Exploring the Impact of Objective Function Selection on Robustness for Ranking Tasks with Limited Training Data

RIXT HELLINGA, JOOST JANSEN, and YUAN TIAN

Learning to rank (LTR) is an important problem in information retrieval, which aims to learn a ranking function from a set of training examples. However, in many real-world scenarios, LTR models often suffer from limited training data, which can lead to overfitting and poor generalisation performance. In this paper, we investigate the robustness of different objective functions in LTR models with little training data.

We first review four popular objective functions for LTR models: Pointwise, RankNet, ApproxNDCG, and NeuralNDCG. We then conduct experiments on two benchmark datasets with limited training data to evaluate the performance of these objective functions. Our experiments demonstrate that ApproxNDCG outperforms other functions in terms of robustness when considering the more trivial dataset MQ2008, achieving higher performance on the test sets with limited training data. With the more complex dataset MSLR-WEB10K, NeuralNDCG outperforms the other objective functions.

Our results suggest that when training LTR models with limited data, the choice of objective function matters in the case of a limited number of training examples. This study contributes to the understanding of the performance of different objective functions in sub-optimal settings and provides a basis for further research into the applicability of objective functions. Our source code is publicly available. [1]

CCS Concepts: • **Information systems** → **Learning to rank**.

Additional Key Words and Phrases: learning-to-rank, limited training data

ACM Reference Format:
Rixt Hellinga, Joost Jansen, and Yuan Tian. 2023. Exploring the Impact of Objective Function Selection on Robustness for Ranking Tasks with Limited Training Data . In *IN4325-Q3-23: Information Retrieval, TU Delft*. ACM, New York, NY, USA, 9 pages.

1 INTRODUCTION

Ranking plays a critical role in various applications, including search engines, recommendation systems, and online advertising. Learning-to-rank (LTR) has emerged as a popular approach to tackle ranking problems, which involves predicting the relevance score of each input instance and ranking the final items based on the predicted scores. LTR has gained significant attention in the information retrieval and machine learning communities due to its effectiveness. While gradient-boosted decision trees (GBDT) [13] are the state-of-the-art method for LTR, deep neural models have also gained popularity due to their ability to capture complex patterns in the data.

Moreover, various objective functions that score the items individually have been proposed to enhance the ranking performance, such as pointwise, pairwise and listwise functions. The resulting list is then sorted in descending order of the scores. Possible interactions between items in the same list are considered in the training phase at the loss level. However, during inference, items are scored individually, and possible interactions between them are not considered. Therefore the authors of [18] introduced a new neural model that determines the relevance of a given item in the

[1]https://github.com/Joost-Jansen/IR_project

context of all other items present in the list, both in training and in inference using a self-attention mechanism. Various objective functions are considered by the authors to claim consistent improvements in overall performance with their novel model.

However, most neural models require large training data to achieve high performance, which may not be feasible in some cases, such as in confidential databases or personal search. Training on small datasets risks a poor generalisation of the model [25]. Therefore, this paper compares the robustness of different objective functions, including Pointwise (implemented as the Root Mean Squared Error), RankNet [4], ApproxNDCG [3], and NeuralNDCG [19], using the same neural model as the authors of [18] with limited training data. Specifically, we've found that ApproxNDCG is more robust to small training sizes with more trivial datasets than NeuralNDCG. To demonstrate this effect we used different partition sizes of the trivial dataset MQ2008 [16] and the more complex and larger dataset MSLR-WEB10k [21].

The rest of the paper is organised as follows. In Section 2 we review related work. In Section 3 we formulate the problem solved in this work. Experimental results and their discussion are presented in Section 4. Finally, a summary of our work is given in Section 5.

2 RELATED WORK

The field of LTR models that consider self-attention as in [18] show little to no work on experiments with small training sets, and most work is conducted on big web data [15, 18, 24]. In the broader field of LTR models, much research has utilised user feedback such as clickthrough data, but typically these models are also still trained on large datasets [6, 11]. While successful in these contexts, these models have yet to be tested in situations with limited training data, such as personal search.

