ICT at TREC 2019: Fair Ranking Track

Meng Wang[1,2], Haopeng Zhang[1,2], Fuhuai Liang[1,2], Bin Feng[1,2], Di Zhao[1,2]
[1]CAS Key Laboratory of Network Data Science and Technology,
Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[2]University of Chinese Academy of Sciences, Beijing, China
{wangmeng18g, zhanghaopeng18g,liangfuhuai18g,fengbin18s,zhaodi17s}@ict.ac.cn

Abstract

In this paper, we will introduce our work in the 2019 TREC fair ranking task. In temporal academic search, more and more people choose to pay attention to the fairness constraints of ranking. The purpose of this task is to provide fairness exposure to different groups of authors, where the definition of group is diverse, such as based on demographics, height, topics, etc. For this reason, the model we design should consider the fairness of the ranking results while ensuring the relevance of the ranking results. We will show how to use our model to achieve the relevance of query results, then adjust the overall ranking results according to the fairness of documents, and finally achieve the relevance and fairness of document ordering. This paper will introduce the framework and methods of the fair ranking system, as well as the experimental results.

1 Introduction

In the Internet age, people turn from the traditional library search to Internet information search. Rankings have become one of the dominant forms with which online systems present results to the user[1]. Traditional sorting is to rank items in descending order of relevance probability, because this is a common metric that is widely used in information retrieval. But when we rank authors, documents, and opinions, this user-focused thinking is no longer comprehensive[2]. In today's ranking system, you need to consider not only the relevance of the results, but also the fairness of the results.

To balance this conflicting metric, we try to use a variety of algorithms to rank documents so as to satisfy certain fairness concepts while ensuring relevance. We evaluated the impact of these two constraints against the evaluation criteria provided[3]. In this article, we will show how to use our framework to achieve relevance and fairness in document ordering.

This article is organized as follows: the first part is the data preprocessing, the second part is the document relevance ranking, then the ranking results are adjusted according to fairness, and the third part is the summary.

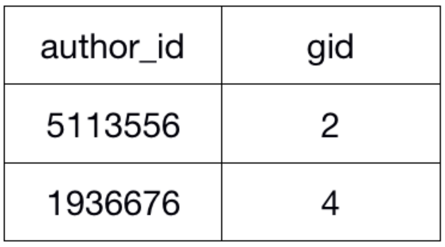
2 Data processing

The given data set is a Semantic(S2) open corpus from the Allen Institute of Artificial Intelligence(http://api.semanticscholar.org/corpus/), containing 47 1GB data files. Most of the data contains doc_id, DOI, title, Abstract, Authors, and the data fields of a small part may be empty. There is also a query-sequence-generator.py file for generating more than 600 queries. Since the data set is too large and the data is in json format, it is in line with the characteristics of mongodb, so we consider importing the data into the mongodb database to enhance the query processing capability of the data. There are approximately 46,747,044 data after importing into the database. Finally, the data is indexed by doc_id, because it is often necessary to query the data according to doc_id. We paired each query with the corresponding document and found that there were only 4641 pairs of (query, document, relevance) data. The data format of each training set is as Figure 1.

```
seq={qid,query,doc_id,title,document,relevance,authors}
```

Figure 1: The data format of each training set

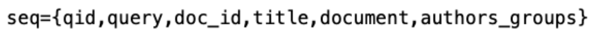
This is because the number of queries is relatively small, so there are only limited training samples. Despite this, there are still more than 1000 pieces of data in the 4641 training data, doc_content, authors or title is empty. As a result, the amount of data is less and the training model will be less effective. The official has also considered this problem, given a script file that samples data by distribution, where we sample 100,000 pieces of data from a limited sample for training according to a given script file. In addition, the most important thing we need to consider is the author's Fairness and Relevance. The official also gives the correspondence between author and group, and as illustrated in Figure 2.



author_id	gid
5113556	2
1936676	4

Figure 2: The correspondence between author and group

Because in the end we evaluate, we need to calculate the Fairness of each author, so we need to add the author_id and gid pair to the evaluation data. The data format of each evaluation data set is as Figure 3.



```
seq={qid,query,doc_id,title,document,authors_groups}
```

Figure 3: The data format of each evaluation data set

In general, the data set itself is large, but the data used for training after pairing is very small, which is disadvantageous for training the neural network model. Although the number of training samples after sampling is increased, it is prone to insufficient training of the model.

