Investigating fair rankers under the expected exposure framework

by

Maaike Visser
Student number: 4597265

This thesis is submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Science.
Track: Data Science and Technology.

Defense date:

Thesis advisor:	C. Hauff
Faculty:	Faculty of Electrical Engineering, Mathematics and Computer Science
Report style:	TU Delft Report Style, modifications by Daan Zwaneveld

TU Delft

Preface

This thesis concludes my master studies in Computer Science. It is the conclusion of a time in my life in which I have grown more as a person than I could see beforehand. I'm excited to see what the future will bring.

This thesis could not have come to fruition without the help of the following people.

I express my gratitude to my thesis supervisor, Claudia Hauff, whose guidance has helped me become a more confident and independent researcher. I also thank Arthur Câmara and Tim Draws for sharing their time and experience. Special thanks to Till Kletti and Andres Ferraro who took the time to explain to me the details of their academic writings. Lastly, my thanks to Lydia Chen for taking a seat in my thesis committee.

Special thanks to Steve for sharing your insights and your support, and Francesca for being a ray of sunshine in my life. Thanks to all of my friends for sharing your support when I needed it.

Lastly, gratitude and love for my mama who helped me through the highs and lows of writing this thesis. Without her help and guidance it would not have come to completion.

Maaike Visser
Delft, August 2022

Abstract

As the amount of information available in the world grows, Information Retrieval (IR) systems have become an integral part of day to day life. They determine what subset of the large pool of information is shown to people. IR algorithms determine which items should be returned in response to a query and rank the results in a ranked list.

Recently, concerns about the fairness of IR algorithms have surfaced. In particular, research is being done into whether IR algorithms are fair to *producers*, people or organizations that provide the items that are retrieved by IR algorithms. The higher an item is in the ranked list, the more attention it receives from users. This attention translates to benefits for the producers, e.g. fame or financial compensation.

In this thesis we investigate the fairness of IR algorithms in terms of a specific measure for provider fairness: the Expected Exposure Loss (EEL). This measure measures whether the providers of equally relevant items receive the same amount of attention in expectation. EEL was first proposed as part of the 2020 TREC Fair Ranking track (FAIR-TREC), which also provided a matching dataset. We investigate for two IR systems whether they achieve fairness on this dataset. We conduct a failure analysis and propose improvements for both systems.

We find that for a system that always returns the same ranking it is not useful to improve its accuracy, but rather that it benefits most from fairness-aware post-processing. By contrast, a fairness-aware system *does* benefit from more emphasis on accuracy. We note that the generalizability of our investigation is limited due to the small size of the FAIR-TREC 2020 dataset and recommend that a larger dataset be made available.

Contents

List of Figures

List of Tables

1

Introduction

Information Retrieval (IR) systems are an indispensable part our day-to-day lives. IR algorithms are at the core of our access to information: they determine the results we get on Google, the songs we see on Spotify, the videos we get recommended on YouTube, and so on. The ubiquity of IR systems has motivated a recent increase in attention for the socio-technical dimensions of IR algorithms, such as fairness, accountability, and transparency, as evidenced by the establishment of the FACTS-IR workshop at SIGIR 2019[1], the acceptance of multiple IR-oriented papers at FAccT[2] from 2018 onward, and a growing body of literature on the topic in general.

Most people are familiar with IR and related systems from the point of view of the *user*. A user is a person who has a certain *information need*, such as "I wonder if it is going to rain today?", and who interacts with an IR system to meet their information need. However, IR and related systems often have further stakeholders. Burke [12] first identified these stakeholders in the context of recommender systems. Aside from the users[3] there are the providers and the platform itself.

Providers are people or organizations that create or otherwise supply the subjects that are ranked and presented to users, for example an artist who makes their songs available on a streaming platform where they are returned in response to a user's query. Providers can also themselves be subjects, for example on job search or networking platforms such as LinkedIn. On such websites a person makes a profile representing themselves, which then is returned in response to a query from someone looking for job applicants.

The *platform* is the interface between the user and the provider. It comprises amongst other things a collection of subjects, the IR algorithm that selects subjects

[1] fate-events.github.io/facts-ir/

[2] facctconference.org

[3] Burke [12] talks about *consumers* rather than users. We use the term users to signify that we are not only talking about commercial platforms.

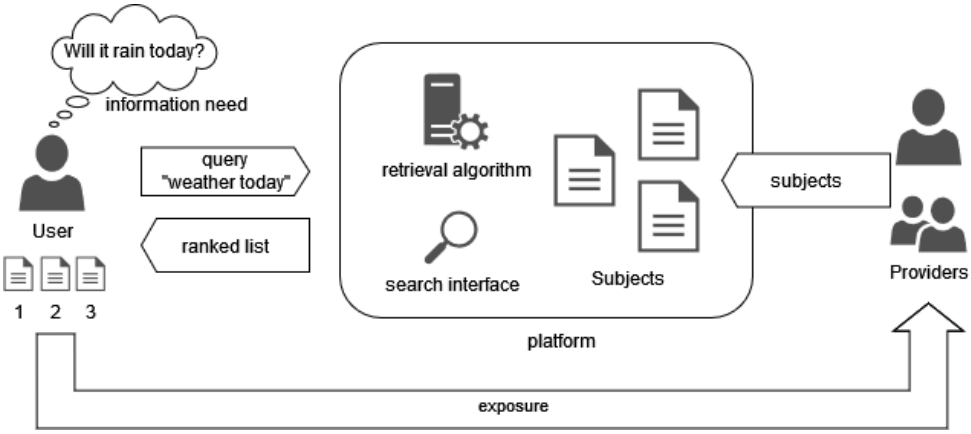


Figure 1.1: Interaction between user, platform, and provider.

for rankings, and the means of entering queries and inspecting results. Figure 1.1 illustrates the relationship between user, platform, and providers.

Each stakeholder derives benefits from participating in the system. For the user, the benefit is to receive useful information and satisfy their information need. For providers the benefit is to receive e.g. financial compensation, acclaim, or opportunities [12]. For platforms, the benefit is usually financial, e.g. ad revenue or subscriptions, but it can also be more esoteric as in the case of non-profit academic search engines that exist to facilitate further research[4].

Historically, the focus of IR has been to optimize ranking algorithms for the benefit of users. The benefit is most commonly measured in terms of a system's *effectiveness* (also called *utility*); its ability to retrieve the *right subject*s in response to a query [17]. Effectiveness can be measured in different ways depending on the underlying beliefs about the user's behavior. For example, a user may be looking for a specific website, say "YouTube". In that case, they only need to receive one result, namely "youtube.com", and preferably it should be the top result. A suitable measure for this scenario would be Mean Reciprocal Rank (MRR) [48], which is higher if a relevant subject occurs at a better rank. At their core, most IR algorithms optimize rankings by predicting a score for each subject that represents the probability of relevance to the user, and then ordering the subjects by decreasing score [42]. This approach is seated in the Probability Ranking Principle (PRP) [31]. It has been shown that ranking according to PRP optimizes most commonly used effectiveness measures [32], given that the relevance criterion is binary and the relevance judgments of subjects within a ranking are independent of each other [37].

Although PRP can be suitable to optimize for effectiveness, it has been shown that optimizing rankings for effectiveness alone can lead to unfair outcomes for providers. Providers only receive benefits when their subjects receive the users'

[4]E.g. arxiv.org.

attention [4]: a user has to listen to your song on a streaming platform before you get royalties; a user has to read your academic paper for you to gain acclaim. The expected amount of attention subjects receive is called *exposure* [51]. For most applications, attention and exposure can be used interchangeably. In this thesis we do not make the distinction unless necessary. In addition, we do not distinguish between exposure for a subject and benefit for its provider unless needed.

The reason not all subjects in a ranking receive the same amount of exposure is due to *position bias*. It has been shown that users tend to pay the most attention to the subjects at the top of the ranking, even when those subjects are not the most relevant [16]. In addition, most modern search systems do not display multiple subjects per rank, even when both subjects are equally relevant. Instead, one is always ranked better than the other and thus receives more attention.

To illustrate the position bias consider the following function for the amount of exposure a subject d receives in ranking π:

$$\text{exposure}(d, \pi) = 1/\pi(d), \tag{1.1}$$

where $\pi(d)$ is the rank of the document. Now let's say we have d_1 and d_2 with $rel(d_1) = 1$ and $rel(d_1) = 0.99$. The difference in *relevance* is 0.01, but the difference in *exposure* is 0.5.

The impact of the position bias effect is exacerbated when the same ranking is returned multiple times, for instance with common queries like `covid 19`[5], queries repeated over time [4] like `activities near me today` , or when using Learning-to-Rank (LtR) algorithms that propagate biases embedded in their training data [41].

The recent attention on fairness in IR has given rise to a multitude of fairness measures and fair IR algorithms. Different attempts have already been made at bringing structure to this emerging field of research. For example, Zehlike, Yang, and Stoyanovich [51] provide a taxonomy of different notions of provider fairness. In addition, in 2019 the first TREC Fair Ranking track (FAIR-TREC) was organized[6] with the goal of developing a benchmark to jointly evaluate IR systems in terms of fairness and effectiveness [5]. Since FAIR-TREC is the only existing benchmark for fair IR we use it as the benchmark for this thesis.

Subsequent editions of FAIR-TREC have expanded the task, datasets, and measures under consideration. In this thesis we focus primarily on fairness as it is defined in FAIR-TREC 2020. We go deeper into the precise fairness definition in Section 2.3.4. For now it is important to know that FAIR-TREC 2020 is concerned with *amortized meritocratic group* fairness. We go into the meaning of each of these terms further in Section 2.3.1, but in short:

- *Amortized* fairness refers to the fact that the fairness is measured across multiple rankings.
- *Meritocratic* fairness refers to the notion that providers should receive attention proportional to the relevance of their subjects to a query [42, 4].

[5] Or whichever plague is most topical for the reader.
[6] https://fair-trec.github.io/2019/index.html

- *Group* fairness means that rankings should be fair towards *groups* of providers.

At the time of starting this thesis, the only editions of FAIR-TREC that were complete and had been published were FAIR-TREC 2019 and 2020. The reason we focus on the 2020 edition is that since it resolves small issues that were present during FAIR-TREC 2019, such as a smaller corpus or missing documents in the training and evaluation data. By 2020 the track had also started receiving more submissions. All in all, FAIR-TREC is a more refined benchmark.

1.1. Research objective

The goal of this thesis is to contribute to the developing field of fair IR in two ways. The first way is to contribute directly by improving existing algorithms. We select two submissions to FAIR-TREC 2020 and identify a number of ways in which we can enhance their performance.

The other way is to contribute on a meta-level. One of the core tools in the IR research toolbox is robust evaluation of IR systems [17, Section 8.1]. We contribute to the practice of robust evaluation by identifying strengths and weaknesses of the FAIR-TREC 2020 track in particular.

To summarize, we address the following research question:

RQ: Which factors affect the fairness of amortized rankers in the context of FAIR-TREC 2020?

1.2. Scientific contributions

We analyze two IR systems with the FAIR-TREC 2020 benchmark. We identify flaws of both the rankers and the FAIR-TREC benchmark, propose and implement improvements, and determine significant influences on fairness. Further contributions include:

- Adapted failure analysis method for amortized fair rankers
- Insights into the strenghts and weaknesses of the FAIR-TREC 2020 benchmark
- Improvements to two amortized fair rankers
- Codebase for reproducibility purposes[7]

1.3. Toy example: Ranking in an academic search engine

Throughout this thesis we use the following toy example to illustrate the different rankers. The toy example is based on FAIR-TREC 2019 and 2020 [5, 6].

Consider an academic search engine that returns documents in response to a user query. Each document has one or more authors. Table 1.1a shows a dataset

[7] github.com/pilmus/thesis

Doc	rel	ρ	Authors
d_1	1	1.0	a_1, a_3
d_2	1	0.75	a_2, a_4
d_3	0	0.5	a_3
d_4	0	0.25	a_4, a_5

(a) A dataset $\mathcal{D}$ of 4 documents. Each document has one or more authors; each author has a value for the h-index and econ level attributes. rel is the true relevance label of each document, ρ is the estimated probability of relevance.

π_1^*	π_2^*	π_3^*	π_4^*
d_1	d_1	d_2	d_2
d_2	d_1	d_1	d_1
d_3	d_4	d_3	d_4
d_4	d_3	d_4	d_3

(b) All optimal (all relevant documents before all non-relevant documents) rankings of the dataset.

Author	H-level	Econ-level
a_1	High	Advanced
a_2	High	Developing
a_3	Low	Advanced
a_4	Low	Developing
a_5	High	Advanced

(c) Group membership of each author.

Table 1.1: A toy example to illustrate various concepts in this thesis.

of documents and their authors. Each author has two grouping attributes: h-index and level of economic development of the country of affiliation. Each document has a true binary relevance label $rel(d)$ and an estimated probability of relevance $\rho(d)$ relative to a query.

Table 1.1b shows all optimal rankings, that is all rankings that have all relevant documents before all non-relevant documents. π_1 and π_2 give all of the exposure to a single h-index based group, namely `h-level=High`, but they divide the exposure equally among the econ-level groups `econ=High` and `econ=Low`.

1.4. Outline

This thesis is structured as follows. We give background information about the FAIR-TREC 2019 and 2020 benchmarks in Chapter 2. We select two rankers to analyze in depth. We describe the selection process and the selected rankers in Chapter 3. We conduct a failure analysis of the rankers in Chapter 4 and propose improvements to remedy detected failures in Chapter 5. We finish with a conclusion and recommendations for future work in Chapter 6.

2

The TREC Fair Ranking Track

We analyze fair IR algorithms with the help of the benchmark developed the TREC Fair Ranking track. The goal of this track is to help with the development of algorithms that are fair to different groups of providers [5]. The providers in this specific scenario are the *authors* of each document. From this moment on, we use providers and authors interchangeably. FAIR-TREC was first organized in 2019 and every year since. Every edition has a corpus, fairness measures, and a task definition.

We use the corpus, measures, and task of FAIR-TREC 2020 in particular. FAIR-TREC 2020 was one of the two editions of the track to have been completed and published at the time of writing this thesis. Compared to the 2019 edition, it resolves small issues such as missing documents in the corpus. By 2020 the track had also gained more recognition, resulting in a larger number of submissions. Even so, we also describe the components of FAIR-TREC 2019 to give the reader insight into how the benchmark evolved and to give them the full background required for the next chapter.

This chapter is organized as follows. In Section 2.1 we describe the specific tasks participants were asked to carry out. We describe the corpus in Section 2.2 and in particular the annotation process of the training and test data in Section 2.2.1. In Section 2.3 we go into great detail into the evaluation measures for both tracks. We describe the underlying browsing model and explain how we go from the browsing model to the measures themselves. We describe the evaluation measure for FAIR-TREC 2019 in Section 2.3.3 and the measure for FAIR-TREC 2020 in Section 2.3.4.

2.1. Task

FAIR-TREC 2019 and 2020 had two tasks: a retrieval task and a reranking task. The goal of the retrieval task is to select, rank, and return documents from the *full* corpus. The goal of the reranking task is to take a smaller set of documents for each query and put those in the optimal order. Retrieval typically is more computationally

intensive than reranking. In this thesis we focus on the reranking task.

The aim of the reranking task was to simulate an academic search engine. The task was to rerank documents that had been retrieved in response to queries that were submitted to a production academic search engine. The goal was to rerank each document in such a way that exposure would be fairly distributed across different groups of authors while maintaining a certain level of relevance towards consumers. However, the exact grouping of authors was unknown at system submission time. The idea then was to create IR systems that are robust to unknown groupings [5].

Unknown groupings can occur in real life for a number of reasons. The attribute on which we want to group could be unknown or unknowable, e.g. if we want to group people by gender but this information is not disclosed. We may also want to be fair to all groupings, in which case there is not a single attribute to base the grouping on. Fair ranking that is robust to unknown groupings is a way to remedy these issues.

2.2. Corpus

The corpora for FAIR-TREC 2019 and FAIR-TREC 2020 are snapshots of the Semantic Scholar Open Corpus (SSOC) from the Allen Institute for Artificial Intelligence [1]. The documents in the corpus consist of abstracts and associated metadata for academic articles. In addition, a portion of the documents were annotated to be used as training and test data. Listing 2.1c show the different pieces of information that come together to form a sin

We use the versions of the corpus as they were available at the time the submissions to the respective tracks were produced. For FAIR-TREC 2019 this is the version from 16-08-2019; for FAIR-TREC 2020 this is the version from 27-05-2020. We retrieve the corpora by following the official track instructions [1] with some changes where necessary, see Appendix A.1. Table 2.1 contains details on the corpora. As we can see, `corp2020` is roughly 4 times larger than `corp2019`. This solidifies our choice to focus on the reranking task, since the larger a corpus is the harder the retrieval task becomes.

