Bachelor Informatica

University of Amsterdam

Creating an experimental environment for participating in TREC Fair Ranking Track

René Emmaneel

January 15, 2021

Supervisor(s): Fatemeh Sarvi

Signed:

Abstract

Ranking systems based on queries are used everywhere in the modern online world, from Google search results to jobs and job applicants. To create a ranking of items based on a query, a learning to rank (LTR) algorithm is used to create a model for the ranking of items. Traditional LTR algorithms maximize the utility of the rankings for the user, however increasingly information access systems care about the utility for both the user and producer of the items. The TREC Fair Ranking Track is an annual academic search task with the stated goal of developing a benchmark for evaluating retrieval systems in terms of fairness. Based on the track definition we create a framework with the goal of creating a robust environment for participating in the fair ranking track. We implement a fair support vector machine LTR algorithm utilizing the framework. We discuss the performance of the implemented algorithm based on both the utility and fairness metrics as defined by the TREC Fair Ranking Track, and the usability of the created framework.

Contents

CHAPTER 1

Introduction

Ranking systems allow items to be seen by the users, where the exposure each item gets is significantly determined by the position in the ranking. Traditionally a ranking system would order items based on their relevance so that the system is of maximum utility to the user. However, modern information access systems often influence both the consumer and the producer of the content that they serve. Examples are hiring platforms where it is important that the system has a high utility for both the employer and the job-seeker. But also more traditional environments such as music, book or video recommendations can be considered two-sided because the indirect matching of users to content creators.

The ranking system allows exposure to the producers, and for various applications it is important to fairly give exposure to various groups. While it is unlikely that a universal definition of fairness can be created, there are various attempts to measure unfairness and devise various fair ranking algorithms to ensure that the received attention of a given subject is approximately equal to its deserved attention [1].

To evaluate different fair ranking algorithms, the TREC Fair Ranking Track was created. The stated goal was to develop a benchmark for evaluating retrieval systems in terms of fairness, as well as releasing a dataset for benchmarking fair ranking algorithms. The TREC 2019 Fair Ranking Track was an academic search task, where a set of academic article abstract and queries were submitted to an adademic search engine. The central goal is to provide fair exposure to different groups of authors, where the group definitions can be arbitrary, while keeping the utility of the search results high. Various sets of data has been provided, as well as a detailed evaluation protocol. The data of the 2019 track includes the Semantic Scolar Open Corpus, which consists of data describing various papers. It also includes query data, and query sequences. The metrics and data used by the TREC Fair Ranking Track will be discussed in chapter 2.

To participate in the TREC Fair Ranking Track, an environment needs to be build for the entire process. This process includes handling of the input data, implementing the evaluation algorithm, testing of the framework and implementing LTR machine learning algorithms. To design and implement the framework the following research question will be answered: *Can we construct a modular framework to create an experimental environment for participating in the TREC Fair Ranking Track?*.

We will create one fair ranking algorithm to participate in the Fair Ranking Track, revolving around first creating a traditional LTR algorithm using a rank support vector machine. This is a machine learning algorithm to classify, given a pair of documents, which of the two is more relevant [2]. We use the RankSVM in combination with an adjust for exposure algorithm to transform the LTR algorithm into a learning to fairly rank algorithm [3], which we will call fair RankSVM. The implementation of the algorithm will be used to answer the main research question. In chapter 3 we will provide a detailed explanation of the algorithm we use.

To answer the main research questions we will give in chapter 3 an overview of the components needed for a framework, as well as the implementation details of the framework. In chapter 4 we will give a discussion about the framework and discuss future work.

CHAPTER 2

Theoretical background

In this chapter we will first give an overview of learning to rank for information retrieval, as well as some common techniques that have been studied in previous work. Then we review the importance of fairness in LTR algorithms, and give an overview of previous work on the subject. Finally we introduce the TREC Fair Ranking Track, including the task description, data and used measures.

2.1 Learning to rank

LTR, when applied to document retrieval, is a task as follows. In training a set of queries and documents for each query is provided, as is the relevance judgement for each document. Using this data, a ranking function is created such that the model can predict a ranked list of documents for a query. The two major approaches for creating a ranking function is either a non-learning approach or a LTR approach. Learning approaches for information retrieval (IR) have been widely studied and shown to be useful for ranking articles. [4]

A LTR framework is systematically represented in Figure 3.3. As can be seen in the figure, a training set is needed to train the ranking model. The training data consist typically of n training queries, and m documents for each query. Each document is represented by a feature vector, which is a list of values calculated before training based on the document and query. Each query also has a relevance judgement y, which can for example be the order in which the documents are ranked, or can be a vector representing the relevance for each document based on the query. After training a model has been created, which can be used to rank a set of given documents based on the query.