3 Model

To begin with, we use the existing model Conv-KNRM[4] to sort documents based on their similarity to the query and documents. The brief principle of the model is as follows.

3.1 Convolutional Kernel-based Neural Ranking Model

Given input query and document, the embedding layer maps their words into distributed representations pre-trained by BERT[5], the convolutional layer generates n-gram embeddings; the cross-match layer matches the query n-grams and document n-grams of different lengths, and forms the translation matrices; the kernel pooling layer generates soft-TF features and the learning-to-rank (LeToR) layer combines them to the ranking score. Conv-KNRM can be learned end-to-end.

Conv-KNRM, compared with KNRM, adds n-gram convolution and increases the level of the original model. Conv-KNRM can capture more subtle semantic entities and cross-granularity is more fine.

3.1.1 N-gram Composing and Cross-matching

Given a query q and a document d, Conv-KNRM embeds their words by a word embedding layer, composes n-grams with a CNN layer, and cross-matches query n-grams and document n-grams of variant lengths to the translation matrices.

Word Embedding Layer: First, the word embedding layer maps each word to an L-dimensional continuous vector (embedding). A text sequence of m words $t_1, ..., t_m$, and is modeled as an $m * L$ matrix.

Convolutional Layer: Second, the convolutional layer applies convolution filters to compose n-grams from the text. For each window of h words, the filter sums up all elements in the h words' embeddings $T_{(i:i+h)}$, weighted by the filter weights $w \in R^h L$,and produces a continuous score.

We use F different filters and get F scores. Then we add a bias and apply a nonlinear activation function, and obtain an F-dimensional embedding for the h-gram. When a convolution filter slides across the boundary of the text, we use $h-1 < PAD >$ symbols for padding.

$$\vec{g}_i^h = \text{relu}\left(W^h \cdot T_{i:i+h} + \vec{b}^h\right), i = 1 \ldots m \quad (1)$$

Cross-match Layer: Translation matrix's elements are the cosine similarity scores between the corresponding query-document n-gram pairs.

$$M_{i,j}^{h_q,h_d} = \cos\left(\vec{g}_i^{h_q}, \vec{g}_j^{h_d}\right) \quad (2)$$

3.1.2 Ranking with N-gram Translations

Conv-KNRM uses the kernel-pooling technique and a learning-to-rank layer to calculate the ranking score using the n-gram translations M.

Kernel Pooling: Kernel-pooling uses K Gaussian kernels to count the soft matches of word or n-gram pairs at K different strength levels. Each kernel summarizes the translation scores as soft-TF counts in the region dened by its mean K_k and width δ_k. As a result, a translation matrix M is pooled to a K-dimensional soft-TF feature vector.

Learning to Rank: The learning-to-rank layer uses linear function to combine the soft-TF ranking features into a ranking score.

Loss Function: We use standard pairwise learning-to-rank loss function to train the model.

$$l = \sum_{q} \sum_{d^+, d^- \in D_q^{+-}} \max(0, 1 - f(q, d^+) + f(q, d^-)) \tag{3}$$

3.2 Adjust for exposure

After sorting D_q according to correlation with KNRM model, the sorted document π_r is obtained. The ranking results at this time only take into account the relevance of query to document d. Suppose the result of KNRM model ranking is the optimal correlation ranking result we can get.

Next, consider the fairness of document sorting.

For the π_q and π_w in the sort result, swapping their positions at this point causes the overall sort result to be less relevant. At the same time, the exposure fairness for authors included in documents q and w may rise or fall.

The result of swapping document q, w is denoted as π^s . The correlation change of the overall ranking result is denoted as $\Delta u_{q,w}^{\pi}$ .Document q precedes document w.

$$\Delta u_{q,w}^{\pi} = \sum_{i=q}^{n} [\gamma^{i-1} \prod_{j=1}^{i-1} (1 - p(s|\pi_j^s))] p(s|\pi_i^s) - \sum_{i=q}^{n} [\gamma^{i-1} \prod_{j=1}^{i-1} (1 - p(s|\pi_j))] p(s|\pi_i) \tag{4}$$

Swapping documents q and w does not affect author relevance. But it changes the exposure of the author.