A comparable situation to little training data is one where sparse data occurs. Research on data sparsity has been more prevalent, such as [2, 26]. However, these focus more on selection bias in personal search rather than the performance of different objective functions on sparse models. Additionally, while data sparsity and limited training examples could lead to similar bias, the effect the chosen objective functions in a situation with limited training data is not guaranteed to be similar.

Although some research exists on the performance of a model trained on few examples, the focus in such papers often lies in analysing other aspects than the objective function. For example, [10], characterises limited training data differently rather than optimising the objective function. They do however note that designing a new objective function could be a potential future direction for research. [17] found a robust LTR method for ranking code samples that is robust to smaller training sets. But their study focuses mainly on testing their ranking schema generator rather than the impact of objective function choice on model performance.

A more comparable research is that of [12], where LTR on images with limited training and compared the performance of different loss functions. The results demonstrate that the choice of objective function affects performance on several datasets. Although [12] has a similar objective to ours, the context of the used model (eg. lack of self-attention, different data-types, and no context awareness) is vastly different and thus the conclusions cannot be extended to our own.

Related work on the attributes and comparisons of different loss functions is more common. [7] compares loss functions in relation to ranking measures. Results show that using the NDCG@k metric, there is a difference in performance for the loss functions. The used loss functions are not directly related to a certain metric, which is the case for some of our chosen loss functions. According to [28, 29] loss functions directly related to performance measures (eg. ApproxNDCG) perform well when those performance measures are used. It is therefore likely that there will be a noticeable difference in performance in our chosen loss functions.

Both ApproxNDCG and NeuralNDCG are directly related to a metric (NDCG). ApproxNDCG shows to outperform other loss functions including RankNet, when measuring with NDCG [3, 14]. ApproxNDCG does however have the disadvantage that it gets stuck in local optima easily [22, 27], an issue that might be enhanced with fewer training examples. In other cases NeuralNDCG and ApproxNDCG perform similarly to one another [8, 20]. However in all mentioned cases the size of the training set is significantly large. An issue with RMSE is that it can be sensitive to outliers [1], the effect of which are possibly amplified by a small training set. Additionally, RankNet might be negatively affected by a small training size too; According to [9, 23] its performance is sensitive to the number of documents. However, another study has shown RankNet to outperform other loss functions in an intranet setting, of which smaller datasets are usual [5].

3 METHOD

In this study, we aim to investigate the impact of different learning-to-rank objective functions on the performance of neural models for ranking tasks with limited training set sizes. To achieve this goal, we evaluate and compare the performance of four different objective functions on subsets of two datasets: MQ2008 and MSLR-WEB10K. To begin, we divide each dataset into subsets. We then train each model on each subset of the dataset using a fixed set of hyperparameters following [18] and evaluate its performance on the validation set. On the epochs run we however differ from [18] as a result of constraints on time and computational power. On the WEB10K and MQ2008 datasets, we apply 10 and 4 epochs respectively. The best-performing model on the validation set is then selected and its performance is evaluated on the test set. We repeat this process for each subset of the dataset and record the results. By comparing the results of each objective function, we can identify which function is more robust to limited training set sizes and may be better suited for learning to rank tasks in scenarios with limited training data.

3.1 Dataset

We use the MQ2008 and MSLR-WEB10K datasets for our analysis. MQ2008 [16] and MSLR-WEB10K [21] are widely used benchmark datasets for LTR research. The datasets consist of feature vectors extracted from query-document or query-URL pairs, along with relevance judgment labels, where queries and relevant documents/URLs are represented by IDs. The details of these two datasets are presented in Table 1.