There are three main LTR algorithms. The pointwise approach, which learns a function for directly predicting the relevance for each document for a query. The pairwise approach, which learns a function that takes in a pair of documents and a query and tries to predict the document that is most relevant for the given query. Lastly there is the listwise approach, which learns a function to directly rank a set of input documents based on a query.

In this thesis we have decided to create both pairwise and listwise ranking algorithm. This decision was made based on the training data available in the TREC Fair Ranking Track. As will be described at the end of this chapter, the training data consist of queries and a certain amount of documents per query, with a boolean relevance score for each document. Because of the boolean relevance score, the pointwise approach would not result in a good ranking model. In chapter 3 we will give an detailed explanation of the feature engineering and the used LTR algorithms.

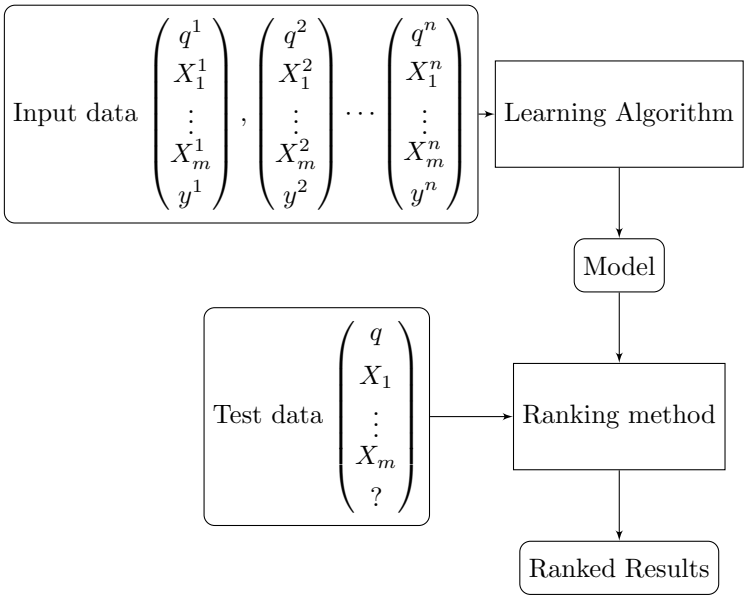

Figure 2.1: LTR framework

2.2 Ethical aspect of fairness

Ranking systems have become one of the dominant forms with which information is presented to the user. The main point of concern considering the ethical aspect of ranking is when the items being ranked are either people or items created by people. Examples include rankings of job seekers, but also content created by user such as scholar articles as in the TREC Fair Ranking Track as described in this paper. In these cases, it is important to consider what impact the ranking has on the producers of the items. The principles behind ranking optimizations have traditionally always been to order items in decreasing order of their probability of relevance, which was first described by the Probability Ranking Principle [5]. Increasingly research have been conducted in creating novel algorithms in which other metrics than only the relevance is considered in ranking of various items.

The metric in this thesis we will consider is fairness, which is defined as the difference between exposure and relevance. The main assumption here is that groups with a high relevance should get a large amount of exposure, and conversely a group with a low amount of relevance should get a low amount of exposure. Here a group is defined as a set of people under a certain condition. This condition could for example be the age group, ethnicity of gender. Fairness, as will be formally defined later, is the sum of difference between exposure and relevance for each group, meaning that if a group gets a low amount of exposure while having a high relevance, the fairness metric will be high. A perfectly fair algorithm would have a fairness metric of 0.

A fair ranking can be considered ethical because of the importance exposure can play in the current world. A certain group getting less exposure than their relevance would indicate can have wide reaching negative results for a society as a whole [6]. Therefore we consider research in fair ranking algorithms a great ehtical necessity.

2.3 Overview of the TREC Fair Ranking Track

In this section we will give an overview of the data provided by the TREC Fair Ranking Track, as well as the metrics used by the track to score the algorithms on fairness and utility. Also this part should be used to talk more about the TREC Fair Ranking Track.

2.3.1 Semantic Scholar Open Research Corpus

The Sementic Scolar Open Research Corpus is a large archive containing metadata of research papers [7]. It contains 47GB data in JSON format, containing information about papers in many different disciplines. A large dataset is needed for sufficient training sample size. The following data is available for most papers.