We indexed the corpora into Elasticsearch[2] with the standard analyzer and tokenizer. The fields we indexed are listed in Table 2.2.

2.2.1. Annotated training and test sets

Recall that the goal of FAIR-TREC 2019 and 2020 is to provide fairness for arbitrary groups of providers. To facilitate developing algorithms that are fair in this way parts of the corpora were annotated with two pieces of information: relevance labels that indicate how relevant a document is to a query, and a group label that indicates which group the document belongs to.

The queries whose documents were annotated were selected from real query

[1] github.com/fair-trec/fair-trec-tools
[2] elastic.co/guide/index.html

```
{
    "qid": 68960,
    "query": "arteterapia",
    "frequency": 1.27e-05,
    "documents":
        [
            {
                "doc_id": "
                   f5ec283a31ae68e07ca52077130894784660898 8
                   ", "relevance": 0
            },
            ...
        ]
}
```

(a) The query text and frequence of query 68960 as well as the relevance of a document to this query.

```
{
    "id": "f5ec283a31ae68e07ca52077130894784660898 8",
    "title": "Trabajar con las Emociones en arteterapia",
    "paperAbstract": "Emotions have a fundamental importance
        in human development ...",
    "entities": [],
    "author_names": ["Norman Duncan"],
    "author_ids": ["78317264"],
    "inCitations": 2,
    "outCitations": 0,
    "year": 2007,
    "venue": "",
    "journalName": "",
    "journalVolume": "2",
    "sources": [],
    "doi": "10.5209/rev_ARTE.2007.v2.9757",
    "fields_of_study": ["Psychology"]
}
```

(b) The indexed fields of the document in Elasticsearch.

```
{
    "id": "f5ec283a31ae68e07ca52077130894784660898 8",
    "HLevel": "None",
    "EconLevel": "Advanced"
}
```

(c) The group annotations for the document.

Listing 2.1: Three pieces of information making up a fully annotated document: the relevance to a query, the indexed fields, and the group annotations.

Name	Year	Number of documents	Number of queries
`corp2019`	2019	46 947 044	
`corp2019train`	2019	4490	652
`corp2019test`	2019	4027	635
`corp2020`	2020	186 723 411	
`corp2020train`	2020	4649	200
`corp2020train_split`	2020	4187	180
`corp2020val_split`	2020	495	20
`corp2020test`	2020	4693	200

Table 2.1: The sizes of the corpora, training sets, and test sets for FAIR-TREC 2019 and 2020.

logs of queries submitted to the Semantic Scholar search engine[3]. The logs contained queries that were issued to the search engine in an undisclosed period for FAIR-TREC 2019 [5] and between February 14 2020 and April 27 2020 for FAIR-TREC 2020 [6]. Because each query had to be annotated manually, a number of filtering steps were used to select the most useful queries.

The documents for each selected query were annotated with two pieces of information: the relevance label and the country of operation of its authors [5, 6]. The relevance labels were derived from the number of clicks logged in the query log. The click data was converted to binary relevance based on a threshold selected by the track organizers. The country of operation of each author was determined manually by NIST assessors.

The annotated training and test sets contain roughly the same number of documents, 4000 to 4500 documents each. However, in 2019 those documents were divided over around 630 to 650 queries while in 2020 the number of queries was 200. To aid with training and evaluation in later parts of this thesis we further split `corp2020train` into a 90/10 training/validation.

2.2.2. Example groupings

While the goal of FAIR-TREC 2019 and 2020 was to provide fairness to *arbitrary* groups, the evaluation measures for both editions are computed across a specific grouping as we will see in Sections 2.3.3 and 2.3.4. As such, to aid participants in developing their rankers each edition of the track provided a sample grouping. After submissions had been closed, the organizers of each edition also released a second grouping which was used to evaluate the submissions.

The attributes for the groupings were the same for each year: there was a group based on the h-index of the authors of each document, and a group based on the economic development level of the country of origin. The economic development level was derived from the country of affiliation as determined by the NIST assessors

[3] `semanticscholar.org`

Field	Type	2019	2020
author_ids	text	x	x
author_names	text	x	x
doi	text	x	x
entities	text	x	
inCitations	numerical	x	x
outCitations	numerical	x	x
journalName	text	x	x
journalVolume	text	x	x
paperAbstract	text	x	x
sources	text	x	x
title	text	x	x
venue	text	x	x
year	numerical	x	x
fieldsOfStudy	text		x

Table 2.2: The fields for the Elasticsearch indices of `corp2019` and `corp2020`.

combined with information from the International Monetary Fund [4]. We refer to the h-index-based grouping as the H-level grouping $\mathcal{G}_h$ and to the economic-level based grouping as the Econ-level grouping $\mathcal{G}_e$.

In 2019, the H-level group had five levels: $h < 5$, $5 \leq h < 15$, $15 \leq h < 30$, $h \geq 30$, and *None*. The Econ-level group had three levels: *Advanced*, *Developing*, and *None*. In 2019, a *paper* could belong to multiple groups if its *authors* belonged to different groups. E.g.: if a paper has one author from an *Advanced* country and one from a *Developing* country, the paper would be annotated with both *Developing* and *Advanced.*

In 2020, the H-level group had four levels: *High*, *Mixed*, *Low*, and *None*. The Econ-level group also had four levels: *Advanced*, *Mixed*, *Developing*, and *None*.

As we see later, FAIR-TREC 2019 and 2020 use different evaluation measures, but both measures take a grouping as an input. We assume that the grouping of the respective year is always used with the measure of the respective year, e.g. we do not distinguish between $\mathcal{G}_{e,2019}$ and $\mathcal{G}_{e,2020}$ because $\mathcal{G}_{e,2019}$ is only used with the Unf measure and $\mathcal{G}_{e,2020}$ is only used with the EEL measure.

The groupings are summarized in Table 2.3.

2.3. Evaluation

Measuring the performance of systems is an integral part of IR research; without evaluation it is impossible to improve systems in a targeted and systematic manner [17, Ch. 8]. As such, an integral part of the FAIR-TREC benchmark is the evaluation measure.

IR measures are based on some underlying perception of the behavior of the

[4] imf.org

Attribute	Level	Count
H-index	$h < 5$	1062
	$5 \leq h < 15$	1667
	$15 \leq h < 30$	1295
	$h \geq 30$	1065
	None	140
Economic level	Advanced	2502
	Developing	445
	None	57

(a) FAIR-TREC 2019.

Attribute	Level	Count
H-index	High	774
	Low	426
	Mixed	849
	None	2777
Economic level	Advanced	3374
	Mixed	543
	Developing	243
	None	533

(b) FAIR-TREC 2020.

Table 2.3: Number of documents in each of the groups based on h-index and economic development level of the country of origin for FAIR-TREC 2019 and 2020.

user, as well as an idea of what constitutes good performance by the system. For example, the Expected Reciprocal Rank (ERR) measure assumes that users are less likely to inspect documents lower in a ranking if they are satisfied with documents higher in the ranking [14]. Correspondingly, for a good performance in terms of ERR the most relevant documents need to be at the top of a ranking. This is in contrast to a measure like precision (P), which simply measures which fraction of retrieved documents is relevant without regard of the relative position of relevant and non-relevant documents [30, Eq. 8.1].

ERR and P only look at the utility of a ranking to the user. Fairness measures additionally are based on a particular notion of fairness. Zehlike, Yang, and Stoyanovich [51] created a taxonomy of types of fairness which includes such distinctions as *group* versus *individual* fairness. However, as the field of fair IR is still developing no universal agreement on the types of fairness has of yet been reached.

In this section we describe the evaluation measures used in FAIR-TREC 2019 and 2020. We first go into the notion of fairness underpinning each measure in Section 2.3.1. We continue in Section 2.3.2 with a description of the ERR browsing model which forms the basis for the fairness measures in FAIR-TREC 2019 and

2020. Then we describe the particular fairness measures for FAIR-TREC 2019 and FAIR-TREC 2020 in Section 2.3.3 and Section 2.3.4 respectively.

2.3.1. Amortized meritocratic group fairness

As mentioned in Chapter 1, FAIR-TREC 2019 and 2020 aim to achieve *amortized meritocratic group* fairness. We now go deeper into what each of these terms means.

Amortized fairness One of the ways in which fairness can arise is through the *position bias* effect. Recall from Chapter 1 that position bias arises from the fact that users tend to pay more attention to highly ranked documents, even when those documents are not the most relevant [16].

Biega, Gummadi, and Weikum [4] were the first to note that it is not possible to combat the position bias effect within a single ranking, at least not when each rank can only contain one document. As a solution they propose to *amortize* the fairness measure across multiple rankings. Or in other words: each document should be treated fairly *on average*.

Meritocratic fairness There are different ways in views on what constitutes fair treatment. We could require that all groups are treated the same regardless of attributes like group size, an approach called *statistical parity* (see e.g. [33]). We could distribute exposure proportionally to the size of a group to achieve *demographic parity* (see e.g. [42, Sec. 4.1]). Or we could require that the amount of exposure a document or group of documents receives is proportional to how relevant those documents are to users, a concept we'll call *meritocratic fairness* [42, 4].

Group fairness Providers can be assigned to groups according to any possible attribute, e.g. gender, race, country of origin, eye color, etc. Attributes can be defined to determine a *protected* group [41], e.g. gender if having a particular gender would be penalized as happened with an algorithm Amazon used for a time to select candidates for job interviews [18]. It is also possible to not define a protected group, in which the goal is simply to be fair to each of the groups [52].

2.3.2. Browsing model: Extended Expected Reciprocal Rank

FAIR-TREC 2019 and FAIR-TREC 2020 use the Extended Expected Reciprocal Rank (e-ERR) browsing model [14, sec. 7.2] to determine the amount of exposure a document receives at a rank. Under e-ERR, the exposure received by the document at rank j is equal to the probability that the user did *not* stop browsing before reaching the current document. A user may stop browsing either because they have found a relevant document or because they got frustrated and stopped looking. Therefore, the probability the user has already stopped browsing in turn depends on how relevant the earlier documents were to the query.

Document exposure The exposure on a document d at rank j in a ranking π for query q is given by

$$\text{e-ERR}(d, \pi, q) = \gamma^{\pi(d)-1} \prod_{j=1}^{\pi(d)-1} 1 - f(rel(\pi_j, q))) \tag{2.1}$$

where γ is the so-called *patience parameter* that represents how likely the user is to give up if they do not find a result they like, $f(x) = \kappa x$ is a function that converts the relevance x to a probability of stopping at this document, $rel(d, q)$ is the relevance of a document to query q, and $\pi(d)$ is the rank of document d ranking π and π_j is the document at rank j.

Table 2.4 shows for each document in π_1^* in Table 1.1b the amount of exposure it receives under e-ERR with $\gamma = 0.5$ and $f(x) = 0.5 \cdot x$.

Document	rank	rel	e-ERR
d_1	1	1	1.0
d_2	2	1	0.25
d_3	3	0	0.125
d_4	4	0	0.0625

Table 2.4: The amount of exposure each document in π_1^* of the toy example in Table 1.1b receives under e-ERR with $\gamma = 0.5$ and $f(x) = 0.5 \cdot x$.

Author exposure FAIR-TREC 2019 and 2020 are concerned with fairness towards groups of authors. Authors receive exposure from each document that they have written. Therefore, the exposure for a single author a over a ranking π is

$$\text{e-ERR}_a(a, \pi, q) = \sum_{j=1}^{|\pi|} \left[\text{e-ERR}(\pi_j, \pi, q)\right] \mathbb{I}\left(\pi_j \in \mathcal{D}(a)\right), \tag{2.2}$$

where $\mathcal{D}(a)$ are the documents in the corpus $\mathcal{D}$ that were written by a.

Table 2.5 shows for each author how much exposure they receive from π_1^* in Table 1.1b under e-ERR$_a$ with $\gamma = 0.5$ and $f(x) = 0.5 \cdot x$. We see that because a_3 and a_4 contributed to multiple documents, they receive more exposure than a_1 and a_2.

Group exposure Authors can further be grouped together based on certain attributes such as their country of affiliation. The group exposure is the first point of divergence between the fairness measures used for FAIR-TREC 2019 and 2020.

We first go into the group exposure for FAIR-TREC 2019. For FAIR-TREC 2019, the group exposure over a single ranking is given by

Author	e-ERR$_a$
a_1	1.0
a_2	0.25
a_3	1.125
a_4	0.3125
a_5	0.0625

Table 2.5: The amount of exposure each author receives in π_1^* of the toy example in Table 1.1b under e-ERR with $\gamma = 0.5$ and $f(x) = 0.5 \cdot x$.

$$\text{e-ERR}_g^{2019}(g, \pi, q) = \frac{\sum_{a \in \mathcal{A}(g)} \text{e-ERR}_a(a, \pi, q)}{\sum_{g' \in \mathcal{G}} \sum_{a \in \mathcal{A}(g')} \text{e-ERR}_a(a, \pi, q)}. \tag{2.3}$$

Here $\mathcal{A}(g)$ are all of the *authors* that belong to *group* g. e-ERR$_g^{2019}$ defines the exposure for a single group g as a fraction of the total exposure received by *all* groups.

For FAIR-TREC 2020, the group exposure over a single ranking is given by

$$\text{e-ERR}_g^{2020}(g, \pi, q) = \sum_{a \in \mathcal{A}(g)} \text{e-ERR}_a(a, \pi, q). \tag{2.4}$$

Note the difference with e-ERR$_g^{2020}$: the exposure is no longer a fraction of the total amount available.

Table 2.6 shows for each group how much exposure it receives from π_1^* in Table 1.1b under e-ERR$_g$ using either the H-level or the Econ-level grouping. The group memberships of each author are shown in Table 1.1c. Like in the previous examples, $\gamma = 0.5$ and $f(x) = 0.5 \cdot x$. We see that the group membership makes a difference in how even the distribution of exposure across groups seems; when measuring with e-ERR$_g^{2019}$ for the H-level group the difference between the High and the Low level is 0.04 while the difference for Econ-level group is 0.63.

We also note that while the values for e-ERR$_g^{2019}$ and e-ERR$_g^{2020}$ are different, since each value of e-ERR$_g^{2019}$ is scaled by the same constant, the ratio between the amount of exposure on each group remains the same.

Group	Level	e-ERR$_g^{2019}$	e-ERR$_g^{2020}$
H-level	High	0.48	1.3
	Low	0.52	1.4
Econ-level	Advanced	0.83	2.3
	Developing	0.20	0.56

Table 2.6: The amount of exposure each group receives in π_1^* of the toy example in Table 1.1b under e-ERR with $\gamma = 0.5$ and $f(x) = 0.5 \cdot x$. Group membership is as in Table 1.1c.

Exposure over a sequence of rankings Since we want to measure *amortized* fairness we need to be able to compute exposure over a sequence of rankings in addition to over a single ranking. For each of the measures of exposure covered thus far, the exposure over a sequence of rankings Π is simply the sum of the exposure over each individual ranking.

The ranking sequence equivalents of each of the previous measures are:

$$\text{e-ERR}(d, \Pi) = \sum_{\pi \in \Pi} \text{e-ERR}(d, \pi, Q(\pi)) \tag{2.5}$$

$$\text{e-ERR}_a(a, \Pi) = \sum_{\pi \in \Pi} \text{e-ERR}_a(a, \pi, Q(\pi)) \tag{2.6}$$

$$\text{e-ERR}_g^{2019}(g, \Pi) = \sum_{\pi \in \Pi} \text{e-ERR}_g^{2019}(g, \pi, Q(\pi)) \tag{2.7}$$

$$\text{e-ERR}_g^{2020}(g, \Pi) = \sum_{\pi \in \Pi} \text{e-ERR}_g^{2020}(g, \pi, Q(\pi)) \tag{2.8}$$

Here Q is the set of all queries and $Q(\pi)$ is the particular query in response to which π was returned. We make this distinction clear because Π can be created in response to the same or in response to different queries. However, whenever it is clear that all rankings in Π were returned in response to the *same* query, we omit the $Q(\pi)$ term or simply use q.