Table 1. Details of the MQ2008 and MSLR-WEB10K datasets

Dataset	Size	Queries	Relevant Docs/URLs	Relevance Label	Feature Dimension
MQ2008	8.92 MB	784	5-121	0-2	46
MSLR-WEB10K	1.29 GB	10,000	1-908	0-4	136

To investigate the impact of limited training set sizes, we sample subsets of the training and validation sets in each dataset in proportions of 20%, 40%, 60%, 80% and 100%, while leaving the test data untouched. We compare the performance of four LTR methods, namely pointwise, RankNet, ApproxNDCG, and NeuralNDCG, on these subsets. We use two different sampling methods, shuffling and not shuffling, to explore the effect of the order of data instances on performance. Shuffling ensures that the results are not biased by any patterns in the data that may exist due to the order in which the data was collected or labelled. However, shuffling might impair the performances in terms of slate length for each query. Table 2 shows the NDCG@5 of the four LTR methods under the two different sampling methods on the 60% MQ2008 dataset. The results demonstrate that not shuffling achieves better performance.

Table 2. NDCG@5 of different subset selection method on MQ2008

Selection Method	Pointwise	RankNet	ApproxNDCG	NeuralNDCG
Shuffling	0.5248	0.6862	0.7396	0.7404
Not shuffling	0.5474	0.7581	0.7494	0.7419

3.2 Learing-to-Rank algorithms

We adopt Root Mean Squared Error (RMSE) as the pointwise LTR method. For MQ2008 and MSLR-WEB10K, we choose 3 and 5 as their normalisation factors, respectively, according to their relevance label ranges. However, as RMSE may not capture the quality of the ranking produced by the algorithm, we employ the RankNet algorithm as a pairwise approach, which learns to predict the relative order of a pair of documents instead of an absolute score for each document.

In addition to RankNet, we use two listwise approaches to evaluate the performance of our ranking models: ApproxNDCG and NeuralNDCG. ApproxNDCG is a popular approximation of the NDCG measure. It approximates the true NDCG value by using only a subset of the top-ranked documents. In our experiments, we vary the subset size to investigate its effect on performance. On the other hand, NeuralNDCG is a neural network-based variant of NDCG that directly optimises for the metric using a differentiable approximation of the ranking function.

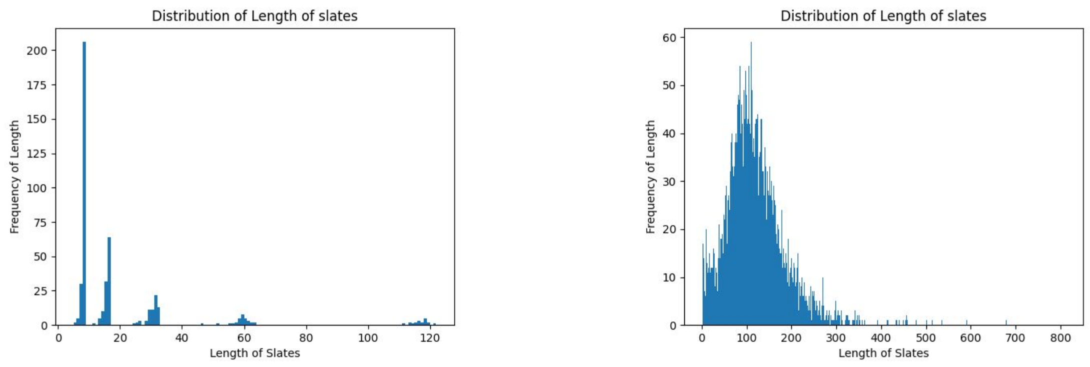
3.3 Metrics

In a query-document matching task, evaluating the effectiveness of a model in retrieving the most relevant documents for a given query is crucial. We use two metrics to compare the results of each model: NDCG@k and MRR@k, where k is the cutoff value. NDCG takes into account the graded relevance of the documents and discounts the relevance score based on its position in the ranked list, while MRR measures the effectiveness of the top-ranked document. Using both metrics provides a more comprehensive evaluation of the ranking algorithms.