If the author $a \in q$, then switch the document q and w positions, and the exposure change for author a is denoted as Δe_a^{π} .

$$\Delta e_a^{\pi} = \gamma^{w-1} \prod_{j=1}^{w-1} (1 - p(s|\pi_j^s)) - \gamma^{q-1} \prod_{j=1}^{q-1} (1 - p(s|\pi_j)) \tag{5}$$

Based on this, it is possible to calculate the changes in exposure of all authors after swapping documents q and w.

Each author belongs to one of $|G|$ groups. After calculating the exposure changes of each author, the exposure changes of group g after exchanging documents a and b can be obtained. The change in group g's exposure is denoted as $\Delta \varepsilon_g$.

$$\Delta \varepsilon_g = \sum_{a \in A_g} \Delta e_a \tag{6}$$

The final fair exposure change can be calculated based on the change in the exposure of each group g. In order not to conflict with the symbol of the guideline, the final fair exposure change is denoted as Δf .

$$\Delta f = \sqrt{\sum_{g \in G} \Delta \varepsilon^2 - 2 \Delta \varepsilon_g R_g} \tag{7}$$

Adjust the parameter so that $\Delta u_{q,w}^{\pi}$ and Δf are in the same range.

The next step is to determine whether document q and document w should be swapped based on the correlation change $\Delta u_{q,w}^{\pi}$ and fair exposure changes Δf . If Δf is greater than $\Delta u_{q,w}^{\pi}$, then document q and document w are exchanged. Adjust all the sorting results in this way.

For the sake of time complexity, when deciding whether document q and document w should be switched, only adjacent documents should be taken, and the whole sorting result, π , should only be iterated once.

3.3 Evaluation result

We submitted a ranking result, and the evaluation results are shown in table 1 and table 2. Validation sets are used to group authors based on h-index and level respectively.

Table 1: Evaluation on Groups Based on H-index Scores

SeqID	util		unfairness	
	mean	run	mean	run
0	0.608679	0.549371	0.074779	0.044143
1	0.609166	0.553952	0.076113	0.045979
2	0.609620	0.550504	0.074923	0.046058
3	0.609211	0.547126	0.075145	0.046286
4	0.611033	0.552450	0.075360	0.045370

Table 2: Evaluation on Groups Based on Level Scores

SeqID	util		unfairness	
	mean	run	mean	run
0	0.608679	0.549371	0.041605	0.039809
1	0.609166	0.553952	0.043898	0.051944
2	0.609620	0.550504	0.044432	0.042368
3	0.609211	0.547126	0.042603	0.042178
4	0.611033	0.552450	0.044838	0.037862

As can be seen, after our strategy ranking, the fairness of the overall results is in the middle level among all submitted results. In the h-index evaluation results, even close to the best results.

However, the utility is poor, which is caused by the fact that the ranking based on fairness destroys the relevance of the ranking results.

4 Conclusion

The fairness of sorting in retrieval system is very important. It is very difficult to consider both relevance and fair exposure in the ranking results.

In this Track, our strategy is to first consider the relevance of documents and queries, and then adjust the results according to fair exposure. However, due to time complexity, we did not iterate the adjust process many times.

This approach has a high time complexity and is difficult to practice in systems with short response times. We think it is a good method to consider the fairness and relevance of the ranking results in the same stage of ranking, and it is also a direction that our method can be improved.

References

[1] Ashudeep Singh and Thorsten Joachims. Fairness of exposure in rankings. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2219–2228. ACM, 2018.

[2] L Elisa Celis, Damian Straszak, and Nisheeth K Vishnoi. Ranking with fairness constraints. *arXiv preprint arXiv:1704.06840*, 2017.

[3] Ke Yang and Julia Stoyanovich. Measuring fairness in ranked outputs. In *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*, page 22. ACM, 2017.

[4] Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 126–134. ACM, 2018.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.