2.3.3. FAIR-TREC2019: Unf

The fairness measure for FAIR-TREC 2019 is based on the concept of Equity of Attention [4]. The idea is that each group of documents should receive exposure proportional to its relevance across a sequence of rankings. The rankings can be in response to different information needs, that is: different queries, different time points, different users, etc.

The exposure across a sequence Π is computed as in Equation (2.7). Additionally, we need to compute the relevance of documents, groups, and authors. The relevance of an a for a single ranking π depends on the relevance of each of the documents the author has written and on the query q in response to which the ranking was returned.

$$\mathcal{R}_a(a, \pi, q) = \sum_{d \in \mathcal{D}(a)} f(rel(d, q)), \tag{2.9}$$

Here again $\mathcal{D}(a)$ is the set of documents written by a. The relevance rel can be computed in different ways; for now it is assumed to be known.

Table 2.7 shows for each author how relevant they are based on the relevances and the rankings in Table 1.1. As before, $f(x) = 0.5 \cdot x$.

The relevance of a group of authors is defined similarly as the exposure, namely as a fraction of the total amount of relevance of all documents:

Author	$\mathcal{R}_a$
a_1	0.5
a_2	0.5
a_3	0.5
a_4	0.5
a_5	0.0

Table 2.7: The relevance of each author computed as in Equation (2.9) based on π_1^* of the toy example in Table 1.1b and the relevance labels in Table 1.1a.

$$\mathcal{R}_g(g, \pi, q) = \frac{\sum_{a \in \mathcal{A}(g)} \mathcal{R}_a(a, \pi, q)}{\sum_{g' \in \mathcal{G}} \sum_{a \in \mathcal{A}(g')} \mathcal{R}_a(a, \pi, q)}. \tag{2.10}$$

Table 2.8 shows for each group how relevant they based on the relevances and the rankings in Table 1.1. As before, $f(x) = 0.5 \cdot x$. We see that each group has the same relevance, even though High for the H-level group and Advanced for the Econ-level group have one more member. This is because the document written by a_5 is not relevant.

Group	Level	$\mathcal{R}_g$
H-level	High	0.5
	Low	0.5
Econ-level	Advanced	0.5
	Developing	0.5

Table 2.8: The relevance of each group computed as in Equation (2.10) based on π_1^* of the toy example in Table 1.1.

As with exposure, the relevance of a group of authors across a sequence of rankings is simply the sum of the relevance of each individual ranking:

$$\mathcal{R}_g(g, \Pi) = \sum_{\pi \in \Pi} \mathcal{R}_g(g, \pi, Q(\pi)). \tag{2.11}$$

The fairness measure used is the difference between e-ERR$_g$ and $\mathcal{R}_g$. It does not have an official name, but here we call it Unf:

$$\text{Unf}(g, \Pi) = \text{e-ERR}_g(g, \Pi) - \mathcal{R}_g(g, \Pi) \tag{2.12}$$

The difference between exposure and relevance for all groups is the l_2-norm of the Unf for each individual group:

$$\text{Unf}(\mathcal{G}, \Pi) = \sqrt{\sum_{g \in \mathcal{G}} \text{Unf}(g, \Pi)^2}. \tag{2.13}$$

Because we take the l_2 norm Unf penalizes both *over* and *under* exposure, ie. the value grows both when a group gets too much exposure relative to its relevance and when it gets too little. This is suitable for a scenario where we do not know the groupings in advance; the safest bet is to allocate the perfect amount of exposure instead of over-exposing some groups. Intuitively, Unf corresponds to a realistic browsing scenario where a single user issues queries for multiple topics [19].

Table 2.9 shows the value of Unf as defined in Equation (2.12) for each group in Table 1.1 across all the rankings in Table 1.1b, as well as the overall Unf as defined in Equation (2.13). Where relevant, $\gamma = 0.5$ and $f(x) = 0.5 \cdot x$. We see that although for both the H-level group and the Econ-level group the sum of the unfairness is close to 0 the overall Unf for the H-level grouping is smaller because the magnitude of the Unf for each group is smaller.

Group	Level	Unf(g, Π)
H-level	High	-0.023
	Low	0.023
	Unf$(\mathcal{G}_h, \Pi)$	0.032
Econ-level	Advanced	0.33
	Developing	-0.3
	Unf$(\mathcal{G}_e, \Pi)$	0.044

Table 2.9: The Unf for each group in Table 1.1 separately as well as together.

2.3.4. FAIR-TREC2020: EEL

The fairness measure for FAIR-TREC 2020 is based on the concept of expected exposure [20]. Understanding this measure requires us to understand a number of concepts: stochastic rankers, target expected exposure, actual expected exposure, and finally the measure itself, Expected Exposure Loss (EEL). We go through these concepts in order.

Stochastic and deterministic rankers Most IR systems are based around a *deterministic* ranker. A deterministic ranker learns a single function which it uses to predict scores for each document. As a result, a deterministic ranking always returns the same result in response to each query. A ranker is deterministic even if it includes e.g. random swapping of results to introduce variation, because at its core it still computes results from a single function.

By contrast, a stochastic ranker learns a *probability distribution* over the documents for a query; for each document the ranker learns the probability it will appear at a certain rank for a certain query. Each time a specific query is issued to the system, it draws a sample from the learned distribution. The sample is then shown to the user.

Equal expected exposure Diaz et al. [20] note that it is impossible for deterministic rankers to achieve perfect fairness. They create a single ranking for each query, and as we have seen before a single ranking cannot be perfectly fair due to the position bias effect [4]. What's more, if a query is repeated more than once any discrepancy in exposure accumulates and the unfairness is exacerbated.

Take for example π_1^* from Table 1.1b. Let's say we create a sequence where we repeat this ranking 4 times, so $\Pi = [\pi_1^*, \pi_1^*, \pi_1^*, \pi_1^*]$. We compute the exposure as in Equation (2.5) with $\gamma = 0.5$ and $f(x) = 0.5 \cdot x$. Assuming that each ranking was returned after the other, we can say that each ranking was returned at a *time* t. The accumulated exposure for each document at time t is shown in Table 2.10. We see that at time $t = 1$, the difference in exposure between d_1 and d_4 is 0.9375. However, by $t = 4$ it has grown to 3.75.

It may seem reasonable for the difference in exposure between d_1 and d_4 to grow over time. After all, $rel(d_1) = 1$ and $rel(d_4) = 0$. However, the difference in exposure between the two relevant documents in the toy example, d_1 and d_2, has also grown, from 0.75 to 3.0.

Document	$t = 1$	$t = 2$	$t = 3$	$t = 4$
d_1	1.0	2.0	3.0	4.0
d_2	0.25	0.5	0.75	1.0
d_3	0.125	0.25	0.375	0.5
d_4	0.0625	0.125	0.1875	0.25

Table 2.10: The accumulated exposure on each document over repetitions of π_1^* from Table 1.1b.

The principle of equal expected exposure states that *in expectation* all *equally relevant* documents should receive *equal exposure*. In other words, across infinitely many rankings, each relevant document should receive equally as much exposure as each other relevant document, and the same for each non-relevant document.

For stochastic rankers the expected amount of exposure is simply the expected value of the learned probability distribution. However, for deterministic rankers no such distribution is available. In that case, we can simulate a stochastic ranker by returning a large number of rankings for the same query and computing the expected exposure over those.

Target and actual expected exposure Under the equal expected exposure framework, unfairness is measured as the difference between the *target* expected exposure $\mathcal{E}^*$ and the *actual* expected exposure $\mathcal{E}$.

The target expected exposure $\mathcal{E}^*$ is determined as follows. Consider the rankings in Table 1.1b. In each ranking, each relevant document comes before each non-relevant document; all of these rankings are *optimal*. Now consider a stochastic oracle ranker that returns each of the rankings in Table 1.1b with equal probability or equivalently, a deterministic oracle that returns each of the rankings in order. Then the expected exposure on each of the relevant documents is the same and the expected exposure on each of the non-relevant documents is the same.

The actual expected exposure $\mathcal{E}$ is the amount of exposure each document actually receives. For example, if a stochastic ranker returns π_1^* and π_2^* in Table 1.1b with probability 0.5, and π_3^* and π_4^* with probability 0, d_1 has a higher *actual* expected exposure than d_2 and d_3 has a higher *actual* expected exposure than d_4.

Expected exposure with e-ERR. The concrete values that $\mathcal{E}^*$ and $\mathcal{E}$ take depend on the underlying browsing model. We now show how to compute $\mathcal{E}^*$ and $\mathcal{E}$ with the browsing model used in FAIR-TREC 2020, e-ERR.

The documents in the FAIR-TREC 2020 corpus are annotated with *binary* relevance grades. Therefore we need to compute only two values for $\mathcal{E}^*$, one for each relevance grade. Starting with the relevant documents, assume there are s relevant documents in the ranking. In an ideal scenario, each of the s relevant documents should occupy each of the s highest positions equally often. We compute $\mathcal{E}^*$ for relevant documents as

$$\mathcal{E}^*_{rel}(s) = \frac{1}{s}\sum_{k=1}^{s}\gamma^{k-1}\prod_{j=1}^{k-1}(1-f(1)) \tag{2.14}$$

$$= \frac{1}{s}\sum_{j=1}^{s}\gamma^{k-1}\prod_{j=1}^{k-1}(1-\kappa) \tag{2.15}$$

$$= \frac{1}{s}\sum_{j=1}^{s}\left(\gamma\left(1-\kappa\right)\right)^{k-1} \tag{2.16}$$

$$= \frac{1-\left(\gamma\left(1-\kappa\right)\right)^{s}}{s\left(1-\gamma\left(1-\kappa\right)\right)} \tag{2.17}$$

This expression computes the amount of exposure that would be on each rank $j \in [1..s]$ considering each document in the top s positions has a relevance of 1 and then take the average value.

For non-relevant documents, we need to take into account that the amount of exposure they receive is diminished by the fact that there are s relevant documents preceding them. Let the total number of documents in the optimal ranking be N. Then the $\mathcal{E}^*$ on each of the $N-s$ documents in the $N-s$ bottom positions is

$$\mathcal{E}^*_{non-rel}(s,N) = \frac{1}{N-s}\sum_{k=s+1}^{N-s}\gamma^{k-1}\prod_{j=1}^{k-1}\left(1-f(rel(\pi_j,\pi^*)\right) \tag{2.18}$$

$$= \frac{1}{N-s}\sum_{k=s+1}^{N-s}\gamma^{k-1}(1-\kappa)^s \tag{2.19}$$

$$= \frac{1}{N-s}\sum_{k=s+1}^{N-s}\gamma^{k-1}(1-\kappa)^s \tag{2.20}$$

$$= \frac{(1-\kappa)^s(\gamma^s-\gamma^N)}{(N-s)(1-\gamma)} \tag{2.21}$$

The target expected exposure for an arbitrary document d for query q is then

$$\mathcal{E}^*_{doc}(d,q) = \begin{cases}\mathcal{E}^*_{rel}(s) & rel(d,q)=1\\ \mathcal{E}^*_{non-rel}(N-s) & rel(d,q)=0\end{cases} \tag{2.22}$$

To make this concrete, consider the toy example. The rankings in Table 1.1b are all optimal rankings because they put each relevant before each non-relevant document. In these rankings, $s=2$ and $N=4$. If we set $\gamma=0.5$ and $\kappa=0.5$, then

$$\mathcal{E}^*_{rel}(2) = \frac{1-(\gamma(1-\kappa))^s}{s(1-\gamma(1-\kappa))} \tag{2.23}$$

$$= \frac{1-(0.5(1-0.5))^2}{2(1-0.5(1-0.5))} \tag{2.24}$$

$$= 0.625 \tag{2.25}$$

and

$$\mathcal{E}^*_{non-rel}(2,4) = \frac{(1-\kappa)^s(\gamma^s-\gamma^N)}{(N-s)(1-\gamma)} \tag{2.26}$$

$$= \frac{(1-0.5)^2(0.5^2-0.5^4)}{2(1-0.5)} \tag{2.27}$$

$$= 0.046875 \tag{2.28}$$

$$\tag{2.29}$$

The actual expected exposure $\mathcal{E}$ depends on the particular ranker. Let Π^*_q be every possible ranking a ranker can generate in response to query q and let $\sigma(\pi)$ be the probability that the ranker generated the ranking π. Then the actual expected exposure on document d is simply the expectation over all the rankings in Π^*_q:

$$\mathcal{E}(d, \Pi_q^*) = \sum_{\pi \in \Pi_q^*} \sigma(\pi) \text{e-ERR}(d, \pi, q). \tag{2.30}$$

For deterministic rankers with a randomization component σ is not actually known, but we can approximate it by generating a large number of rankings and empirically finding the distribution that way.

To illustrate the difference between $\mathcal{E}^*$ and $\mathcal{E}$ consider a deterministic ranker that always returns π_1^* in Table 1.1b. In other words, for this ranker $\sigma(\pi_1^*) = 1$ and $\sigma = 0$ for all other possible rankings. Then the $\mathcal{E}$ of d_1 is the same as in Table 2.4:

$$\begin{aligned}
\mathcal{E}(d_1, \Pi_q^*) &= \sum_{\pi \in \Pi_q^*} \sigma(\pi) \text{e-ERR}(d_1, \pi, q) && (2.31)\\
&= \sigma(\pi_1^*) \text{e-ERR}(d_1, \pi_1^*, q) && (2.32)\\
&= \text{e-ERR}(d_1, \pi_1^*, q) && (2.33)\\
&= 1.0. && (2.34)
\end{aligned}$$

If we used a ranker that has $\sigma(\pi_1^*) = 1$ for π_2^* and $\sigma = 0$ for all other rankers, $\mathcal{E}(\Pi_q^*, d_1) = 0.25$. However, $\mathcal{E}_{doc}^*(d_1, q)$ remains the same, namely 0.625 as we computed earlier.

The expected exposure is propagated from documents to *authors* and *groups of authors* in the same way as we did for exposure in Section 2.3.2. For a ranking of length N with s relevant documents, the target expected exposure of an author a is given by

$$\begin{aligned}
\mathcal{E}_a^*(a, \pi, q) = \mathcal{E}_{rel}^*(s) \left|\{\pi_j \in \mathcal{D}(a) \wedge rel(\pi_j, q) = 1\}\right| && (2.35)\\
+ \mathcal{E}_{non-rel}^*(N - s) \left|\{\pi_j \in \mathcal{D}(a) \wedge rel(\pi_j, q) = 1\}\right|, && (2.36)
\end{aligned}$$

and the target expected exposure for a group is given by

$$\mathcal{E}_g^*(g, \pi, q) = \sum_{a \in \mathcal{A}(g)} \mathcal{E}_a^*(a, \pi, q). \tag{2.37}$$

Recall that here, $\mathcal{A}(g)$ is the set of authors in group g.

The actual expected exposure for authors and groups are given by

$$\mathcal{E}_a(a, \Pi^*) = \sum_{j=1}^{N} \left[\mathcal{E}(\pi_j, \Pi^*)\right](I) \left(\pi_j \in \mathcal{D}(a)\right) \tag{2.38}$$

and

$$\mathcal{E}_g(g, \Pi^*) = \sum_{a \in \mathcal{A}(g)} \mathcal{E}_a(a, \Pi^*) \tag{2.39}$$

Fairness in FAIR-TREC 2020: Expected Exposure Loss In practise, it is difficult to use $\mathcal{E}$ directly. Most common rankers are deterministic rather than stochastic. For the deterministic rankers, it is not always feasible to generate a the number of rankings that is required to empirically determine the probability distribution σ. For instance, many rankers used in production environments update their ranking function based on feedback by users, meaning that σ would change over time. Therefore, it can be more useful to look at the *difference* between target *expected* exposure and actual realized (so *not* expected) exposure.

This idea is the basis of the fairness measure used in FAIR-TREC 2020: Expected Exposure Loss (EEL). EEL measures the l_2-norm between the target exposure and actual exposure received by groups of authors across a sequence of queries. The idea is that rather than determining in advance what the actual *expected* exposure of will be, we let a ranker generate a sequence of rankings and for each realized ranking compare how much the resulting exposure differs from the target expected exposure.