We use NDCG as the primary evaluation metric for comparison, as it considers both the relevance score and ranking positions. However, it should be noted that ApproxNDCG and NeuralNDCG may have an advantage over NDCG, as they are specifically designed to optimise this metric. Nevertheless, comparing their performance on NDCG with pointwise and RankNet is useful in providing insights into how well they are performing relative to each other. We use MRR as well to get a more complete picture of how well each algorithm is performing.

To determine the cutoff values for the two metrics, we take into account the slate length distributions of the datasets and the performance changes. Different cutoff values can lead to different interpretations of the algorithm's performance. For instance, if a dataset has mostly short slates, but with some long ones, using a fixed cutoff value may bias our evaluation results by giving more weight to the longer slates. Conversely, using a smaller cutoff value may bias the results by ignoring a significant portion of the longer slates.

When using a subset of the dataset, the distribution of slate length may change, which can impact the effectiveness of different cutoff values. Therefore, we compute the distribution of slate lengths for the subset of data being used and adjust the cutoff values accordingly. Figure 1 shows an example of the slate length distribution for the full MQ2008 dataset.



(a) MQ2008

(b) MSLR-WEB10K

Fig. 1. slate length distribution

4 EXPERIMENTS

4.1 MQ2008 Dataset

The results of the experiments can be seen in Table 3, with the best result per experiment shown in bold. As expected from previous research a Pointwise objective function performs worst across all metrics compared to the other objective functions. This could be attributed to the fact that Pointwise objective functions do not take dependencies into account, a feature our model was specifically designed for.

In experiments with a lower training set ratio (20%) we can see that the NeuralNDCG method is outperformed only barely by the ApproxNDCG. In the experiments with a larger training set, NeuralNDCG shows better results, thus NeuralNDCG seems to benefit from larger training sets and ApproxNDCG seems to be more resistant to smaller training sets.

The performance of NeuralNDCG is likely to be affected by the number of epochs our model runs. Considering the time restriction for this research the number of epochs has been kept low, however, an initial experiment with an increased number of epochs showed an interesting result; Experiments with a low training set ratio (20% and 40%) show a decrease in performance for the NeuralNDCG objective function when the number of epochs is decreased while repeating the experiments with a higher training set ratio (60%, 80%, and 100%) shows the model to actually improve performance for this objective function. This impact of the number of epochs could possibly be explained by that experiments with small training sets and a large number of epochs are more prone to overfitting and the model loses its ability to generalise properly.

Noticeably, considering all four objective functions the MRR@k metric seems to be more sensitive to the size of the training set. This could be explained by that an MRR@k metric is easily influenced by outliers. The NDCG@k metric fluctuates less throughout the training ratios.

4.2 MSLR_WEB10K Dataset

The results of the experiments on the MSLR_WEB10K dataset show that NeuralNDCG, RankNet, and Pointwise all benefit in with an increase training set size. ApproxNDCG oddly enough seems to deteriorate when it comes to larger training sets, same patterns for MQ2008. One possible explanation for this is that the model is not able to generalise well