- **S2 Paper ID**
- **DOI**
- **Title**
- **Year**
- **Abstract**
- **Venue name**
- **Authors** (resolved to author IDs)
- **Inbound citations** (resolved to S2 paper IDs)
- **outbound citations** (resolved to S2 paper IDs)

For the 2019 TREC Fair ranking Track there was no semantic corpus subset available, however a training set was provided. Using this training set we extract the semantic corpus for the documents in the training set using the available API, which gives the data for a specific document given the S2 Paper ID.

Using this data, we can extract the corpus into smaller csv files containing the metadata of each paper and author. The TREC Fair ranking Track for 2020 provided the following 3 csv files, however we also created functionality to extract the following information ourself given the Sementic Scolar Open Research Corpus. The paper metadata is as follows.

paper_metadata.csv:

- **S2 Paper ID**
- **Title**
- **Year**
- **Venue**
- **Number of inbound citations**

Using the same data, we can also extract metadata information for each author of the papers. During the extraction, we count for each author the amount of papers they worked on and the amount of citations each papers has. We use this information for calculating the i10 and H_index scores, as explained below.

author_metadata.csv:

- **Author ID**
- **Name**
- **Citation count:** sum of all citations of each published paper
- **Paper count**
- **i10:** amount of publications with atleast 10 citations
- **H_index:** The maximum value of h such that the given author has published h papers that have each been cited at least h times.

- **H_class:** The class is 'H' if the author has a H_index of atleast 10, and the class is 'L' if the H_index is less than 10

To combine the papers and authors, we need a third linker file, which combined the paper ID with each corresponding author ID, and the position in which they were attributed.

- **S2 Paper ID**
- **Author ID**
- **Position**

2.3.2 Measuring fairness

Fairness can be subjective and differ for various use cases, and as such the TREC Fair ranking Track has provided a fairness definition for the acadamic search task. The goal of the Fair ranking Track is to provide fair exposure to different groups of authors, where the group definition may be arbitrary. Before describing the fairness definition we have to describe the way exposure can influence the discoverability for a certain group, by defining formulas for both the exposure and the relevance an author has.

The exposure an author has is related to the positions of the papers he contributed to, where the first ranked paper has the highest exposure and lower ranked papers have lower exposure. The formula used to measure exposure is the Expected Reciprocal Rank metric [8]. This metric describes the idea that a user is more likely to stop browsing if they have seen a highly relevant document. The user will look at each document in order, and will have a chance to stop browsing depending on the relevance of the document. It also implements an abandonment probability γ, which represent the chance the user will examine the next document, meaning that there is a $1 - \gamma$ chance of abandoning the search. The exposure of the document on place i given the Expected Reciprocal Rank metric is as follows.

$$e_i = \gamma^{i-1} \prod_{j=1}^{i-1} (1 - f(r_j))$$

where $f(r_j)$ is a function to transform the relevance of document d_j into a probability of stopping the search after seeing the document. For the Fair ranking Track the relevance is either 0 or 1, and the function is given as $f(r_d) = 0.7 \cdot r_d$. The continuation probabilty is given as $\gamma = 0.5$.

Because the task is to provide fair exposure to authors instead of documents, we have to calculate the cumulative exposure of the documents which each author has contributed to. Given a ranking of documents π, we can calculate the exposure of author a as follows.

$$e_a^\pi = \sum_{i=1}^{n} (e_i) \cdot I(\pi_i \in \mathcal{D}_a)$$

Where π_i is the document on position i, and $\mathcal{D}_a$ is the documents which include a as an author. Because the final result of the track consist of a series of different rankings, we can calculate the amoritezed exposure of author a as,

$$e_a = \sum_{\pi \in \Pi} e_a^\pi$$

where Π is the sequence of all rankings.

