The University of Maryland at the TREC 2020 Fair Ranking Track

Mahmoud F. Sayed and Douglas W. Oard
University of Maryland, College Park
{mfayoub,oard}@umd.edu

Abstract

In this paper, we describe our submission to the second Fair Ranking Track at TREC 2020. We leverage the flexibility of listwise Learning to Rank (LtR) techniques which directly optimize towards a custom evaluation measure. To do this, we developed an objective function that balances between relevance and fairness.

1 Introduction

In this second year of the track, the Fair Ranking Track[1] had two tasks: 1) re-ranking, and 2) retrieval. We submitted one run for the re-ranking task. The training data consists of 200 queries and 8,879 academic papers from the Semantic Scholar open research corpus [1]. Each query is associated with a pool of papers that were judged for relevance. The evaluation data consists of another 200 queries where each query is associated with an evaluation set of documents, without relevance judgments. For the re-ranking task the task is to re-rank that evaluation set in a way that minimizes the squared error of the difference between the group expected exposure of the system and desired group expected exposure specified by the organizers [5].

Our approach relies on using listwise learning to rank algorithms because they can directly optimize an objective function. We developed an objective function that is a weighted average of relevance and fairness measures. Then a listwise LtR model is used to optimize that evaluation function, thus jointly optimizing both relevance and fairness.

2 Processing Pipeline

We created one submission based on a listwise Learning to Rank (LtR) model with a custom objective function. Our pipeline starts by indexing the training data using an Elasticsearch instance with an LtR plugin. Then we generate the features for documents and for query-documents pairs that are shown in Table 1.

Our evaluation measure is computed as the weighted average between a relevance measure and a fairness measure. The relevance measure is the same as in TREC 2019 [2] which is a generalized version of ERR with a patience parameter. It is defined as follows.

$$R = \sum_{i=1}^{n} [p(s|d_i)\gamma^{i-1} \prod_{j=1}^{i-1} p(s|d_j)] \tag{1}$$

where $p(s|d_{\mathrm{i}})$ is the stopping probability at a document at rank i, γ is the patience parameter, and n is the number of documents. For the fairness measure, we compute entropy from probability of seeing an author with class L. It can be defined as follows.

[1]https://fair-trec.github.io/

ID	Feature	Category
1	BM25 score between title and query	Q-D
2	BM25 score between abstract and query	Q-D
3	BM25 score between venue and query	Q-D
4	BM25 score between sources and query	Q-D
5	BM25 score between all authors and query	Q-D
6	BM25 score between fields of study and query	Q-D
7	Min h-index of paper's authors	D
8	Avg h-index of paper's authors	D
9	Max h-index of paper's authors	D
10	Min number of papers of paper's authors	D
11	Avg number of papers of paper's authors	D
12	Max number of papers of paper's authors	D
13	Min i10-index of paper's authors	D
14	Avg i10-index of paper's authors	D
15	Max i10-index of paper's authors	D
16	Min number of citations of paper's authors	D
17	Avg number of citations of paper's authors	D
18	Max number of citations of paper's authors	D
19	Number of in citations to the paper	D
20	Number of out citations from the paper	D
21	Number of authors with class L	D
22	Number of authors with class H	D

Table 1: Learning to Rank features. Q=Query, D=Document.

$$F = -p_{n/2} \log p_{n/2} - (1 - p_{n/2}) \log(1 - p_{n/2}) \tag{2}$$

where p_k is the percentage of authors with class L that appear by position k. So the fairness is maximized if the probability is equal to 1/2. Our final objective function to be maximized, Z, is then a weighted average of the previous two measures, defined as follows:

$$Z = \beta R + (1 - \beta) F \tag{3}$$

We used the training data to evaluate three ranking algorithms: 1) AdaRank [8], 2) Coordinate Ascent [7], and 3) ListNet [4]. For our official submission, we selected the Coordinate Ascent model because it had the best 5-fold cross-validation results on the training data, as shown in Table 2.

LtR Algorithm	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Avg.
AdaRank	0.536	0.556	0.560	**0.596**	0.517	0.553
Coordinate Ascent	**0.558**	**0.572**	**0.570**	0.577	**0.527**	**0.561**
ListNet	0.400	0.424	0.517	0.519	0.371	0.446

Table 2: Objective function Z on the training data (higher is better).

3 Results

Figure 1 shows the difference between the score per query (difference in expected exposure) of our system and the median over all submitted runs to the track, with topics sorted in increasing order of that difference. As can be seen, our system achieves below-median expected exposure on 79 of the 200 queries.

