other Dataset		
	NDCG@5	NDCG@10	NDCG@all	NDCG@5	NDCG@10	NDCG@all
StochasticRank + Gradient Boosting	48.3	49.2	78.9	–	–	–
Ordinal Loss + Transformer	53.0	54.9	75.2	–	–	–
ListNet + our Transformer	**79.5**	**79.7**	**91.7**	**–**	**–**	**–**

Table 2. Performance comparison of methods on two datasets. The second dataset results are currently unavailable.