While the exposure of an author is dependent on the ranking of the documents, the relevance of the author is dependent on the relevance of the individual documents the author worked on. The author relevance is simply the sum of the stop probability for each document where a is an author

$$r_a^\pi = \sum_{d \in \mathcal{D}_a} f(r_d)$$

And the amortized relevance for an author is defined as

$$r_a = \sum_{\pi \in \Pi} r_a^\pi$$

We can assume that each author is part of exactly one group. The goal of the Fair ranking Track to provide fair exposure for each group relatively to the relevance for each group. Let $\mathcal{G}$ be the set of all groups and $\mathcal{A}_g$ be the set of all authors in group g. The group exposure and relevance are defined as,

$$\mathcal{E}_g = \frac{\sum_{a \in \mathcal{A}_g} e_a}{\sum_{g' \in \mathcal{G}} \sum_{a \in \mathcal{A}_{g'}} e_a}$$

$$\mathcal{R}_g = \frac{\sum_{a \in \mathcal{A}_g} r_a}{\sum_{g' \in \mathcal{G}} \sum_{a \in \mathcal{A}_{g'}} r_a}$$

Note that $\sum_{g \in \mathcal{G}} \mathcal{E}_g = 1$ and $\sum_{g \in \mathcal{G}} \mathcal{R}_g = 1$. We can use these metrics to calculate the deviation between the exposure and relevance for each group.

$$\Delta_g = \mathcal{E}_g - \mathcal{R}_g$$

Groups should receive exposure proportional to relevance, meaning that a perfectly fair model provides a deviation of zero for each group. We can compute the fair exposure using the square norm.

$$\Delta = \sqrt{\sum_{g \in \mathcal{G}} \Delta_g^2}$$

We measure the relevance of a ranking using the Expected Reciprocal Rank metric, which is the same metric as used in the fairness measurment. Recall that the exposure of document i is defined as,

$$e_i = \gamma^{i-1} \prod_{j=1}^{i-1} (1 - f(r_j))$$

We multiply the exposure of each document with the stop probability to calculate the total utility of a ranking π as follows,

$$u^\pi = \sum_{i=1}^{n} e_i \cdot f(r_i)$$

The average utility of all rankings, $U = \frac{1}{|\Pi|} \sum_{\pi \in \Pi} u^\pi$, will be the final relevance metric.

CHAPTER 3

Method

To participate in the TREC Fair Ranking Track, a framework has to be constructed according to the specifications of the track. This includes extraction of the data, implementation of the fairness metrics and a modular approach of feature extraction and LTR modelling. We first give an overview of the different parts of the framework, then we describe various parts of the framework in more detail. Beside the framework we give the description and implementation details of the LTR models we implemented

3.1 Framework

The framework consist of various files which work together to create a pipeline for the entire process of competing in the TREC Fair Ranking Track. To visualize the interaction between the files a systematic overview is given in figure 3.1. The framework can be split up in three distinct parts, where the first is the extraction of the data and creating the needed subsets. The second part is the feature extraction and the last part is the implemented fair RankSVM algorithm.

3.1.1 Extraction of data

The first step is downloading the data of the documents in the training file. The training file consist of queries and their relevance to in total a few thousand documents, given as their document id. As described in chapter 2, we first retrieve the data from the Sementic Scolar Open Research Corpus based on the document ids in the training set. This is done in the file `create_corpus_subset.py`.

This file first loops over the training set to create a list of document ids in the training set, and then retrieve the S2 Corpus data for each document id, and stores the information in a json file. This is done by executing the API described by the S2 Open Corpus[1].

After all informations for all documents are extracted, the created corpus subset will be used to create smaller csv files as described in chapter 2. These files consist of `paper_metadata.csv`, `author_metadata.csv` and `linker.csv` to link the papers and authors together. These files are stored in the input files, and are used in the feature extraction step. Furthermore, in the input file is also a group sample file stored, which was provided by the TREC Fair Ranking Track.

3.1.2 Feature extraction

To create an usuable input space for the LTR algorithms, each query-document pair has to be described as a feature vector. By calculating various features such as term frequency the documents can be translated in a set of n-dimensional points, which can be used in the implemented RankSVM algorithm. Our implementation has a total of 27 distinct features, and are calculated in `feature_extraction.py`. The output is file in libsvm format, where each line is feature vector