To make this concrete, consider the following example. Say we have a ranker R. We do not know the associated value of σ, but we can let R generate rankings. At time $t = 1$, R generates a ranking π_1 of length N in response to query q. Assume for now we know that there are s relevant documents in π_1. Then the Expected Exposure Loss is the l_2 norm of the difference between the *target* expected exposure (Equation (2.37)) and the *actual* exposure (Equation (2.4)):

$$\text{EEL}(\mathcal{G}, \pi) = \sqrt{\sum_{g \in \mathcal{G}} \left(\text{e-ERR}_g^{2020}(g, \pi, q) - \mathcal{E}_g^*(g, \pi, q)\right)^2}. \tag{2.40}$$

If we let R generate a whole sequence Π of rankings, we can measure the loss as

$$\text{EEL}(\mathcal{G}, \Pi) = \sqrt{\sum_{g \in \mathcal{G}} \left(\text{e-ERR}_g^{2020}(g, \Pi, q) - \mathcal{E}_g^*(g, \Pi, q)\right)^2}. \tag{2.41}$$

This is the measure of unfairness as it is used in FAIR-TREC 2020. Note that EEL is computed over a specific grouping $\mathcal{G}$.

The only remaining unknown component here is the target expected exposure over a sequence of rankings $\mathcal{E}_g^*(g, \Pi, q)$. Since the target expected exposure is independent from the particular ranking, to get the $\mathcal{E}^*$ over a *sequence* of rankings we simply multiply the $\mathcal{E}^*$ by the length of the sequence:

$$\mathcal{E}_g^*(g, \Pi, q) = |\Pi| \cdot \sum_{a \in \mathcal{A}(g)} \mathcal{E}_a^*(a, \pi, q). \tag{2.42}$$

The interpretation of EEL is as follows. We can decompose the term $\left(\text{e-ERR}_g^{2020}(g, \Pi, q) - \mathcal{E}_g^*(g, \Pi, q)\right)^2$ in Equation (2.41) into three parts:

$$\underbrace{\sum_{g \in \mathcal{G}} \text{e-ERR}_g^{2020}(g, \Pi, q)^2}_{\text{disparity EEL-D}} - 2 \underbrace{\sum_{g \in \mathcal{G}} \text{e-ERR}_g^{2020}(g, \Pi, q) \mathcal{E}^*_{g,\Pi,q}(g, \Pi)}_{\text{relevance EEL-R}} + \underbrace{\sum_{g \in \mathcal{G}} \mathcal{E}^*_g(g, \Pi, q)^2}_{\text{information need constant}} \tag{2.43}$$

The disparity EEL-D has an intuitive interpretation: for a fixed amount of exposure, it is smallest when the exposure is distributed uniformly over relevant documents [19]. The relevance EEL-R is not one-to-one interpretable as user utility[19] but it reflects how much of the exposure is on relevant documents [6]. From this decomposition we can see that EEL is smallest when (i) the exposure is distributed equally over documents of equal relevance, and (ii) most of the exposure is on relevant documents. Overall, the intuition behind EEL is that it corresponds the behavior of either a stochastic ranker for a single query, or a topic for which queries are issued often [19].

Similarly to Unf, EEL is defined as the l_2 norm of a difference, meaning both over- and under-exposure is punished equally. In contrast to Unf however, EEL is only suitable for comparing rankings that were generated in response to the *same* query. This is due to the $\mathcal{E}^*_g$ term. The target expected exposure changes depending on the number of relevant and non-relevant documents for a query. This means that if Π contained rankings for different queries, the value of EEL would be skewed.

2.4. Summary

In this chapter we described different components of FAIR-TREC 2019 and 2020. We described the tasks, the corpora, and the evaluation measures.

The corpora for FAIR-TREC 2019 and 2020 are snapshots of the Semantic Scholar Open Corpus from the Allen Institute for Artificial Intelligence. Each document in the corpus consists of a number of metadata fields of an academic article, such as the title, abstract, and year. Each corpus is accompanied by annotated training and evaluation data. The training and evaluation data consists of a set of queries and corresponding documents. The documents for each query are annotated with binary relevance labels as well as different group labels, based on the h-index or the country of origin of the paper's authors.

The evaluation measures for FAIR-TREC 2019 and FAIR-TREC 2020 are the *Unfairness* Unf and the *Expected Exposure Loss* EEL respectively. Both measures are based on the e-ERR browsing model. This browsing model favors documents in the top positions and assumes that attention decreases as users inspect documents further down. Additionally, e-ERR incorporates the possibility that a user gives up completely if they do not find a satisfying result quickly enough.

Unf and EEL diverge in how exactly they compute unfairness. Unf measures the difference between the amount of exposure each group of documents receives across multiple rankings, and how relevant thos documents are to a user. The magnitude of the unfairness Unf is straightforwardly defined the l_2 norm of the difference between this group exposure and relevance.

EEL measures the difference in the *expected* exposure on documents, ie. the amount of exposure documents receive across a large number of rankings for the same query. The magnitude of EEL is the l_2 norm of the difference between the *target* expected exposure and the exposure documents have actually received.

The main difference between the measures lies in the fact that Unf is suitable for measuring fairness across rankings for different queries, while EEL is more suitable for measuring fairness across multiple rankings for the same query. Unf corresponds to how a single user's browsing behavior, looking for multiple topics in the same session. By contrast, EEL corresponds to the behavior of either a stochastic ranker on a single topic, or a common topic for which queries are issued often.

3

Rankers

The goal of this thesis is to discover which factors affect the fairness of rankers as measured in terms of the fairness measure associated with the FAIR-TREC 2020 benchmark, the EEL (see Equation (2.41)). There were 15 submissions from 4 teams to FAIR-TREC 2019 and 28 submissions by 6 teams to FAIR-TREC 2020. In the interest of time, we are unable to analyze every single ranker underlying each submission. Instead, we select two rankers we think are interesting and suitable. We implement these rankers ourselves and verify the correctness of our implementations relative to the official submissions.

This chapter is organized as follows. We first explain how we came to select the rankers we analyze in this thesis in Section 3.1 and explain which papers we selected. Then in Section 3.2 we describe how we verified the correctness of our implementations of each selected ranker. the process of reproducing submissions. In sections Section 3.3 and Section 3.4 we describe the rankers we selected.

3.1. Selection process

To select rankers to analyze we looked at a number of characteristics of each submission. We looked at the following characteristics:

- Approach. We wanted to select rankers with a different approach (e.g. Learning-to-Rank, term weighting) to gain a broader insight.
- Performance. We looked at the performance of rankers in terms of the fairness measures. We favored rankers that performed better because we felt any improvements made to those rankers would be more useful.
- Tools. Some submissions used the same tools, e.g. many of the submissions were based on the Elasticsearch[1] engine. We aimed to select rankers that used the same underlying tools to simplify the implementation process.

[1] elastic.co/

- Codebase. For the sake of reproducibility, we favored rankers with a publicly available codebase.
- Communication with the authors. We reached out to the authors of all of the submissions to FAIR-TREC 2019 and FAIR-TREC 2020 with questions about their submissions. Communication with the authors greatly helped with reproducing their submissions.
- Level of detail of companion paper. As we explain in more detail in the next section, we wanted to reproduce the results of the official submissions to verify the correctness of our own implementations. The more detailed the companion paper for a particular submission the easier it is to reproduce.

In the end, we selected one submission to FAIR-TREC 2019 and one submission to FAIR-TREC 2020. The submissions and underlying rankers are fair_LambdaMART with LambdaMART [7] and NLE_META_9_1 with Advantage Controller [27] respectively. We selected these submissions for the following reasons.

The algorithm used to generate the first submission, fair_LambdaMART ,is LambdaMART (LM) [11]. LM is a LtR algorithm commonly used in IR. When it was first conceived, LM won the Yahoo! learning-to-rank challenge [10]. A cross-benchmark comparison of 87 LtR methods confirms that LM is among the top performers [44]. In addition to LM's status as a well-performing LtR algorithm, the codebase used to generate the fair_LambdaMART submission is publicly available[2].

Other points in favor of LM were the detailed companion paper with information on e.g. the features and tools. We were also able to get into contact with the authors and ask for clarification on numerous occasions. The one area where LM lags behind other submissions is performance. LM achieved fairness scores of $\text{Unf}(\mathcal{G}_h, \Pi) = 0.0741$ and $\text{Unf}(\mathcal{G}_e, \Pi) = 0.0855$, the 3rd and 7th worst performance of all submissions to FAIR-TREC 2019 respectively. Even so, its ubiquity in IR applications makes it a suitable baseline implementation in our eyes.

the downsThe algorithm used to generate the second submission, NLE_META_9_1, is Advantage Controller (AC) [27]. AC is a controller-based re-ranking algorithm, ie. it creates multiple rankings for the same query but re-computes the score for the documents at each iteration based on some error term. NLE_META_9_1 was the top submission to FAIR-TREC 2020 with $\text{EEL}(\mathcal{G}_e, \Pi) = 0.428$. Controller-based algorithms are uncommon in fair IR. Morik et al. [32] use a controller-based mechanism for unbiased ranking, but they do this for online rather than offline LtR. Since the controller approach is uncommon, we expect we may be able to to contribute more significantly than if we analyzed an approach that has been investigated in detail before. The companion paper for AC is detailed, but some information is obfuscated or assumed to be known in advance. However, this leads us to another point in favor of AC, namely that the authors were able and willing to answer any remaining questions. A downside of analyzing AC was that no open source code was available, meaning we had to put in more effort to implement it from scratch.

[2] github.com/irgroup/fair-trec

Submission	Measure	Published score	Reproduced score
fair_LambdaMART	Unf$(\mathcal{G}_e, \Pi)$	0.0741	0.0771
NLE_META_9_1	EEL$(\mathcal{G}_e, \Pi)$	0.428	0.429

Table 3.1: The published and reproduced scores for the selected submissions to FAIR-TREC 2019 and 2020. Submission names and published scores are from Biega et al. [5, 6].

3.2. Verifying implementation correctness

To analyze LM and AC in detail we had to implement both algorithms ourselves. We verified the correctness of our implementations by creating runs on the test data of the corresponding benchmark and computing either Unf or EEL. We then compared these values to the published values for fair_LambdaMART and NLE_META_9_1. Once the values for our submissions were within 10% of the published submissions we considered our implementation to be correct. The values we achieved for LM and AC, as well as the published values, are shown in Table 3.1.

Aside from reproducing the results for fair_LambdaMART and NLE_META_9_1 we also attempted to reproduce results for 6 other submissions: fair_random [7] for FAIR-TREC 2019, and Deltr-gammas [25], umd_relfair_ltr [40], and NLE_META_99_1, NLE_TEXT_99_1, and NLE_TEXT_9_1 [27] for FAIR-TREC 2020. Of these, we were able to verify the correctness of fair_random and NLE_META_99_1, NLE_TEXT_99_1, and NLE_TEXT_9_1. NLE_META_99_1, NLE_TEXT_99_1, and NLE_TEXT_9_1 share the underlying ranker of NLE_META_9_1. fair_random is a random re-ranking of each document for each query, making it less interesting to investigate. All submissions starting with NLE are variations of the AC algorithm. The submissions we attempted but were not able to reproduce failed due to a combination of missing details in the accompanying papers, lack of available code, and a lack of time on our part.

3.3. LambdaMart

LM is a pairwise LtR algorithm. What this means is that it learns the *relative* order two documents should have to each other in response to a query. This is in contrast to pointwise algorithms, which learn to predict the individual relevance grades of documents, and listwise algorithms which learn to predict the order of all documents for a query [29].

LM itself is based on the LambdaRank algorithm. LambdaRank is also a pairwise algorithm. It minimizes the loss caused by documents being in the wrong order, ie. a more relevant document occurring below a less relevant document. A key characteristic of LambdaRank is that it approximates the cost-gradients with so-called λ-gradients. The λ-gradient for each document represents the loss that would be caused by swapping this document with any other document.

Consider the visualization in Figure 3.1. The blue bars represent relevant documents while the gray bar represent non-relevant documents. The arrows show the direction and magnitude of the λ-gradients of the relevant documents; because

Figure 3.1: Visualization of the optimization process of LambdaMART. The blue bars represent relevant documents, the gray bars represent non-relevant documents. The arrows indicate in which direction the relevant documents should move to improve the ranking. Figure adapted from Burges [11, Fig. 1].

they are pushed up in the ranking the number of *pairwise* errors diminishes and the quality of the ranking increases overall.

The main downside of LM is that it does not allow for a way to include fairness, especially not the kind of fairness used expressed by Unf or EEL. While it is possible to adapt LM to optimize different objectives [10], it still produces only a single ranking for each query. Unf and EEL are both amortized fairness metrics and since LM cannot vary the ranking for a query it will inevitably achieve a low score.

Multiple open source implementations of LM are available. Bonart [7]'s implementation[3] is based on `pyltr`[4]. We train LM with the features and hyperparameters also used by Bonart [7]. These are listed in Tables 3.2 and 3.3 respectively.

3.4. Advantage Controller

The AC algorithm is based on the concept of a proportional controller. [3]. Note that the algorithm is originally unnamed but that here we call it Advantage Controller.

[3] github.com/irgroup/fair-trec
[4] github.com/jma127/pyltr

Level	Feature
query-document (BM25)	title abstract entities venue journal author's names
document	year number of out-citations number of in-citations
query	query length in characters

Table 3.2: Features used to train LambdaMart.

The idea of AC is that for each query we iteratively create N rankings that when assessed together give a low value for EEL. The scores for each document for each ranking are computed based on the exposure the documents have received over previous rankings.

The scoring function h for query q, document d, and ranking $t < N$ is given by

$$h(d, t, q) = \theta\rho(d, q) - (1 - \theta)A_d(d, t). \tag{3.1}$$

Here $\rho(d, q)$ is the *estimated probability of relevance* of d for q and $A_{doc}(d, t)$ is the *accumulated advantage* of d over the rankings up to the tth ranking.

The estimated probability of relevance of a document $\rho(d, q)$ indicates how likely it is that a document is relevant for a query. Kletti and Renders [27] use *estimated* rather than absolute relevance values to make the method suitable both for scenarios where the relevance labels are available and where they are not. The details on how the ρ-values are generated are not fully clear, but the process involves training a LtR ranker with features based on a number of fields, predicting scores for each document, and normalizing those scores to lie in the $[0, 1]$-range with Isotonic Regression [50] so they can be used as probabilities [36].

We were unable to reproduce the exact process of generating estimated probabilities of relevance. However, Kletti and Renders [27] generously shared with us two sets of estimated probabilities of relevance. We refer to these as ρ_A and ρ_B.

The advantage A_{doc} is based on the concept of expected exposure that EEL is also based on. It tracks how much the actual expected exposure for a document has thus far deviated from its target expected exposure. As Kletti and Renders [27] note, at ranking time we do not typically have access to the true relevance of a document and as such cannot calculate its true $\mathcal{E}^*$ and $\mathcal{E}$.

To circumvent this problem, Kletti and Renders [27] model the true relevance of a document for a query as a realization of a Bernoulli random variable with the estimated relevance of the document for a query as its parameter. Using this, we can compute the expected value of the $\mathcal{E}^*$ of document d for query q as

Hyperparameter	Value
metric	NDCG
learning_rate	0.02
n_estimators	1000
query_subsample	0.5
subsample	1.0
min_samples_split	2
min_samples_leaf	64
max_depth	3
random_state	0
max_features	None
max_leaf_nodes	10
warm_start	true

Table 3.3: Hyperparameter settings used with our implementation of LM. Settings are identical to those used by Bonart [7].

$$\hat{\mathcal{E}}^*(d,q) = \mathbb{E}\left[\mathcal{E}^*_{doc}(d,q)\right] \tag{3.2}$$

$$= \sum_{s=0}^{N-1} PB(s|\boldsymbol{\rho} - \rho(d,q))(\rho(d,q)\mathcal{E}^*_{rel}(s+1) + (1-\rho(d)\mathcal{E}^*_{non-rel}(N-s)) \tag{3.3}$$

(3.4)

Here $\mathcal{E}^*_{rel}$ and $\mathcal{E}^*_{non-rel}$ are computed as in Equations (2.14) and (2.18), $\rho(d,q)$ is the estimated relevance of d and $\boldsymbol{\rho}-\rho(d,q)$ is the vector of the estimated probability of relevant of each document for q, *except* for $\rho(d,q)$. PB is the Poisson-Binomial distribution[5].