Table 3. Comparison of LTR Algorithms on MQ2008 Dataset

Ratio	Algorithm	ndcg@5	ndcg@10	ndcg@20	ndcg@50	mrr@1	mrr@5	mrr@10
20%	ApproxNDCG	**0.587**	**0.673**	**0.699**	**0.712**	**0.423**	**0.575**	**0.592**
	NeuralNDCG	0.523	0.601	0.647	0.665	0.378	0.522	0.539
	RankNet	0.486	0.564	0.610	0.631	0.314	0.469	0.489
	Pointwise	0.311	0.423	0.489	0.521	0.179	0.302	0.337
40%	ApproxNDCG	**0.589**	**0.660**	**0.690**	0.703	**0.423**	**0.595**	**0.606**
	NeuralNDCG	0.588	**0.660**	**0.690**	**0.704**	0.417	0.594	0.603
	RankNet	0.579	0.653	0.682	0.697	0.417	0.585	0.597
	Pointwise	0.316	0.434	0.504	0.531	0.212	0.318	0.357
60%	ApproxNDCG	0.563	0.633	0.664	0.679	0.365	0.544	0.555
	NeuralNDCG	**0.596**	**0.664**	**0.688**	**0.703**	**0.378**	**0.565**	**0.576**
	RankNet	0.549	0.630	0.661	0.676	0.372	0.529	0.546
	Pointwise	0.347	0.451	0.517	0.544	0.218	0.342	0.375
80%	ApproxNDCG	0.633	0.701	0.722	0.734	0.468	0.624	0.641
	NeuralNDCG	0.634	0.708	0.728	0.739	0.455	0.620	0.638
	RankNet	**0.645**	**0.710**	**0.731**	**0.740**	**0.474**	**0.632**	**0.642**
	Pointwise	0.363	0.466	0.531	0.558	0.269	0.360	0.394
100%	ApproxNDCG	0.618	0.687	0.714	0.727	0.455	0.614	0.629
	NeuralNDCG	**0.630**	**0.696**	**0.721**	**0.733**	**0.462**	**0.621**	**0.632**
	RankNet	0.597	0.673	0.703	0.718	0.455	0.604	0.620
	Pointwise	0.340	0.452	0.522	0.545	0.244	0.350	0.383

to new data, its performance may have suffered, resulting in a drop in performance as the proportion of the training set increased. But when exposed to more data like 100%, it may have allowed the model's performance to recover and eventually increase to its original values. The overall performance on this dataset is worse than for the MQ2008 dataset. This could be explained by the ratio of epochs to data set size. [18] state that a 100-epoch method was used for a data set this large, meaning that using just 10 epochs might have limited the ability of the model to learn. Also, as the MSLR_WEB10K Dataset is more complex with 136 features, it might need more complex model to train as well. The MRR@k metrics seems to increase performance more than the NDCG@k metrics do, insinuating that, just as in the MQ2008 dataset, the MRR@k metric is more sensitive to training set size than the NDCG@k metric.

5 CONCLUSIONS

In this research, we addressed to issue of training a Learning-To-Rank model in the case of having only limited training data available. We took an existing context-aware model that considered self-attention and trained it on 2 datasets with 4 different objective functions: ApproxNDCG, NeuralNDCG, RankNet, and Pointwise. We illustrate that, in a scenario with a trivial dataset such as MQ2008, ApproxNDCG is relatively robust to small training sets, while NeuralNDCG appears to benefit from larger training sets and would therefore be considered less robust. In cases with larger datasets,ApproxNDCG shows poor results and NeuralNDCG is to be favoured.

Our experiments also insinuate that the model performance is not only affected by the choice of the objective function but also by the choice of metric; MRR@k metrics increase more than NDCG@k metrics do when training on larger data sets. A limiting factor on the results of this research is the number of epochs. Considering that the used model was