[1] https://api.semanticscholar.org/v1/paper/{document_id}

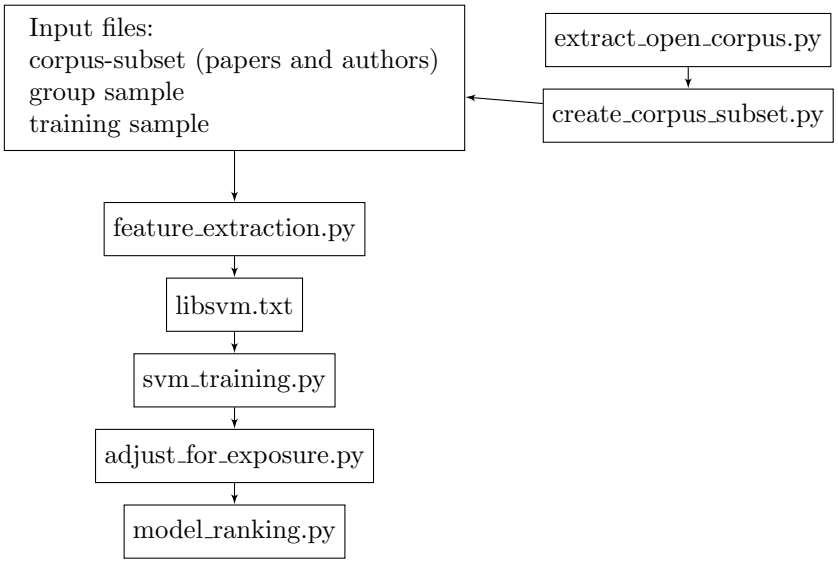

Figure 3.1: Systemtic overview of framework for the TREC Fair Ranking Track

ID	Feature Description
1-3	TF in title, venue, abstract
4-6	IDF in title, venue, abstract
7-9	TF_IDF in title, venue, abstract
10-12	BM25 in title, venue, abstract
13-21	LMIR features in title, venue, abstract
22-24	length of title, venue, abstract
25	amount of in citations

Table 3.1: Features used in the rankSVM model

for a single query-document pair. A small part of feature vector for the 2019 data is shown in picture 3.1.2.

```
0 qid:5438 0:2 1:10.092980591162295 2:0.18745894014040543 3:12.752662792706108 4:11.0104675175592
0 qid:5438 0:12 1:10.092980591162295 2:0.9638605767158267 3:18.917735455211055 4:6.50651382149438
0 qid:5438 0:12 1:10.092980591162295 2:0.6909815243028906 3:18.375288233446927 4:6.95992648794524
0 qid:5438 0:13 1:10.092980591162295 2:0.6556668642188569 3:17.635165707317725 4:7.56290387479298
1 qid:5438 0:23 1:10.092980591162295 2:0.9949286534039548 3:19.33614959683031 4:6.359808965127442
0 qid:5438 0:6 1:10.092980591162295 2:0.4362328277905335 3:15.685238622032799 4:8.646284860214298
0 qid:5438 0:8 1:10.092980591162295 2:0.4813769819601414 3:15.474231165356258 4:8.66879549442146
0 qid:5438 0:15 1:10.092980591162295 2:0.7723400186152123 3:17.539341671119523 4:7.53378538724612
0 qid:5438 0:18 1:10.092980591162295 2:0.7152288512622519 3:18.17273610568088 4:7.266142475287602
0 qid:5438 0:9 1:10.092980591162295 2:0.544847491091292 3:16.951853035573734 4:7.926777077115934
```

Figure 3.2: Part of libsvm file for 2019 data

The first column is a boolean representing whether or not the query-document pair is relevant or not. The second column is the qid of the query, which is used in the RankSVM algorithm. The other columns are all features numbered from 0 to 24 and the calculated values for each feature, as described below.

Features can be categorized in 3 distinct categories. The query-document category means that the feature is dependant on both the feature and document, examples are the term frequency feature. The other two categories are only dependant on the query or the document respectively.

The different features as used in the rankSVM model for the TREC fair ranking track are described in table 3.1. The explanation about these features are listed below.

- Features 1-3 are the term frequency (TF) features. The features will count the amount of occurences of each term in the query in the respective field on the document.

- Features 4-6 are the inverse document frequency (IDF) features. The inverse document frequency function is a measure about how much information a given word provides. The function is defined as follows,

$$IDF(t) = \log \frac{N}{n(t)}$$

with N being the total number of documents, and n(t) being the amount of documents with the term t.

- Features 7-9 are the term frequency - inverse document frequency (TF-IDF) features. For every term of the query, the TF and IDF of the specific term is multiplied. This feature is correlated by a high frequency of a term, which has a low document frequency.

- Features (G-H) are the BM25 features. This is a TF-IDF-like retrieval function, first developed for TREC-3 [9]. The function used is defined as follows.

$$BM25(q,d) = \sum_{q_i \in q \cap d} IDF(q_i) \cdot \frac{TF(q_i,d) \cdot (k+1)}{TF(q_i,d) + k \cdot (1 - b + b \cdot \frac{length(d)}{avglength()})}$$

where length(d) is the amount of words in document d, and avglength() is the average amount of words in all documents. k and b are free variables, which can be set to various values depending on the dataset, but are set to k=1.2 and b=0.75, which experiments have shown to be reasonable values [10].