The expected $\mathcal{E}^*$ of a group g of documents is the sum of the expected $\mathcal{E}^*$ of the documents in the group:

$$\hat{\mathcal{E}}^*_{group}(g) = \sum_{d \in g} \hat{\mathcal{E}}^*_{doc}(d) \tag{3.5}$$

The target expected exposure does not change over time, so the expected target exposure for group g by the t-th ranking is given by

$$\hat{\mathcal{E}}^*_g(g,t) = t * \hat{\mathcal{E}}^*_g(g) \tag{3.6}$$

The expected actual expected exposure of a document d in a ranking π is given by using the estimated relevance $\rho(d,q)$ in Equation (2.1):

[5]The specific implementation is found here github.com/tsakim/poibin

$$\hat{\mathcal{E}}_d(d,\pi) = \mathbb{E}\left[\mathcal{E}_d|\pi\right] \tag{3.7}$$

$$= \gamma^{1-\pi(d)} \prod_{j=1}^{\pi(d)-1} \left(1 - f(\rho(\pi(j), q))\right) \tag{3.8}$$

The expected actual expected exposure of a group g is the sum of the $\hat{\mathcal{E}}$ of each document in the:

$$\hat{\mathcal{E}}_g(g,\pi) = \sum_{d \in g} \hat{\mathcal{E}}^*_d(d,\pi) \tag{3.9}$$

The expected actual expected exposure of a group g by the tth ranking is

$$\hat{\mathcal{E}}_g(g,t) = \sum_{t'=1}^{t} \hat{\mathcal{E}}_g(g,\pi_{t'}) \tag{3.10}$$

We can now properly define the advantage from Equation (3.1). The advantage of a document is defined as the arithmetic mean of the advantage of the group it belongs to, A_g. The advantage of a single group g over the rankings up to ranking t is given by

$$A_g(g,t) = \left(\hat{\mathcal{E}}_g(g,t-1) - \hat{\mathcal{E}}^*_g(g,t-1)\right)^2 \text{sign}\left(\hat{\mathcal{E}}_g(g,t-1) - \hat{\mathcal{E}}^*_g(g,t-1)\right) \tag{3.11}$$

and finally the advantage of a document d at time t is

$$A_d(d,t) = \frac{1}{|\mathcal{G}(d)|} \sum_{g \in \mathcal{G}(d)} A_g(g,t), \tag{3.12}$$

where $\mathcal{G}(d)$ is the set of groups d belongs to.

To make the re-ranking process executed by AC more intuitive, consider the dataset in Table 1.1. If we generate 5 rankings of these documents, the h-scores and the advantages for each document change as shown in Table 3.4a. We see that the h-score of d_1 is brought down in each iteration while the score for d_4 increases. Once we reach $t = 5$, the advantage for d_1 has grown so large that the overall h-score dips below that of d_2. Since the h-scores determine the rankings, this means that d_2 is now shown before d_1, meaning it receives more exposure, see also Table 3.4b. In this way, AC balances the exposure and the relevance.

The main advantage of AC over an algorithm like LM is that it creates multiple rankings for each query and that it explicitly controls for the fairness across those rankings. A weakness is that it depends on the estimated probabilities of relevance. If the ρ-values are of poor quality, AC cannot compensate for that.

Document	$h(d,t)$ $t=1$	$t=2$	...	$t=5$	$A_d(d,t-1)$ $t=1$	$t=2$	...	$t=5$
d_1	**0.9**	**0.88**	...	0.63	0.0	0.16	...	1.5
d_2	0.68	0.68	...	**0.78**	0.0	−0.063	...	−0.56
d_3	0.45	0.44	...	0.30	0.0	0.093	...	0.83
d_4	0.23	0.23	...	0.30	0.0	−0.048	...	−0.43

(a) The values of h$-score(d,q)$ (Equation (3.1)) and $A_d(d,t)$ (Equation (3.12)) over multiple rankings. The best h-score at each time t is bolded.

π_4	π_5
d_1	d_2
d_2	d_1
d_3	d_4
d_4	d_3

(b) The change in the ranking caused by the change in h$-score$ as show in Table 3.4a.

Table 3.4: Re-ranking by AC of the documents in Table 1.1.

3.5. Summary

In this chapter we described the process of selecting and validating the rankers we analyze in depth in the rest of this thesis. We selected the rankers based on a combination of ease of reproduction (open source code available, detailed companion paper, communication with authors), availability of the used tools (open source, easy to use), performance in terms of Unf or EEL, and variety in approaches.

In the end we selected two rankers: LM and AC. LM is a pairwise LtR algorithm that is used widely in IR research. Since it does not include a fairness component it will act as a baseline ranker. AC is a controller-based re-ranker that explicitly incorporates fairness.

We validated the correctness of our implementation of each ranker by comparing their performance in terms of Unf$(\mathcal{G}_e, \Pi)$ for LM and in terms of EEL$(\mathcal{G}_e, \Pi)$ for AC with the performance of the runs published in Biega et al. [5, 6].

4

Failure analysis

Evaluation is an indispensable component of IR research. Without evaluation we cannot say whether an improvement to our systems actually works, or only *feels* like it works [17]. An important step in the evaluation process is the *failure analysis* [9, 2]. The goal of the failure analysis is to determine on *which* topics rankers fail, and more importantly, *why*.

The goal of this chapter is twofold. We adapt an existing method for failure analysis described by Buckley [9] for the context of FAIR-TREC 2020. Then, we use our adapted method to conduct a failure analysis of the two rankers described in Chapter 3 and identify potential errors. In the next chapter, Chapter 5, we propose improvements to remedy these errors.

This chapter is organized as follows. In Section 4.1 we describe an existing method for failure analysis. Then, in Section 4.2 we describe the ways in which we modify the existing failure analysis method to suit the context of FAIR-TREC 2020. We conduct a failure analysis with our modified method and identify errors for LM in Section 4.4 and AC in Section 4.5. We finish the chapter with a discussion and summary in Section 4.7.

4.1. Failure analysis

Our method for failure analysis is based on the method developed by Buckley [9] during the Reliable Information Access workshop (RIA) workshop of 2004. Their method consists of four steps:

1. Create representative runs for each system.
2. Select queries to investigate.
3. Fill in a failure analysis template and detect patterns in representative runs.
4. Identify errors.

The failure analysis process is represented visually in Figure 4.1. We now describe each step in more detail.

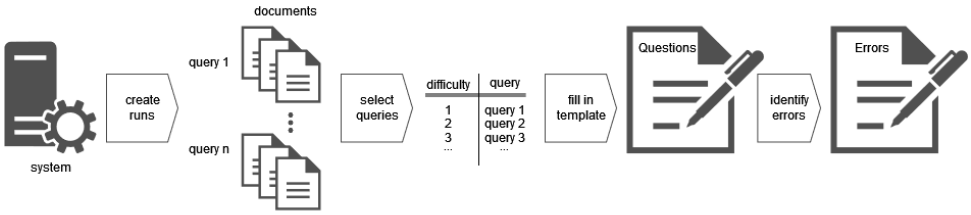


Figure 4.1: The steps in the failure analysis process.

Create representative runs A run is a set of ranked lists for each query in a validation or testing subset of the corpus. The runs are created automatically, ie. without help or intervention from a human. Runs from different systems should be made on the same dataset for comparability purposes. The purpose of the representative runs is to accurately reflect the performance of the systems so that the failure analysis can be expected to generalize to the system's performance in different circumstances.

Select queries Failure analysis is a labor intensive task; during the RIA researchers spent around 12 hours per topic. As such, it is generally not possible to investigate every query in a dataset. Buckley [9] prioritize the queries that are most likely to yield information on system-dependent failures. It is difficult to say beforehand which queries will be the most worthwhile to investigate; finding out on which queries the systems fail is part of the purpose of the failure analysis after all. To select queries Buckley [9] use a combination of the following criteria:

- High query *difficulty*. Difficult queries are those queries for which the mean system effectiveness in terms of Mean Average Precision (MAP) [47, Ch. 3] is lower than mean MAP across all queries. The idea is that difficult queries may more readily uncover system failings.
- Large variance in MAP between systems. If some systems perform well on a query and others do not, this can point in the direction of specific failures of the system that do not perform as well.
- Queries for which any individual researcher feels that it may yield interesting information on system failures.

Fill in failure analysis template During the RIA workshop, Buckley [9] developed a template to help researchers detect patterns in runs in a structured manner. This template is reproduced in Template 4.2.

The questions fall into five categories:

- Behavior on the system on the top positions in the rankings. Which relevant documents are ranked highly? Which non-relevant documents? Which relevant documents are missing from the top positions?

- Behavior of the system with respect to the query. Which query terms were important to the system?
- Other failures of the system, e.g. due to stemming or tokenization.
- Failures in assessment. Were there documents ranked as relevant when they were non-relevant and vice-versa?
- Potential improvements.

Identify errors During the process of filling in the failure analysis template the person conducting the failure analysis will likely already detect some patterns in the runs they are analysing. Indeed, Buckley [9] do not mention explaining patterns and identifying errors as a separate step. Rather, it is subsumed into the failure analysis template as the question "What should the system do to improve performance?", see Template 4.2. However, it is the implicit end goal of the failure analysis to identify the points of failure of each system, which is why we list it as a separate step.

4.2. Failure analysis of amortized rankers

Buckley [9]'s method was designed for rankers that (i) optimize rankings for effectiveness in terms of MAP, and (ii) return a single ranking per query. However, the rankers for FAIR-TREC 2020 optimize for EEL and return multiple rankings per query. Therefore, we need to modify the process slightly.

The steps of the failure analysis process remain the same as outlined in Figure 4.1, but we make changes to Steps 1 and 3, the way we select queries to inspect and the template with questions.

Step 2: Select queries Buckley [9] select queries for which the MAP is lower than the average across systems, for which there is a large variance in the MAP scores across systems, and queries that seem interesting to individual researchers. There are two changes we make.

Buckley [9] use MAP as a way to determine the difficulty of queries. However, the goal the submissions to FAIR-TREC 2020 is not to reach the highest MAP, but to reach the highest fairness in terms of EEL (Equation (2.41)). We follow the example set by [39] and adapt the measure of difficulty to our specific research context by measuring the difficulty of queries in terms of EEL.

7 systems with similar approaches were analyzed during the RIA workshop [9], which means it is possible to make meaningful statements about average performance in terms of MAP across systems and about varienace between systems. By contrast, we investigate two systems with different approaches, so it is less meaningful to say something about the average performance and performance variance. Therefore, we rely solely on the difficulty score in terms of EEL to determine which queries we investigate.

As mentioned in Section 2.3.4, EEL is computed over a specific grouping. We use two different groupings: the H-level grouping and an *individual* grouping.

The H-level grouping is as described in Section 2.2.2; authors are divided into 4 groups based on their h-index, documents are labeled according to the group of their authors. If a document's authors belong to different groups, the document's group is *Mixed*. The H-level grouping was provided as an example grouping with `corp2020train` when FAIR-TREC 2020 was running. We use it here to determine whether there is an interaction between the grouping and the performance of the rankers.

The individual grouping was not a part of FAIR-TREC 2020. Instead, it is based on the concept of a *singleton* grouping [45]. The idea is that each document is placed in its own group, ie. each document belongs to one group and each group has one document. We use the individual grouping to find errors that may otherwise be obfuscated by the interaction between a particular grouping and the rankers.

Throughout this document, if we measure EEL with the H-level group we will use the expression $\text{EEL}(\mathcal{G}_h, \Pi)$, if we measure with the singleton group we use $\text{EEL}(\mathcal{G}_i, \Pi)$.

Step 3: Fill in template As mentioned by Buckley [9], the failure analysis template is meant to function as a tool for researchers to systematically uncover failures of the system. Therefore, we tailor the template to the FAIR-TREC 2020 context. We make three changes: add new questions, remove superfluous questions, and redefine necessary concepts. We hope that in describing this process in detail we will inspire others to make similar changes. The adapted template is given in Template 4.3. We now describe each of the changes in more detail.

First, we add a question. As shown in Equation (2.43), EEL can be split into three terms. Of these, EEL-D and EEL-R represent the disparity and the relevance of a ranking respectively. Since each of these terms gives information about a different aspect of EEL it is important to look at the balance between them. For example, if EEL is high, this may be due either to low relevance or high disparity, and this informs the type of errors and improvements. Therefore we add the following question: "Is the performance of the system in terms of $\text{EEL}(\mathcal{G}_i, \Pi)$ or $\text{EEL}(\mathcal{G}_h, \Pi)$ mostly due to high disparity, low relevance, or both?".

The second change is to remove superfluous questions. The original template contained questions about query expansion, which our systems do not do. It also contained a question specifically about how to improve the systems. We answer this question in depth in Chapter 5. Lastly, the original template contained a question about Beadplot[1] observations. Beadplot is a ranking visualization tool, but it does not work with multiple rankings per query and was not found to be indispensible during the RIA workshop [9, Section 3.4] so we do not use it.

Lastly, we redefine the concept of *top positions*. The first category of questions on the original template concerns documents at the top of rankings. This makes sense for the task for the RIA workshop which was the Question Answering (QA) task, where users are most likely to look at the top documents as is implied by the choice of MRR, a measure that favors relevant documents towards the top of the ranking, as a measure for the original QA track [48].

[1] www-nlpir.nist.gov/projects/beadplot/

π_1^*	π_2^*	π_3^*	π_4^*
d_1	d_1	d_2	d_2
d_2	d_2	d_1	d_1
d_3	d_4	d_3	d_4
d_4	d_3	d_4	d_3

Table 4.1: The top positions of the rankings of the dataset in Table 1.1a.

The task for FAIR-TREC 2020 is an academic search task and the EEL measure is built on the ERR measure which also favors good results towards the top of the ranking. Therefore it still makes sense to look at documents at the top of the ranking. However, we redefine what exactly it means for a document to be at the top of a ranking. Each ranker returns 150 rankings per query; it is infeasible to inspect each ranking for each query and detect patterns in the occurrence of documents in the top positions that way, especially since the 150 rankings is an arbitrary choice and it could also be 1000. Additionally, the cutoff point for a top document is given [9].

Instead we aggregate the occurrence of documents in top positions as follows. Recall from Section 2.3.4 that EEL is minimized by a policy that (i) puts all relevant documents before all non-relevant documents, and (ii) does so at random [20]. The positions of interest then are all those positions in the ranking where relevant documents *should* appear. Since no relevant document should occur higher than any other relevant document more often, the number of top documents per query equals the number of relevant documents for that query. More formally, given a query q with documents $\mathcal{D}_q$, we define as the top positions those first k positions such that $k = |\{d|d \in \mathcal{D}_1 \text{and} rel(d) = 1\}|$, where $rel(d)$ is the relevance grade of the document d.

We illustrate this concept with the toy example defined in Table 1.1. Table 1.1b shows all of the optimal rankings of the dataset in Table 1.1a. We see that in each of the optimal rankings, d_1 and d_2 are in the first two positions. The top positions for the rankings for dummy query q_{dummy} are $j = 1$ and $j = 2$ as visualized in Table 4.1.

Methodological limitations We note two methodological limitations of our approach. Firstly, for AC we make use of estimated relevance probabilities provided to us by its authors. It is unclear what portion of the training data was used to predict these values, so it is likely that there is some information leakage between our training and validation split for AC in particular.

Secondly, early on in the failure analysis we made a methodological error. We performed the failure analysis on the true test set `corp2020test` rather than on `corp2020train`, which helped us detect certain patterns in the data. Since these patterns also show up in the training data however we do include them in our analysis.

4.3. Failure analysis

We conduct a failure analysis of the two rankers described in Chapter 3, LambdaMART and Advantage Controller. We go through each of the steps in Section 4.1.

Step 1: Create representative runs We create representative runs of LM and AC on `corp2020val_split`, the validation dataset.

We create one run for LM, `lm`. We use our implementation as described in Section 3.3. We train LM on `corp2020train` using the same features and hyperparameter settings as when validating our implementation as given in Table 5.2 and Table 3.3.

For AC we have the choice of four runs: one for each combination of set of estimated probabilities of relevance and value for θ. As noted by Buckley [9], failure analysis is a time-intensive activity. In addition, during preliminary investigations we noticed that investigating each combination of relevance sets and θ-values was more confusing than informative because it is difficult to determine whether the observed effects are due to failures of the system or expected differences due to different parameters. For these reasons we focus on the run that performed best in the original submission by Kletti and Renders [27]: `rel_set_A` and $\theta = 0.9$.

Step 2: Select queries We compute the EEL($\mathcal{G}_i, \Pi$) and EEL($\mathcal{G}_h, \Pi$) for each query for each run and select the 10 most difficult queries, ie. the queries with the highest EEL($\mathcal{G}_i, \Pi$) or EEL($\mathcal{G}_h, \Pi$).

Step 3: Fill in failure analysis template For each configuration we answer the questions in our modified template. For sake of brevity we do not reproduce the answers here, but instead refer to specific insights gained during the process of filling in the template when necessary.