Table 4. Comparison of LTR Algorithms on MSLR_WEB10K Dataset

Ratio	Algorithm	ndcg@10	ndcg@50	ndcg@100	mrr@10	mrr@50	mrr@100
20%	ApproxNDCG	0.281	0.441	0.551	0.150	0.169	0.172
	NeuralNDCG	**0.322**	**0.496**	**0.591**	**0.174**	**0.195**	**0.197**
	RankNet	0.289	0.477	0.576	0.128	0.150	0.152
	Pointwise	0.265	0.444	0.553	0.126	0.147	0.150
40%	ApproxNDCG	0.256	0.424	0.537	0.113	0.158	0.161
	NeuralNDCG	**0.345**	**0.506**	**0.600**	**0.200**	**0.220**	**0.221**
	RankNet	0.291	0.487	0.585	0.147	0.169	0.171
	Pointwise	0.268	0.446	0.556	0.144	0.165	0.168
60%	ApproxNDCG	0.266	0.430	0.542	0.138	0.158	0.161
	NeuralNDCG	**0.353**	**0.511**	**0.604**	**0.208**	**0.228**	**0.230**
	RankNet	0.291	0.483	0.579	0.133	0.155	0.157
	Pointwise	0.269	0.445	0.553	0.141	0.162	0.164
80%	ApproxNDCG	0.262	0.434	0.541	0.135	0.158	0.160
	NeuralNDCG	**0.359**	**0.515**	**0.608**	**0.225**	**0.243**	**0.245**
	RankNet	0.306	0.492	0.588	0.147	0.169	0.171
	Pointwise	0.278	0.450	0.558	0.142	0.163	0.165
100%	ApproxNDCG	0.278	0.446	0.553	0.142	0.163	0.165
	NeuralNDCG	**0.364**	**0.515**	**0.608**	**0.225**	**0.244**	**0.246**
	RankNet	0.298	0.487	0.583	0.137	0.159	0.160
	Pointwise	0.280	0.451	0.560	0.147	0.167	0.169

originally trained 100 epochs, our smaller number of epochs, which was a consequence of the time allowance, could considerably have affected the trainability of the model.

6 SELF-ASSESSMENT OF CONTRIBUTION

6.1 Self-Assessment of Joost

During the project I made significant contributions by focusing on the data preparation, properly implementing the configuration files to run the different objective functions and re-implementing our code to work in Google Colab to reduce time constraints. I also ran the final experiments successfully. In the report, I wrote the introduction, which sets the tone for the rest of the report. Overall, I enjoyed working with the team and discussing our objectives coherently. The project provided me with insights into further information retrieval methods, and I am grateful for the opportunity to work on it.

6.2 Self-Assessment of Rixt

My main contributions included researching and writing the related work. The related work in this paper remains somewhat without direction and could use some more of a structure that highlights knowledge gaps and guides the reader towards the research question. I was also responsible for determining the hyperparameters of the model, and matching them to hyperparameters used in [18]. I also contributed largely to the analysing and writing of the results discussion in section 4, as well as writing the conclusion and abstract of the paper. Even though there are some imperfections in this research, I found the experience of working hands-on with Information Retrieval techniques to be very interesting.

6.3 Self-Assessment of Yuan

My contributions to the project included running the codes and analyzing the datasets and algorithms. Through this process, I identified that certain parameters in the configuration files needed to be changed to improve the results. I adjusted the parameters accordingly and collaborated with my teammates to ensure that everyone was on the same page. In the report, I focused on documenting the methodology and experimental aspects of the project, highlighting the changes made to the parameters and their impact on the outcomes. Despite the limitations of the study, this experience gave me valuable insights into the research process and provided me with more ideas for future projects. Overall, I am grateful for the opportunity to contribute to this project and for the knowledge gained from it.

BIBLIOGRAPHY

[1] Ali R. Abdellah, Abdullah Alshahrani, Ammar Muthanna, and Andrey Koucheryavy. Performance estimation in v2x networks using deep learning-based m-estimator loss functions in the presence of outliers. *Symmetry*, 13(11), 2021.

[2] Michael Bendersky, Xuanhui Wang, Marc Najork, and Donald Metzler. Learning with sparse and biased feedback for personal search. pages 5219–5223, 7 2018.

[3] Sebastian Bruch, Masrour Zoghi, Mike Bendersky, and Marc Najork. 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 (SIGIR '19)*, pages 1241–1244, 2019.

[4] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, page 89–96, New York, NY, USA, 2005. Association for Computing Machinery.

[5] Christopher Burges, Robert Ragno, and Quoc Le. Learning to rank with nonsmooth cost functions. 19, 2006.

[6] Chapelle, Olivier, and Ya Zhang. A dynamic bayesian network click model for web search ranking. 04 2009.

[7] Wei Chen, Tie-Yan Liu, Yanyan Lan, Zhiming Ma, and Hang Li. Ranking measures and loss functions in learning to rank. pages 315–323, 01 2009.