- Features 13-21 are the language model information retrieval (LMIR) features. LMIR is a statistical language model, with the goal of predicting the probability of the document's language model generating the terms of the query [11]. The way LMIR aims to find this probability is by combining the various probabilities of a certain term in the query being generated by the document's model ($P(t \mid d)$), which gives the equation as follows.

$$P(q \mid d) = \prod_{i=1}^{M} P(t_i \mid d)$$

where $t_1, \cdots, t_M$ are the terms in query q and $P(t \mid d)$ is the document's language model. There are many variants of LMIR, which mainly differ in the use of the document's language model, of which we use three different methods as described by Zhai and Lafferty [12].

- Features 13-15 are the Jelinek Mercer LMIR features. The language model is based on a combination of the percentage of occurence of a term in the given document, as well as the percentage of occurence of a term in all documents combined. The document's language model can then be constructed as follows.

$$P(t_i \mid d) = (1 - \lambda)\frac{TF(t_i,d)}{LEN(d)} + \lambda \frac{TOT\_TF(t_i,C)}{TOT\_LEN(C)}$$

where $TOT\_TF(t_i, C)$ is the term frequency of term i in the entire corpus, $TOT\_LEN(C)$ is the total amount of terms in the corpus and λ is the smoothing factor, which is set to 0.1 in our framework.

- Features 16-18 are the Dirichlet LMIR features.

$$P(t_i \mid d) = \frac{TF(t_i,d) + \mu TOT\_TF(t_i,C)}{TOT\_LEN(C) + \mu}$$

with μ is 2000

- Features 19-21 are the Absolute discount LMIR features.

$$P(t_i \mid d) = \frac{max(TF(t_i, d) - \delta, 0)}{TOT\_TF(t_i, C)} + \sigma \frac{TOT\_TF(t_i, C)}{TOT\_LEN(C)}, \text{with } \sigma = \delta \frac{UNIQUE(d)}{LEN(d)}$$

  with μ is 0.7

- Features 22-25 are features various other features, including the length in words of various parts of the documents, as well as the amount of citations a documents has and the year of publication.

These features have been chosen as they provide a wide variety of query-document dependant features, and the features take the title, venue and abstract into account because those data points are available for most documents and can provide a strong indicator for relevance. Because the framework should be made modular, it is easy to provide a wide variety of different features.

3.1.3 Training the models

The framework has an environment for creating ranking models depending on the libsvm file. The input of the `svm_training.py` is a libsvm.txt file, and the output is a model based on the RankSVM algorithm, which will be explained in detail in the next section.

This output model is used `model_ranking.py` for calculating the TREC Fair Ranking Track metrics based on the training set. This is done by first calculating the ranking based on the model previousily calculated, and then calculating the metric based on the ranking.

3.2 LTR Models

As describe in the framework section, we have implemented a LTR algorithms with the goal of providing a baseline for the framework. In this section we will give the definition of the used algorithms.

3.2.1 Ranking Support Vector Machine

Support Vector Machine (SVM) algorithm is a supervised learning model with the goal of categorizing data represented as points in space [13]. Given a set of training examples, each being labeled in one of the two possible categories, the SVM algorithm builds a model to categorize new examples in one of the two categories. The training examples are represented as a combination of a label and a feature vector, and the model output is represented as the maximum-margin hyperplane that divides the points of one category and the other, so that the distance from the hyperplane and the closest point from either group is maximized. Utilizing the SVM algorithm to categorize documents as relevant or not has been used to create information retrieval models before [14].

Ranking SVM is a variant of the above described SVM algorithm. The goal of the algorithm is to learn a model which, given two documents and a query, learns which document is more relevant [2]. The main benefit of the pairwise approach is that creating a model directly creates a way to sort documents by relevance utilizing the model.

The algorithm starts by creating a dataset of points with labels, where each point is the difference between two feature vectors of documents with the same query, and the relevance is either 1 or -1, depending on if the first document is more relevant than the second document or not. After the new dataset is created, we use the standard SVM algorithm to create a model to seperate the points with different labels.

Finding the maximum-margin hyperplane to seperate the two categories is done using the python library sklearn, which provides an efficient way to calculate the model. After the machine learning algorithm is done finding the optimal hyperplane, we have created the ranking model. Because the model determines which document is more relevant for a given query, we can use this model in combination with a sorting algorithm to create a ranking of documents and a query.