Step 4: Identify causes of failure Lastly, we identify causes of failure. The causes and the reasoning behind them are described in the following two sections, Section 4.4 and Section 4.5. For the sake of completeness and to prevent other from doing double work, we also mention a number of things we thought in advance might cause failures but for which we were unable to find evidence.

	$\mathcal{G}_i$			$\mathcal{G}_h$		
Run	EEL	EEL-D	EEL-R	EEL	EEL-D	EEL-R
LM	1.378	1.239	0.208	0.762	2.154	1.243
AC	0.515	0.238	0.139	0.498	1.653	1.124

Table 4.2: Mean EEL($\mathcal{G}_i, \Pi$) and EEL($\mathcal{G}_h, \Pi$) achieved by LM and AC on dataset `corp2020val_split`.

4.4. LambdaMart

We see in Table 4.2 that LM performs roughly 2.5 times worse than AC in terms of EEL($\mathcal{G}_i, \Pi$) and almost 1.5 times worse in terms of EEL($\mathcal{G}_h, \Pi$). We identify a number of causes for the high value of EEL. We group the causes into three groups: (i) causes affecting the *disparity* EEL-D, (ii) causes affecting the *relevance* EEL-R, and (iii) causes related to the design of the EEL measure overall. The causes are summarized in Table 4.3a. We now explain each cause in more detail.

Ranker	Cause	Description
LM	`CLM_single`	LM produces a single ranking for each query, leading to a high disparity.
	`CLM_size`	The dataset is small, likely causing overfitting.
	`CLM_noisy`	The relevance labels for the documents are not always accurate, making it harder for LM to be trained properly.
	`CLM_ambiguous`	Many queries are short or can be interpreted in multiple ways.
	`CLM_features`	Certain features affect the performance of the ranker negatively.
	`CLM_measure`	Optimizing for NDCG may negatively affect the relevance.
AC	`CAC_no_author`	Some non-relevant documents do not have author information, causing them to be over-represented in the rankings.
	`CAC_rel`	Some relevant documents have low estimated relevances, causing them to be under-represented in the rankings.

(a) Causes of failures of LM and AC. Improvements addressing some of these causes are listed in Table 5.1.

	Cause	Description
	`C_complex`	Unable to detect complex interactions with the modified failure analysis method.
	`C_eel_constant`	Due to the EEL-C-component, EEL is a sub-optimal difficulty measure.
	`C_eel_grouping`	Evaluating rankings with the H-level grouping did not yield useful information.

(b) Flaws in the modified failure analysis method.

Table 4.3: Causes of failure for LM and AC as well as flaws with the modified failure analysis.

4.4.1. Causes affecting EEL-D

We find that the main reason that LM performs worse than AC is because of a high disparity value EEL-D($\mathcal{G}_i, \Pi$) or EEL-D($\mathcal{G}_h, \Pi$). LM's EEL-D($\mathcal{G}_i, \Pi$) is 5 times *worse* than the worst EEL-D($\mathcal{G}_i, \Pi$) for AC; for EEL-D($\mathcal{G}_h, \Pi$) the difference is a factor 1.2. We know the issue is not the relevance EEL-R($\mathcal{G}_i, \Pi$) or EEL-R($\mathcal{G}_h, \Pi$): the EEL-R($\mathcal{G}_i, \Pi$) of LM is 1.4 higher than that of the next best value for AC; for EEL-R($\mathcal{G}_h, \Pi$) the difference is smaller with 1.1 times the value.

Single ranking By design LM returns a single ranking for each query. However, the goal of FAIR-TREC 2020 is to create *amortized* rankers. To meet this requirement, the single ranking created by LM is repeated 150 times. As we know, a single ranking cannot be perfectly fair [4], so there is by definition some disparity in the ranking, which then is exacerbated by the ranking being repeated so many times.

Recall the example we showed in Table 2.10. Here we saw clearly that repeating the same ranking over and over leads to a discrepancy in exposure not only between relevant and non-relevant documents, but between equally relevant documents as well.

We refer to this cause as `CLM_single`.

4.4.2. Causes affecting EEL-R

As we see in Table 4.2, the value of both EEL-R($\mathcal{G}_i, \Pi$) as well as EEL-R($\mathcal{G}_h, \Pi$) is better for LM than for the next best option for AC. However, it may be possible to boost EEL-R even further. We remind the reader at this point that EEL-R is *not* an effectiveness measure in the same sense as e.g. NDCG or ERR. It does not measure the ability of the system to retrieve the most useful documents for the user. Instead, EEL-R measure how much of the exposure is given to relevant documents.

Even so, for relevant documents to receive more exposure, they have to be ranked more highly, which is more likely to happen if LM is more effective. Therefore, we identify some errors that may cause a lower effectiveness of the system.

Noisy data To answer questions 1-3 in Template 4.3 we inspected every relevant and non-relevant document in the top of the rankings, as well as every relevant document in the bottom of the rankings. A recurring observation is that many documents that are marked as relevant we would judge to be non-relevant and vice-versa. This makes it difficult for LM to be trained well.

The noisiness of the dataset is most likely due to the process through which it was created. As explained in Section 2.2.1, the relevance labels are derived from logged queries and clicks. As the organizers themselves note, click data is biased because users may click on things that are not relevant but grab their attention for another reason [5].

Another factor is that the queries were mined from a short period, February 14 2020 to April 27 2020. This was around the start of the COVID-19 pandemic [28]. It is likely that the intense need for information on this topic influenced users' search and click behavior. We see for example that a number of the queries pertaining

to specific aspects COVID-19 have general documents on COVID-19 marked as relevant as well.

Ambiguous queries When answering questions 1-3 in Template 4.3 we also found that there are many queries that are ambiguous. For example, some queries are so short that they can apply to many documents, e.g. "beauty", or they are open for multiple interpretations, e.g. "dover beach" could be the poem by Mathhew Arnold or a place. Query ambiguity is a known challenge for IR systems [30, Ch. 9]. We refer to this cause as `CLM_ambiguous`.

Choice of features When answering question 3 in Template 4.3 we noticed that many of the relevant documents that were not ranked in the top positions had low values for either the `inCitations` or `outCitations` field. `inCitations` is the number of documents that cite the document, and `outCitations` is the number of other documents this document cites. Intuitively, it makes sense that a document with fewer incoming citations could seem less relevant, but it is possible that the `inCitations` and `outCitations` fields are weighed too heavily. We refer to this error as `CLM_features`.

Small dataset The causes `CLM_noisy` and `CLM_ambiguous` are likely exacerbated by the small size of the dataset; `corp2020train` consists of 200 queries with a total of 4000 documents. A known risk for small datasets is that the model will underfit, leading to low predictive power. Especially in combination with noisy labels, the size of the dataset likely makes it difficult for LM to be trained properly. We refer to this cause as `CLM_size`.

Optimizing for NDCG The goal of training LM is to optimize for a particular effectiveness measure. As explained in Chapter 3, this is done by scaling the loss at each training step by the difference in effectiveness resulting from swapping two items [11]. The standard choice of effectiveness measure is NDCG, but it has been shown that LM can also optimize for, inter alia, MAP, MRR [22].

In filling in Template 4.3 we were not able to find evidence that optimizing for NDCG has a detrimental effect on EEL. However, previous literature shows that it *can* be more effective to optimize for the same measure you evaluate the system with [22]. Therefore, it is possible that the fact we optimize for NDCG is a source of failure. We refer to this cause as `CLM_measure`.

4.5. Advantage Controller

As we see in Table 4.2, AC performs better than LM both when using an individual grouping and when using an h-level grouping. The improved performance is mostly due to a lower disparity: the EEL-D$(\mathcal{G}_i, \Pi)$ is 5 times lower for AC than for LM; for EEL-D$(\mathcal{G}_h, \Pi)$ the difference is a factor 1.3. On the other hand, both the EEL-R$(\mathcal{G}_i, \Pi)$ and EEL-R$(\mathcal{G}_h, \Pi)$ are higher than for LM.

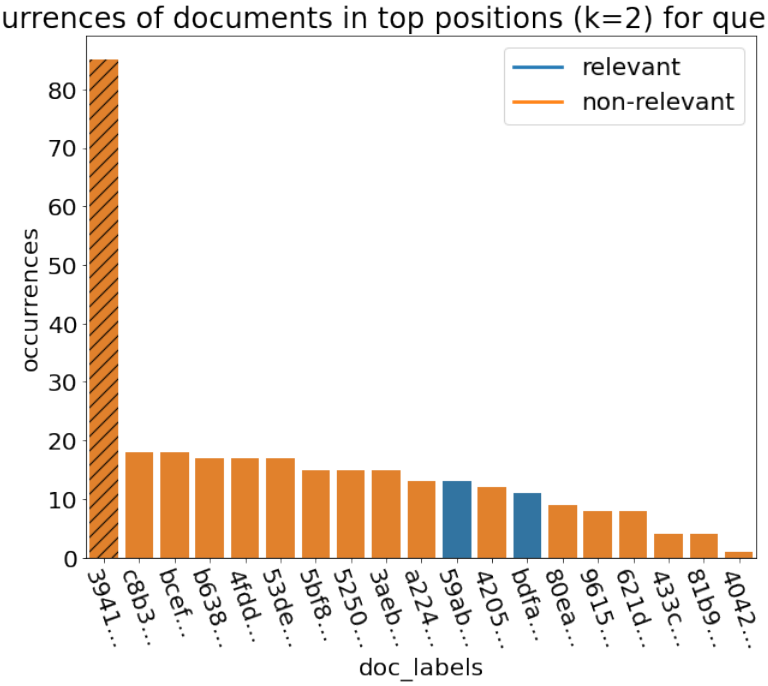


Figure 4.4: The number of times each document appears in a top position for query 31412 when ranked by baseline AC. Hatched documents do not have an author.

It was challenging to identify causes for AC with the failure analysis method we use. In the end, we identified a two causes cause. In this section we describe the causes we identified. In the next section we go deeper into the limitations of our failure analysis method .

Susceptibility to missing author information When filling in questions 1-3 in Template 4.3 we noticed that some non-relevant documents occurred in a top position noticeably more often than other documents. Figure 4.4 shows one such example: we see that the document that occurs the most often (i) is non-relevant and (ii) does not have an author. The authorless document also occurs much more often than any other document, almost 5 times more often than the next most frequent document.

This behavior only occurs for non-relevant documents. As we see in Table 4.4, 8.3% of non-relevant documents in the top positions of the rankings for the most difficult queries (according to EEL$(\mathcal{G}_i, \Pi)$) do not have an author, while this number is 0% for relevant documents in either top or bottom positions. The same pattern occurs when we measure difficulty in EEL$(\mathcal{G}_h, \Pi)$.

This effect is due to the way AC introduces fairness into a sequence of rankings. Recall the re-scoring function Equation (3.1) for a document d for query q at time

Measurement	Relevant/ non-relevant	Top/ bottom	Difficulty measure EEL($\mathcal{G}_h, \Pi$)	EEL($\mathcal{G}_i, \Pi$)
Avg. %	Relevant	Top	17	14
Avg. # of authors	Relevant	Top	2.3	1.9
	Non-relevant	Top	2.8	3.2
	Relevant	Bottom	2.5	4.9
Avg. # of documents/author	Relevant	Top	1.1	1.0
	Non-relevant	Top	1.0	1.1
	Relevant	Bottom	1.2	1.0
# without author	Relevant	Top	0	0
	Non-relevant	Top	4	5
	Relevant	Bottom	0	0
% without author	Relevant	Top	0.0	0.0
	Non-relevant	Top	4.5	8.3
	Relevant	Bottom	0.0	0.0
Avg. estimated probability of relevance	Relevant	Top	0.18	0.20
	Non-relevant	Top	0.16	0.19
	Relevant	Bottom	0.18	0.17
Avg. relative position based on estimated probability of relevance	Relevant	Top	10	3.6
	Non-relevant	Top	14	5.4
	Relevant	Bottom	9.1	7.4
Avg. difference between estimated probability of relevance and true relevance	Relevant	Top	0.50	0.56
	Non-relevant	Top	0.020	0.029
	Relevant	Bottom	0.35	0.58

Table 4.4: Various statistics over the rankings for the 10 most difficult queries according to baseline AC. The estimated probabilities of relevance are taken from `rel_set_A`. Relative position is computed as in Equation (4.2).

t:

$$h(d,t,q) = \theta\rho(d,q) - (1-\theta)A(d,t), \tag{4.1}$$

where $\rho(d,q)$ is the estimated relevance probability of d for q and $A(d,t)$ is the advantage of d at time t as defined in Equation (3.12).

The advantage indicates whether a document has received more or less of its fair share of exposure at time t; h boosts documents that thus far have received too little exposure and hinders documents that have received too much. Each document's advantage at time t $A(d,t)$ is defined as the arithmetic mean of the advantage of its authors. If a document has no authors, $A(d,t) = 0 \forall t$.

Because the advantage is a core element of AC's ranking strategy and documents without an author are treated differently, we can expect AC to be sensitive to missing

authors. Documents without an author cannot be advantaged, but crucially, they are also never *hindered*, resulting in the observed spiking behavior.

The over-representation of authorless documents is detrimental to the EEL value both in terms of relevance and disparity. The disparity grows because a single document receives most of the attention, and the relevance decreases because most of the authorless documents in the validation set in reality are non-relevant.

We refer to the failure cause of missing author information as `CAC_no_author`.

Low estimated relevances We see a number of of patterns specific to the relevant documents in bottom positions. As we see in Table 4.4, when we measure difficulty in EEL$(\mathcal{G}_i, \Pi)$ the non-relevant documents in bottom positions on average have an estimated relevance value of 0.17 versus a value of 0.20 and 0.18 for relevant and non-relevant documents in top positions respectively.

Additionally, if we were to rank the documents by predicted relevances alone, the relevant documents in bottom-positions would end up at higher (and therefore worse) relative positions. We compute the relative position of a document in ranking π as

$$\text{relative position}(d) = \frac{\pi(d)}{|\pi|}. \tag{4.2}$$

For the relevant documents in bottom positions, the mean relative position(d) is 7.4 versus 3.6 and 5.4 for relevant and non-relevant documents in top positions respectively.

Lastly, we see that the non-relevant documents in bottom positions have a higher difference with the true relevance values than the relevant documents in top positions. In other words, the predicted relevances are less accurate for the relevant documents in bottom positions.

The low and inaccurate estimated relevances of the relevant documents in bottom positions in particular are detrimental to the EEL. The overall relevance goes down because less exposure is on relevant documents, and the disparity goes up because not all relevant documents get equal exposure. We refer to this cause of failure as `CAC_rel`.

4.6. Limitations of the modified failure analysis

The modified failure analysis method we use in this chapter was designed for the purposes of this thesis and has not been used before. As such, during the the process of analyzing LM and AC we ran into a number of limitations in our chosen method. These are not failure causes in the same sense as the ones we identified for the rankers but they *do* affect the outcome of the failure analysis. We detected three flaws: (i) limited ability to detect complex interactions, (ii) limited utility of EEL as a measure of difficulty, and (iii) superfluous questions. The flaws are summarized in Table 4.3b.

4.6.1. Detecting complex interactions

It was more challenging to identify causes for failures for AC than for LM. From the point of view of fairness, LM is a simpler ranker since it does not try to take into account fairness.

AC on the other hand has numerous components that affect the final score. There is the scoring function h, the way of computing advantage for documents from the advantage of their authors, the way of computing the expected exposure terms for the advantage, and the input relevance sets. This multitude of factors made it difficult to distinguish which observed effect was due to which cause.

One of the challenges was to determine to which extent the choice of grouping by authors in computing advantage affects the overall score. For example, we expected to see some difference in the number of authors of relevant and non-relevant documents in top and bottom positions. We also expected to see a difference in the number of documents each author had contributed to the ranking. We see in Table 4.4 that the relevant documents in the bottom positions indeed have more authors than the relevant and non-relevant documents in top positions, 4.9 on average versus 1.9 and 3.2 respectively. However, those authors then all on average contribute around 1.0 document to the ranking, meaning that even if a document has many authors they are not affected by other documents in the ranking.

We were unable to determine the cause of this kind of interaction effects by using our modified failure analysis because it depends not only on the frequency of occurrence in top or bottom positions of certain documents and their characteristics, but also on the behavior of all other documents in the ranking.

4.6.2. EEL as a measure of difficulty

During the failure analysis we ran into two issues that were due to our choice of EEL as the measure of difficulty: the sub-measure EEL-C and measuring over multiple groupings.