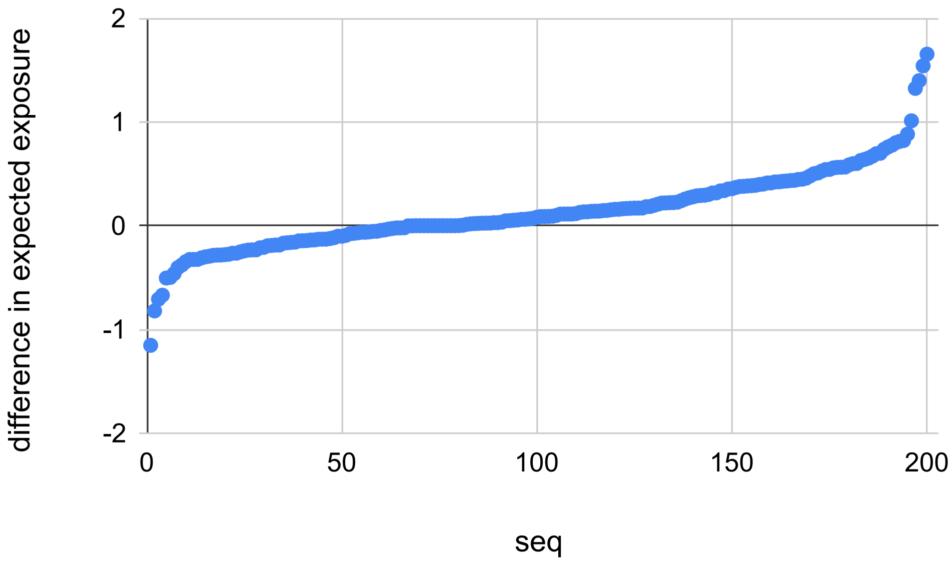

Figure 1: Sorted per-topic difference between our system's expected exposure score and median over all submitted runs (lower is better).

4 Conclusion

In this paper, we describe our submission to the second Fair Ranking Track at TREC 2020. That was our first hands-on experience with the problem of balancing relevance and fairness. There are two apparent limitations in our work. First, we based our objective function solely on the percentage of authors with class L, which is not the same as the group definition chosen by the track organizers (since the organizer's group definition is not known *a priori*). To overcome these limitations, we suggest producing multiple ranked lists for a given query based on different group definitions (e.g., number of citations, gender, or country of origin). Each ranked list could then be optimized to balance between relevance and fairness according to a specific group definition. Some form of system combination (e.g., CombMNZ [6]) might then be used in an effort to produce a ranked list for each query that is robust to the user's preferred definition of fairness.

Second, we submitted a static ranking per query without randomizing those results. No such ranked list can achieve the target expected exposure because two equally relevant documents will not receive the same exposure. One way to address this would be to use a stochastic version of learning to rank [3], and that is something we are interested in trying.

Acknowledgments

This work was supported in part by NSF grant IIS-1618695.

References

[1] Waleed Ammar, Dirk Groeneveld, Chandra Bhagavatula, Iz Beltagy, Miles Crawford, Doug Downey, Jason Dunkelberger, Ahmed Elgohary, Sergey Feldman, Vu Ha, et al. Construction of the literature graph in semantic scholar. *arXiv preprint arXiv:1805.02262*, 2018.

[2] Asia J Biega, Fernando Diaz, Michael D Ekstrand, and Sebastian Kohlmeier. Overview of the TREC 2019 fair ranking track. *arXiv preprint arXiv:2003.11650*, 2020.

[3] Sebastian Bruch, Shuguang Han, Michael Bendersky, and Marc Najork. A stochastic treatment of learning to rank scoring functions. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 61–69, 2020.

[4] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning*, pages 129–136, 2007.

[5] Fernando Diaz, Bhaskar Mitra, Michael D Ekstrand, Asia J Biega, and Ben Carterette. Evaluating stochastic rankings with expected exposure. *arXiv preprint arXiv:2004.13157*, 2020.

[6] Edward A Fox and Joseph A Shaw. Combination of multiple searches. *NIST special publication SP*, 243, 1994.

[7] Donald Metzler and W. Bruce Croft. Linear feature-based models for information retrieval. *Information Retrieval*, 10(3):257–274, 2007.

[8] Jun Xu and Hang Li. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 391–398, 2007.