The algorithm will keep iterating until the error is under the given tolerence, which can be set before executing the algorithm.

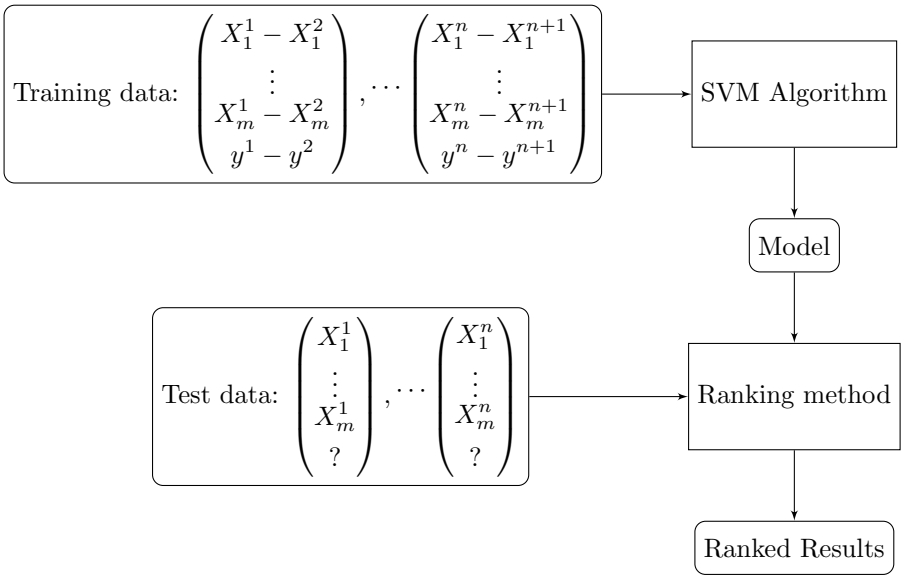

Figure 3.3: Learning to rank framework

3.2.2 Fair RankSVM

To turn the above described ranking algorithm to a fair ranking algorithm, a reranking algorithm is used, which reranks documents based on exposure [3]. We assume that the results of the ranksvm algorithm is optimised for relevance, and therefore reranking will decrease relevance. When swapping the ranked documents π_q and π_w the relevance decrease and the exposure fairness will either decrease or increase. To determine wether the documents should be swapped or not, we have to calculate the change in relevance and fairness, as denoted by Δ_r and Δ_f respectively.

The change in relevance is calculated by the difference in relevance between the ranking when the documents are swapped and when they are not, using the relevance metric as described in chapter 2:

$$e_i = \gamma^{i-1} \prod_{j=1}^{i-1} (1 - f(r_j))$$

$$u^{\pi} = \sum_{i=1}^{n} e_i \cdot f(r_i)$$

$$\Delta_r = u^{\pi'} - u^{\pi}$$

Where π' is the swapped ranking and π is the original ranking. $f(r_i)$ is a custom function to estimate the relevance for a certain document. The relevance metric as used by the TREC Fair Ranking Track is defined as $f(r_i) = r_i * 0.7$. We first have to estimate the relevance of a document using the rankSVM model. Because the rankSVM model is a pairwise algorithm, we do not have an easy way to estimate the relevance of a document. However, we can make a rough estimate by using the ranking model. By subtracting the feature vector of the most relevant document and the document we want to know the relevance of, and calculating the distance of the resulting point with the distance of the calculated maximum-margin hyperplane, we can estimate the relevance of the document. If the distance is low, we assume the relevance is only slightly lower than the most relevant document, while if the distance is large we assume the relevance is much lower than the most relevant document.

Afterwards the total fairness change has to be calculated using the TREC Fair Ranking Track metric. For calculating this metric a group defintion has to be provided.

$$\Delta = \sqrt{\sum_{g \in \mathcal{G}} (\mathcal{E}_g - \mathcal{R}_g)^2}$$

$$\Delta_f = \Delta^{\pi'} - \Delta^{\pi}$$

When both the Δ_f and Δ_r have been calculated, we can compare the two and see if we should swap the two documents. If Δ_f is greater than Δ_r, we choose to swap the two documents.

For the sake of time complexity, only items directly adjacent to each other are considered to be swapped. Furthermore the whole sorting result, π, is only iterated once.

CHAPTER 4

Discussion

We have created a robust framework for participating in the TREC Fair Ranking Track. The framework consist of functionality of extracting data from the S2 Open Corpus, functionality to create feature vectors and the possibility of creating a learning to rank algorithm to create a fair ranking model. We have used the framework to create the fair RankSVM algorithm. In this section we discuss the usability of the framework and the implemented LTR algorithm.