Constant component of EEL Recall that EEL can be split into three components (see Equation (2.43)):

$$\text{EEL} = \text{EEL-D} - \text{EEL-R} + \text{EEL-C}. \tag{4.3}$$

To answer question 5 in Template 4.3 we plotted EEL versus EEL-D − EEL-R for queries of decreasing difficulty for both LM and AC and for $\text{EEL}(\mathcal{G}_i, \Pi)$ and $\text{EEL}(\mathcal{G}_h, \Pi)$. The plots are shown in Figure 4.5.

As we see, the decrease in EEL is not directly related to the decrease in EEL-D − EEL-R. This points at the influence of the third component, EEL-C.

EEL-C depends completely on $\mathcal{E}_g^*$ and as such is the same for queries that have the same number of relevant and non-relevant documents. However, as a result its value is inherently higher or lower for different queries. As a result, if we use EEL as a measure for difficulty, some queries will *seem* inherently more or less difficult than other queries.

As long as we evaluate rankers on the same sets of queries this is not necessarily an issue since in that case the EEL-C components add the same value to

the evaluation of each ranker. However, it means that EEL is flawed as a difficulty measure, since we cannot compare the difficulty of two queries without accounting for the EEL-C component. We refer to this flaw as `C_eel_constant`.

Evaluation with different groupings We computed the EEL of the rankings with two different groupings, $\mathcal{G}_{ind}$ which places each document in its own group, and $\mathcal{G}_h$ which groups documents by the H-index of authors.

The goal of using the H-level grouping in particular was to detect interactions between the behavior of the rankers and the chosen grouping. We used this particular grouping because it was available to participants of FAIR-TREC2020 when it was ongoing. However, we found that it was difficult to pinpoint the reactions that occurred with our failure analysis method. We do detect *some* effect: As we see in Table 4.2, the EEL$(\mathcal{G}_h, \Pi)$ is lower than the EEL$(\mathcal{G}_i, \Pi)$ for both LM and AC, particularly due to a higher relevance EEL-R. However, we were unable to determine what characteristic of the H-level grouping is responsible for this effect. Additionally, since we use a single grouping, it is unclear whether the observed effect would generalize to other groupings such as the Econ-level grouping provided with `corp2020test`.

The latter point points at a limitation of EEL as a measure for fairness overall. The goal of EEL is to measure fairness with respect to arbitrary groupings. However, computing EEL requires *a* group definition, meaning any conclusions drawn from values of EEL will be applicable to that grouping only. It is of yet unclear whether a good performance with respect to one grouping generalizes to a good performance on *arbitrary* groupings.

We refer to this flaw as `C_eel_grouping`.

4.7. Summary

In this chapter we defined a modified failure analysis method based on the method defined by Buckley [9]. We then performed a failure analysis of two rankers, the baseline configuration of LM as described in Section 3.3 and AC with `rel_set_A` and $\theta = 0.9$. All identified causes are summarized in Table 4.3a.

For LM, we found the following causes of failure:

- `CLM_single`: LM generates a single ranking per query, resulting in high disparity EEL-D.
- `CLM_noisy`, `CLM_size`, and `CLM_ambiguous`: The training set `corp2020train` is small and the relevance labels are noisy, making it hard for LM to be trained properly. Additionally, many of the queries in `corp2020train` are ambiguous, making it hard for LM to distinguish between relevant and non-relevant documents.
- `CLM_features`: Some features impact the the score of LM in unexpected or undesirable ways.
- `CLM_measure`: LM is optimized for NDCG but EEL is based on ERR.

For AC we found the following causes of failure:

- `CAC_no_author`: AC is susceptible to missing author information. Documents without authors are disproportionately advantaged.
- `CAC_rel`: Some relevant documents have disproportionately low estimated relevance values ρ, hurting their position in the final ranking.

Finally, we identified a number of limitations of our chosen failure analysis method. These are listed in Table 4.3b. The limitations are:

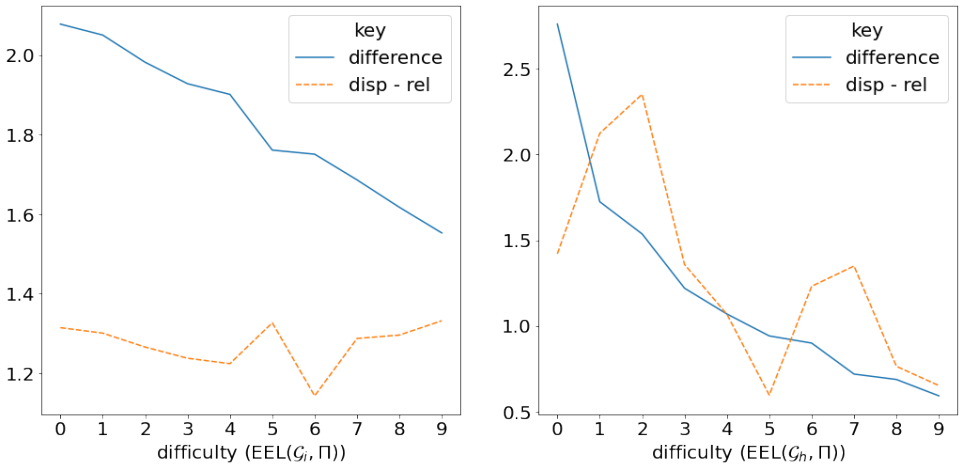
- `C_complex`: Our failure analysis method was insufficient to uncover complex causes of failure.
- `C_eel_constant` and `C_eel_grouping`: EEL is flawed as a measure of query difficulty due to the influence of sub-measure EEL-C and the influence of the chosen grouping.

1. Behavior on top non-relevant documents [Why were the top non-relevant documents retrieved?]
2. Behavior on unretrieved relevant documents [Why weren't these relevant document retrieved within the top 1000?]
3. Beadplot observations [How does the ranking (especially among the top 50 documents) of this system compare to all other systems?]
4. Base Query observations [What did the system think were the important terms of the original query, and were they good?]
5. Expanded Query observations [If the system expanded the query (4 out of 6 systems did), what were the important terms of the expansion, and were they helpful?]
6. Blunders of system [What obvious mistakes did the system make that it could have easily avoided? Examples might be bad stemming of words or bad handling of hyphenation]
7. Other features of note [Anything else.]
8. What should system to do improve performance? [The individual's conclusion as to why the system did not retrieve well, and recommendations as to what would have made a better retrieval.]
9. What added information would help performance? How can system get that information? [Is there implicit information in the query, that a human would understand but the system didn't? Examples might be world knowledge (like Germany is part of Europe).]
10. Assessing agreement (were there major issues? was relevance determined by "Desc"?) [The NIST assessor who originally judged relevance of documents might have a different idea of what was relevant than the text of the description indicates or than the workshop participant thinks should be relevant. It also may be unclear where and why the NIST assessor drew the line between marginally relevant and non-relevant documents.]

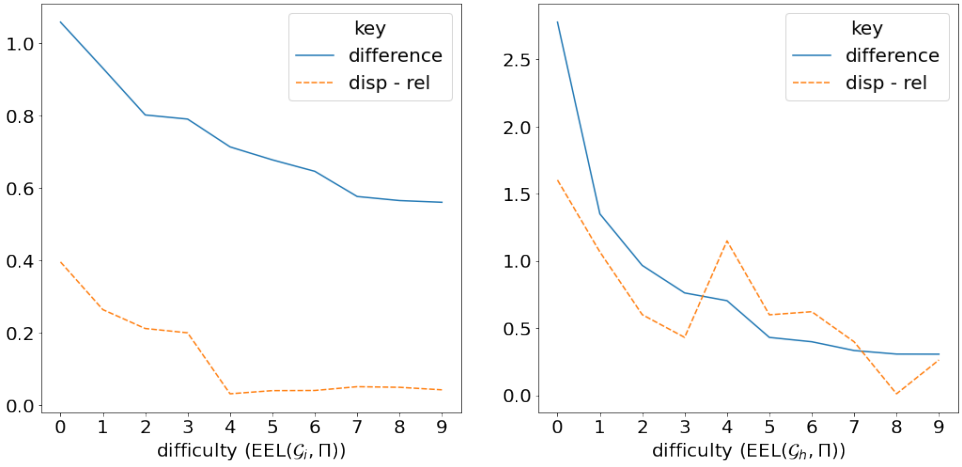
Template 4.2: Topic Failure Analysis Template for Individual System. Caption and template from Buckley [9, Figure 1].

1. How often were top positions occupied by relevant documents? Can these documents be categorized and distinguished from other documents?
2. How often were top positions occupied by non-relevant documents? Can these documents be categorized and distinguished from other documents?
3. Which relevant documents appear rarely or not at all in the top positions? Why did these documents not appear in the top positions more often?
4. What did the system think were the important terms of the original query, and were they good?
5. Is the performance of the system in terms of EEL$(\mathcal{G}_i, \Pi)$ or EEL$(\mathcal{G}_h, \Pi)$ mostly due to high disparity, low relevance, or both?
6. What obvious mistakes did the system make that it could have easily avoided? (E.g. stemming or hyphenation issues.)
7. Other features of note
8. What information is there in the query that the system does not understand (e.g.: Germany is part of Europe)? How can the system get that information?
9. Assessing agreement. Do the relevance labels seem accurate?

Template 4.3: Topic failure analysis template for a single amortized ranking system.



(a) Ranker: LM.



(b) Ranker: AC.

Figure 4.5: EEL versus EEL-D – EEL-R for queries in `corp2020val_split`. Queries are ordered by decreasing difficulty.

5

Improvements

We propose improvements to remedy some of the causes we identified in Chapter 4. We implement the proposed improvements and assess whether they remedy the causes. All proposed improvements are listed in Table 5.1.

The structure of this chapter is as follows. In Section 5.1 we describe our approach to proposing, implementing, and evaluating the improvements. In Sections 5.2 and 5.3 we describe the improvements in detail and assess whether they remedy the causes. We close the chapter with a summary in Section 5.4.

Ranker	Improvement	Variations	Description	Addressed cause
LM	ILM_post	none ac rfre	Apply post-processing on the rankings produced by LM.	CLM_single
	ILM_fsm	none mpt msd	Select the best features for LM with a feature selection algorithm.	CLM_features
	ILM_measure	ndcg err	Optimize LM for a different effectiveness measure.	CLM_measure
AC	IAC_dummy	none ind one	Add dummy authors to documents without authors.	CAC_no_author
	IAC_hfunc	linear min max	Equalize the amount of advantage for each document with a different scoring function.	CAC_rel
	IAC_rels	rel_set_A rel_set_lm	Use a more accurate set of estimated probabilities of relevance.	CAC_rel

Table 5.1: The proposed improvements for LM and AC. The causes are listed in Table 4.3a.

5.1. Approach

To improve each ranker we perform the same actions. We select causes to improve, propose and implement propose and implement improvements, create runs on and evaluate performance on corp2020val_split, select improvements that work on corp2020val_split, and create runs and evaluate performance on corp2020test.

Causes and improvements In the interest of time we are unable to investigate solutions to every cause in Table 4.3a. Additionally, sometimes we lack certain resources that would enable us to propose a solution. For causes where we are unable to propose an improvement we explain why.

We propose a solution for each of the selected causes. All solutions and the causes they target are listed in Table 5.1. Throughout this chapter we provide implementation details where necessary.

Runs Much like in Step 1 of the failure analysis, we create representative runs of each improved ranker. The runs are generated automatically, that is, without intervention from our side after starting the training and ranking process. The runs are assumed to reflect the performance of the system at large.

To determine which improvements perform significantly better than the baselines we create the runs on the same data that we used for the failure analysis, namely `corp2020val_split`. To assess whether improvements generalize well we use the test data `corp2020test`.

We compare the performance of each improvement relative to that of the unaltered ranker upon which it is meant to improve. We refer to the runs with the unaltered rankers as the *baseline* runs. These runs are the same as described in Section 4.3 with the same hyperparameter settings and random seeds as described in Table 3.3. Table 4.2 shows each baseline run and the performance it achieved on `corp2020val_split`.

Evaluating the performance We measure the performance of the runs with EEL. We use three different groupings: the singleton document grouping, the H-level grouping, and the Econ-level grouping.

As mentioned in Section 4.2, we use the individual-level grouping EEL($\mathcal{G}_i, \Pi$) because it allows us to more directly compare the results on `corp2020val_split` and `corp2020test`, since there is no interference from the grouping. In addition, we use the H-level grouping with `corp2020val_split` and EEL($\mathcal{G}_e, \Pi$) with `corp2020test` since the H-level grouping and the Econ-level grouping were provided with the `corp2020train` and `corp2020test` respectively. While using different groupings for the training and testing data makes it more difficult to draw a direct comparison it may provide insight in the influence of different groupings on performance.

We investigate whether the difference in performance is significant with the procedure given by Sakai [38]. For the sake of legibility all statistical test results are collected in Appendix B.

Sakai [38] describes two procedures: (i) a procedure for comparing two systems, and (ii) a procedure for comparing more than two systems. Under *system* we understand any configuration of a ranker, be it a completely different approach, e.g. LM versus AC, or a difference within the same ranker, e.g. a different h-score function within AC.

We assess the significance of the difference in performance between *two* systems with a two-sided paired t-test [43]. The samples for the paired test are the scores on each topic achieved by each system. We use the `scipy.stats` implementation to compute the test statistic[1].

We use a *two*-sided rather than a *one*-sided comparison because we want to know whether the performance of the improved system is *different* from that of the baseline system rather than whether the improved system is *better* than the baseline system. This is to cover the case that the improved system actually performs

[1] `docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_rel.html`.

worse than the baseline, which we would not be able to detect with a one-sided t-test.

We assess the significance of the difference in performance between *three or more* systems with a two-way ANOVA. In particular, we use the `statsmodels` implementation[2]. The factors for the ANOVA are the `system` and the query identifier `qid`. The `system` factor in this cases encompasses both different systems, e.g. LM or AC, and configurations of the same system, e.g. LM with different post-processing methods applied. Since we assess the influence of different improvements, within each ANOVA we rename the `system` to a more descriptive name, e.g. `post` when we assess the influence of different post-processing methods.

It is possible that some topics are inherently more difficult than others. By treating `qid` as a factor we can separate the variance due to query difficulty from that due to the systems themselves.

Since we fixed all random seeds to 0, we have a single measurement per system/`qid`-pair. As such, we perform ANOVA without replication, which means we cannot say anything about the interaction between `qid` and system [38].

A statistically significant difference does not necessarily imply that there is a *practical* difference in performance [13]. To give insight into the magnitude of the observed effects we must also provide the effect size and 95% Confidence Interval (CI) Sakai [38]. For ANOVA, we report the effect size in terms of (partial) ω^2 [34, Table 10]:

$$\omega^2 = \frac{df_{effect}\left(MS_{effect} - MS_{error}\right)}{SS_{total}}$$

and

$$\omega^2_{partial} = \frac{df_{effect}\left(MS_{effect} - MS_{error}\right)}{df_{effect}MS_{effect} + (N - df_{effect}MS_{error})}.$$

Here df is degrees of freedom, MS is the mean squared error, SS is the sum of squares, and N is the total number of observations.

For the t-test, we report the effect size as in Sakai [38, Eq. 2]:

$$ES_t = \bar{d}/\sqrt{V},$$

where $\bar{d}$ is the mean of the sample and V is the unbiased variance of the sample. The mean is given by

$$\bar{d} = \sum_{j=1}^{n} d_j/n$$

and

[2] statsmodels.org/dev/generated/statsmodels.stats.anova.anova_lm.html

$$V = \frac{1}{n-1}\sum_{j=1}^{n}\left(d_j - \bar{d}\right)^2$$

where d_j is the difference between the score on the jth topic achieved by the first and the second configuration and n is the total number of topics.

For ANOVA we compute the CI as

$$CI = [\bar{d}_i - ME_a, \bar{d}_i + ME_a] \tag{5.1}$$

where $\bar{d}_i$ is the mean of the ith group and

$$ME_a = t(df_{error}; 0.05)\sqrt{MS_{error}/n_{effect}}. \tag{5.2}$$

For the t-test we compute it as

$$CI = [\bar{d} - ME_t, \bar{d} + ME_t] \tag{5.3}$$

where $ME_t = t(n-1; 0.05)\sqrt{V/n}$ and $t(df; \alpha)$ is the Student t-distribution.

Because we have a single measurement per sytem/`qid`-combination, our experiments violate the assumptions of homoscedacity and normality underlying the t-test and ANOVA. However, both tests have been shown to be robust to these violations [38, 46], and the t-test in particular has been shown to be more appropriate than non-parameteric alternatives for the task of comparing performance [46].