[8] Tanya Chowdhury, Razieh Rahimi, and James Allan. Rank-lime: Local model-agnostic feature attribution for learning to rank, 2022.

[9] Wenkui Ding, Xiubo Geng, and Xu-Dong Zhang. Learning to rank from noisy data. *ACM Trans. Intell. Syst. Technol.*, 7(1), oct 2015.

[10] Kevin Duh and Katrin Kirchhoff. Learning to rank with partially-labeled data. page 251–258, 2008.

[11] Georges E. Dupret and Benjamin Piwowarski. A user browsing model to predict search engine click data from past observations. page 331–338, 2008.

[12] Songhe Feng, Zheyun Feng, and Rong Jin. Learning to rank image tags with limited training examples. *IEEE Transactions on Image Processing*, 24(4):1223–1234, 2015.

[13] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[14] Lakshya Kumar and Sagnik Sarkar. Listbert: Learning to rank e-commerce products with listwise bert, 2022.

[15] Jinzhong Li, Huan Zeng, Lei Peng, Jingwen Zhu, and Zhihong Liu. Learning to rank method combining multi-head self-attention with conditional generative adversarial nets. *Array*, 15:100205, 2022.

[16] Tie-Yan Liu et al. Letor: Learning to rank for information retrieval. https://www.microsoft.com/en-us/research/project/letor-learning-rank-information-retrieval/, 2008.

[17] Haoran Niu, Iman Keivanloo, and Ying Zou. Learning to rank code examples for code search engines. *Empirical Software Engineering*, 22:259–291, 2017.

[18] Przemyslaw Pobrotyn, Tomasz Bartczak, Mikolaj Synowiec, Radoslaw Bialobrzeski, and Jaroslaw Bojar. Context-aware learning to rank with self-attention. *ArXiv*, abs/2005.10084, 2020.

[19] Przemyslaw Pobrotyn and Radoslaw Bialobrzeski. Neuralndcg: Direct optimisation of a ranking metric via differentiable relaxation of sorting. *CoRR*, abs/2102.07831, 2021.

[20] Przemysław Pobrotyn and Radosław Białobrzeski. Neuralndcg: Direct optimisation of a ranking metric via differentiable relaxation of sorting, 2021.

[21] Tao Qin and Tie-Yan Liu. MSLR-WEB10K: Microsoft learning-to-rank datasets. https://www.microsoft.com/en-us/research/project/mslr/, 2010.

[22] Tao Qin, Tie-Yan Liu, and Hang Li. A general approximation framework for direct optimization of information retrieval measures. *Inf. Retr.*, 13:375–397, 08 2010.

[23] Tao Qin, Xu-Dong Zhang, Ming-Feng Tsai, De-Sheng Wang, Tie-Yan Liu, and Hang Li. Query-level loss functions for information retrieval. *Information Processing Management*, 44(2):838–855, 2008. Evaluating Exploratory Search Systems Digital Libraries in the Context of Users' Broader Activities.

[24] Shuo Sun and Kevin Duh. Modeling document interactions for learning to rank with regularized self-attention, 2020.

[25] Muhammad Uzair and Noreen Jamil. Effects of hidden layers on the efficiency of neural networks. pages 1–6, 2020.
[26] Xuanhui Wang, Michael Bendersky, Donald Metzler, and Marc Najork. Learning to rank with selection bias in personal search. page 115–124, 2016.
[27] Xuanhui Wang, Cheng Li, Nadav Golbandi, Michael Bendersky, and Marc Najork. The lambdaloss framework for ranking metric optimization. page 1313–1322, 2018.
[28] Hai-Tao Yu. Optimize what you evaluate with: A simple yet effective framework for direct optimization of ir metrics. 08 2020.
[29] Ke Zhou, Gui-Rong Xue, Hongyuan Zha, and Yong Yu. Learning to rank with ties. page 275–282, 2008.

Received 08 March 2023