By splitting up the pipeline of the framework in seperate modules, the framework has shown to be modular. We will discuss the three distinct sections separately. The first section of the framework is the functionality of extracting data from the S2 Open Corpus has shown to be useful in downloading and extracting the data from the documents in the training set. This functionality will be easy to use in future tracks, or in other functionality using the S2 Open Corpus. The second part of the framework is the ability to create features to transform query-document pairs into feature vectors. The framework can be changed to use any number of features, which can be changed depending on use case, which results in a more modular framework.

The final part of the framework consist of the LTR algorithm. We have implemented a fair RankSVM algorithm, to show that the framework can be used to implement fair ranking algorithms for participating in the TREC Fair Ranking Track.

Based on the resulting framework we can answer the research question: We have shown that we can create a modular framework for participating in the TREC Fair Ranking Track.

4.1 Future work

For future work we can focus more on the implemented fair ranking algorithm. The implemented Fair RankSVM is compared to other algorithms that participated in previous Fair Ranking Tracks rather primitive. We can use the created framework for further experiments with exisisting or new ranking algorithms. One algorithm that would be interesting to implement is the Fair-PG-Rank [15]. The proposed LTR algorithm has as main benefit over Fair RankSVM the ability to utilize the goal of fairness in the loss function, which is lacking Fair RankSVM.

Bibliography

[1] Asia J. Biega, Krishna P. Gummadi, and Gerhard Weikum. Equity of attention: Amortizing individual fairness in rankings. *CoRR*, abs/1805.01788, 2018.

[2] Thorsten Joachims. Optimizing search engines using clickthrough data. KDD '02, page 133–142, New York, NY, USA, 2002. Association for Computing Machinery.

[3] Meng Wang, Haopeng Zhang, Fuhuai Liang, Bin Feng, and Di Zhao. ICT at TREC 2019: Fair ranking track. In Ellen M. Voorhees and Angela Ellis, editors, *Proceedings of the Twenty-Eighth Text REtrieval Conference, TREC 2019, Gaithersburg, Maryland, USA, November 13-15, 2019*, volume 1250 of *NIST Special Publication*. National Institute of Standards and Technology (NIST), 2019.

[4] Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. Letor: A benchmark collection for research on learning to rank for information retrieval. 2016.

[5] Stephen Robertson. The probability ranking principle in ir. *Journal of Documentation*, 33:294–304, 12 1977.

[6] Solon Barocas and Andrew D. Selbst. Big data's disparate impact. *California Law Review*, 104(3):671–732, 2016.

[7] Waleed Ammar, Dirk Groeneveld, Chandra Bhagavatula, Iz Beltagy, Miles Crawford, Doug Downey, Jason Dunkelberger, Ahmed Elgohary, Sergey Feldman, Vu Ha, Rodney Kinney, Sebastian Kohlmeier, Kyle Lo, Tyler Murray, Hsu-Han Ooi, Matthew Peters, Joanna Power, Sam Skjonsberg, Lucy Lu Wang, Chris Wilhelm, Zheng Yuan, Madeleine van Zuylen, and Oren Etzioni. Construction of the literature graph in semantic scholar. In *NAACL*, 2018.

[8] Olivier Chapelle, Donald Metlzer, Ya Zhang, and Pierre Grinspan. Expected reciprocal rank for graded relevance. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, page 621–630, New York, NY, USA, 2009. Association for Computing Machinery.

[9] S.E. Robertson, S. Walker, S. Jones, M.M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. pages 109–126, 1996.

[10] Hinrich Schütze Christopher D. Manning, Prabhakar Raghavan. An introduction to information retrieval. *Cambridge University Press*, page 270, 2009.

[11] Tao Tao, Xuanhui Wang, Qiaozhu Mei, and ChengXiang Zhai. Language model information retrieval with document expansion. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 407–414, New York City, USA, June 2006. Association for Computational Linguistics.

[12] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, page 334–342, New York, NY, USA, 2001. Association for Computing Machinery.

[13] Corinna Cortes and Vladimir Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995.

[14] Ramesh Nallapati. Discriminative models for information retrieval. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, page 64–71, New York, NY, USA, 2004. Association for Computing Machinery.

[15] Ashudeep Singh and Thorsten Joachims. Policy learning for fairness in ranking, 2019.