We find the significant differences between levels within a factor with a Tukey HSD test[3]. We report the effect size in terms of Cohen's d [15, ch. 2]:

$$d_c = \frac{|\bar{d}_i - \bar{d}_j|}{\sqrt{MS_{error}}}.$$

Again, $\bar{d}_i$ and $\bar{d}_j$ are the means of the ith and the jth groups, and MS_{error} are the mean squares of the error terms. It should be noted that we do not perform Tukey tests for the `qid` factor even when it is significant, because knowing which topics yields significantly better or worse performance does not yield actionable information in this scenario.

5.2. Improvements to LambdaMart

In this section we propose and evaluate improvements for the causes of failure for LM listed in Table 4.3a. All improvements are listed in Table 5.1. We now discuss each improvement in more detail.

[3] statsmodels.org/dev/generated/statsmodels.stats.multicomp.pairwise_tukeyhsd.html

5.2.1. Improving disparity with post-processing

We identified one cause affecting the disparity EEL-D, `CLM_single`. In short: because LM returns a single ranking for each query rather than a series of different rankings, any unfairness that exists in the single ranking accumulates over time.

There are two ways in which a ranker could yield different rankings in response to a single query. the first way is if the ranker is a *stochastic* ranker. Stochastic rankers learn a distribution over the documents and sample from this distribution to create a ranking [8]. Since the ranking is sampled it is different each time. A stochastic version of LM has been proposed by Wang et al. [49]. However, no implementation of this method is available for the library we used for LM, `pyltr`. Therefore, in the interest of time, we do not investigate this further here.

The second way to generate multiple rankings with LM is by using *post-processing*. Post-processing is one of the three ways of introducing fairness to existing algorithms identified by i.a. Zehlike, Yang, and Stoyanovich [51]. Post-processing methods are all those methods that *re-rank* the output from a ranker to introduce more fairness. By re-ranking the output from LM differently 150 times we can create a fairer sequence of rankings.

Post-processing methods are usually implemented as separate modules that take the output from one ranker as input and generate new rankings from it. This makes it easy to swap out and test different post-processing modules.

We refer to this improvement as `ILM_post`. We implement two post-processors, each of which we describe in more detail now.

Advantage Controller The first post-processor at this point should be familiar to the reader: the Advantage Controller. AC takes in a set of estimated probabilities of relevance and based on those generates as many different rankings as needed while balancing exposure.

To use AC as a post-processor we only need to change the estimated probability of relevance set so that it is generated by LM. LM returns both positive and negative scores for documents, but AC needs scores that are in the $[0, 1]$-range because they are used as parameters for the Poisson-Binomial distribution (see Equation (3.2)).

We use the baseline version of AC as the re-ranker. We use h as defined in Equation (3.1) and set $A_d(d, t)$ to 0 if the author information for d is missing. It is hard to say whether we should use $\theta = 0.9$ or $\theta = 0.99$ as a baseline version, since Kletti and Renders [27] evaluate both and neither option is necessarily better than the other. However, $\theta = 0.9$ gives slightly more weight to the component of h balancing the disparity. The goal is to improve the disparity of LM so therefore we set $\theta = 0.99$.

Robust Fair Ranking with Expected Exposure Loss The second post-processor is based on the Multi-grouping Robust Fair Ranking (MRFR) algorithm [45]. The goal of MRFR is to achieve fairness across *multiple* rankings for a single query with respect to an *unknown* evaluation grouping, much like is the case for the FAIR-TREC 2020 benchmark.

In words, the approach of MRFR is to: (i) compute a pre-ordering score and order the documents, (ii) create candidate rankings by permuting the top K documents of the pre-ordered list of documents, (iii) select the best candidate ranking according to a second scoring function. We summarize the approach in Algorithm 1. In the pseudocode,

$$\phi(d,t) = \rho_d - \beta\ \delta(d,t) \tag{5.4}$$

with

$$\delta(d,t) = \sum_{g \in \mathcal{G}} [\mathrm{E}(g,t) - \mathrm{R}(g,t)]\,, \tag{5.5}$$

and

$$\psi(\pi^c,t) = \mathrm{U}(\Pi_{<t} \cup \pi^c) - \lambda \mathrm{Unf}(\Pi_{<t} \cup \pi^c). \tag{5.6}$$

Here $\rho(d)$ is the (predicted) relevance of document d and E, R, U and Unf are exposure, relevance, utility and unfairness respectively as used in FAIR-TREC 2019 [5]. $\Pi_{<t}$ is the set of rankings created up to time t. $\mathcal{G}$ is a particular grouping, e.g. an H-level or Econ-level grouping.

Algorithm 1 Outline of the MRFR re-ranking procedure for a single query q. $\mathcal{D}(q)$ is the set of documents for the query.

1: **procedure** MRFR
2: **for** t=1 to t=n **do**
3: $\forall d \in \mathcal{D}(q)$ compute $\phi(d,t)$ ▯ Compute pre-ordering score with Equation (5.4).
4: $\pi_{pre} \leftarrow (\phi(d_1,t), \phi(d_2,t), \ldots, \phi(d_N,t))$ ▯ (Order documents by pre-ordering score)
5: Create candidate ranking π^c for each permutation of the top K documents of π_{pre}
6: $\forall \pi^c$ compute $\psi(\pi^c,t)$ ▯ Compute candidate ranking score with Equation (5.6).
7: Return $\pi = \underset{\pi^c}{\operatorname{argmax}}\ \psi(\pi^c)$ ▯ Return the ranking for this time t
8: **end for**
9: **end procedure**

Thonet and Renders [45] evaluate the performance of MRFR in terms of the fairness metric used in FAIR-TREC 2019, Unf (see Equation (2.12)). As explained in Section 2.3, while Unf and EEL are similar in some ways, they do not target the exact same kind of fairness. Therefore, we alter MRFR to work with the setting of FAIR-TREC 2020. We call our adapted method Robust Fair Ranking with Expected Exposure Loss (RFRE).

We make the following changes. We adapt Equations (5.4) to (5.6) to work with EEL as follows:

$$\phi_{EEL}(d,t) = \rho(d) - \beta\ \delta_{EEL}(d,t), \tag{5.7}$$

$$\delta_{EEL}(i,t) = \sum_{g \in \mathcal{G}(d)} \left(\text{e-ERR}_g^{2020}(g, \Pi_{<t}) - \mathcal{E}_g^*(g, \Pi_{<t})\right), \tag{5.8}$$

and

$$\psi_{EEL}(pi^c, t) = \text{U}(\Pi_{<t} \cup \pi^c) - \lambda\ \text{EEL}(\mathcal{G}, \Pi_{<t}). \tag{5.9}$$

In the above, e-ERR$_g^{2020}$ is computed as in Equation (2.8) and $\mathcal{E}_g^*$ is computed as in Equation (2.42). U is the utility as defined in [5], the same to the original MRFR formulation. Even thoughEELcan be decomposed into a disparity and a relevance component, the relevance component can *not* be directly interpreted as a utility metric [19]. Additionally, because relevance is subsumed intoEEL, it is difficult to explicitly make a trade-off between fairness and utility. Therefore we explicitly include utility as part of ψ_{EEL}.

As in the original description of MRFR, δ is computed over a particular grouping $\mathcal{G}$. In their work, Thonet and Renders [45] assessed how well their method performed using different groupings. To best compare the performance of AC and RFRE as post-processing methods, we use the same grouping for both, namely the grouping used in the baseline version of AC: a grouping by author that ignores missing author information. Similar to how it is for AC, if a document does not have a group (defined by author in this case), we set δ_{EEL} to 0.

A note on estimated probability of relevance Both AC and RFRE need a set of estimated probabilities of relevance ρ to compute their scoring functions. Since we want to use AC and RFRE as post-processing methods for LM, we want to use the scores LM computes for each document as their estimated probabilities of relevance. For both AC as well as RFRE the values of ρ are, inter alia, used as the parameters of a Poisson-Binomial distribution. These parameters should lie between $[0, 1]$ However, LM can predict both positive and negative scores for a document. To make the scores from LM suitable for use as parameters for the Poisson-Binomial distribution we normalize the predictions for each document for each query to lie in the range $[0, 1]$.

Results

We evaluate the performance of AC and RFRE as post-processors. The levels of the `post` factor are `none`, `ac`, and `rfre`. We create runs for each level on `corp2020val_split`. The performance of each level is summarized in Table B.1.

Since we compare three or more systems we use the ANOVA test. We perform two ANOVAs, one for EEL($\mathcal{G}_i, \Pi$) and one for EEL($\mathcal{G}_h, \Pi$) as dependent variable. The results of the ANOVAs are given in Table B.3. Since `post` is significant for both dependent variables, we perform additional Tukey's HSD test, the results of which are shown in Table B.4.

Finally, Figure B.1 shows the mean performances of the three systems with 95% CI using the MS_{error} from Table B.3.

We see from Table B.4 that both `ac` and `rfre` perform significantly better than `none` in terms of EEL($\mathcal{G}_i, \Pi$), although not significantly better than each other. Although we do not see the same significance results when using EEL($\mathcal{G}_h, \Pi$) as a DV, the result on EEL($\mathcal{G}_i, \Pi$) prompts us to evaluate the performance of `ac` and `rfre` on `corp2020test`.

We create runs of `none`, `ac`, and `rfre` on `corp2020test`. The performance of each run is summarized in Table B.2.

To assess whether the performance of post-processing with AC or RFRE is better than that of LM alone, we perform two-sided paired t-tests between `none` and `ac` and `none` and `rfre` with DV EEL($\mathcal{G}_i, \Pi$) and EEL($\mathcal{G}_e, \Pi$). The results of the t-tests are given in Table B.5.

Discussion

The goal of the improvement `ILM_post` was to improve the EEL-D of rankings produced by LM by applying post-processing.

We applied two post-processing methods, AC and RFRE. We see that both methods significantly decrease the EEL($\mathcal{G}_i, \Pi$) on `corp2020val_split`; the value for either set of post-processed rankings is around 3 times smaller than for baseline LM. As we hoped, the improvement is mostly due to a decrease in disparity: EEL-D($\mathcal{G}_i, \Pi$) goes from 1.378 to 0.459 for AC and from 1.378 to 0.554 for RFRE. The EEL-R($\mathcal{G}_i, \Pi$) also goes down, but ends up being only around 1.3 times smaller for the post-processed rankings than for the baseline. In other words: post-processing with AC or LM improves the EEL($\mathcal{G}_i, \Pi$) by reducing the disparity at a small cost to the relevance.

The positive effect of post-processing on EEL($\mathcal{G}_i, \Pi$) generalizes to `corp2020test`. The effect is somewhat less pronounced with a factor of around 2.5, but the reduction is still significant. We also see that the effect again is mainly due to a reduction in EEL-D($\mathcal{G}_i, \Pi$).

When we evaluate the performance with different groupings the effect of the post-processing is less pronounced. For EEL($\mathcal{G}_h, \Pi$) on `corp2020val_split` neither post-processing method has a significant effect. Interestingly, the effect size for the comparison between baseline and either `ac` or `rfre` is large. For EEL($\mathcal{G}_e, \Pi$) on `corp2020test` the difference between the baseline and either AC or RFRE *is* significant, but the effect size ES_t is about 3 times smaller than for EEL($\mathcal{G}_i, \Pi$). It is hard to say whether the difference in significance between `corp2020val_split` and `corp2020test` is due to the chosen grouping or due to the fact that `corp2020test` is much larger than `corp2020val_split`. In any case it seems that performance on an arbitrary grouping on training data does not necessarily translate to similar performance on test data.

We note that there is no significant difference in the performance of AC and RFRE as post-processors. This may be due to the fact that both methods work in a similar way. Both post-processors compute a kind of accumulated advantage across multiple rankings that is then used to advantage or penalize documents in

the next ranker. Additionally, we chose to adapt the original Multi-grouping Robust Fair Ranking algorithm in such a way that made the resulting RFRE algorithm even more similar to AC by including the term $\mathcal{E}_g(t)$ in Equation (5.8) and computing it in the same way as we did for AC.

5.2.2. Improving relevance with feature selection

One of the causes affecting the relevance of LM is `CLM_features`. It was at times unclear whether all features used for the baseline were equally useful.

For any learned ranker to achieve good accuracy it is important to select the correct features [26]. Feature selection can remove noisy and redundant features resulting in a smaller featureset, which in turn reduces the chance of overfitting. This is especially useful for situations in which we have limited training data, such as is the case for us.

It is difficult to select good features by hand; even with a small number of features the number of possible combinations rapidly grows too large to try manually. A possible solution to this problem is to use a feature selection algorithm. There are three kinds of feature selection algorithms: filter algorithms which select features before training, wrapper algorithms which test sets of features at a time, and embedded algorithms which test features during the training process [26].

Since wrapper and embedded feature selection algorithms require altering the ranker itself, we focus on a filter approach. Djafari Naini and Altingovde [21] compare five filter-based feature selection algorithms: Top-K Relevant (Top-K), Greedy search Algorithm for feature Selection (GAS), Maximal Marginal Relevance (MMR), MaxSum Dispersion (MSD), and Modern Portfolio Theory (MPT). The goal of each of these select those features that maximize an effectiveness measure such as NDCG when used to rank a set of training samples. In the case of GAS, MMR, MSD, and MPT, the goal is also to select features that are as diverse as possible.

[21] evaluate the performance of the aforementioned algorithms on the OHSUMED, MQ2008, and Yahoo! SET2 datasets. Of these sets, OHSUMED and MQ2008 are the closest to our dataset in terms of number of queries and number of samples. The algorithms that perform best on OHSUMED and MQ2008 are MSD and MPT, so we investigate for these feature selection methods whether they improve LambdaMart's performance in terms of accuracy.

Since MSD and MPT combine a focus on diversity and a good performance on the most similar dataset we assess whether these feature selection algorithms improve the performance of LM. We refer to the improvement of applying a feature selection algorithm as `ILM_fsm` for *feature selection method*.

Both selected methods allow us to specify the number of selected features and a balancing factor. The number of selected features speaks for itself. We set it to 10 to match the number of features used for baseline LM. The balancing factor is a value between $[0, 1]$ which determines whether the feature selection should favor effectiveness or diversity. We set the balancing factor to 0.5.

We devise a number of new features for the feature selection algorithms to choose from. We combine the features used for baseline LM with applicable features for LETOR 4.0 [35]. LETOR is a set benchmark datasets for LtR that are widely used.

In the interest of time, we add only those features that can be computed with the Elasticsearch LtR plugin and features that can be computed from the queries. The full list of features is given in Table 5.2. The tables should be read as follows: Table 5.2a lists the number for each feature. The *fields* column shows which fields can be used with each feature. The fields themselves are shown in Table 5.2b and should be read in order. For example: feature 1 is the BM25 score of the *title* field; feature 7 is the BM25 score of *fields of study*.

The functions listed in the *column* function of Table 5.2a are BM25, TF, IDF, and TF*IDF, sections 11.4.3, 6.2, 6.2.1, 6.2.2 respectively in Manning, Raghavan, and Schuetze [30].

Finally, Table 5.2c lists the features selected by MPT and MSD.

Results

We evaluate the performance of each feature selection method on `corp2020val_split`. The levels of the feature selection method factor `fs` are `none`, `msd`, and `mpt`. We create runs for each level on `corp2020val_split` and compute the EEL($\mathcal{G}_i, \Pi$) and EEL($\mathcal{G}_h, \Pi$). The mean performance of each level is given in Table B.6.

Since we compare three or more systems we use the ANOVA test. We perform two ANOVAs, one for EEL($\mathcal{G}_i, \Pi$) and one for EEL($\mathcal{G}_h, \Pi$) as dependent variable. The results of the ANOVAs are given in Table B.7. Since `fs` is not significant for either DV we do not perform additional Tukey's HSD test.

Finally, Figure B.2 shows the mean performances of the three systems with 95%CI using the MS_{error} values from Table B.7.

Because the effects of using a feature selection method were not significant we do not assess the performance on `corp2020test`.

Discussion

The goal of the improvement `ILM_fsm` was to improve the relevance of LM by selecting better features.

We used two feature selection methods, MPT and MSD. We see that neither method has a significant effect on the EEL on `corp2020val_split`. There are a number of possible reasons for this.

The first reason is that while we gave the feature selection methods a lot of features to choose from, some of the features were simply variations on each other. For example, there is likely a relationship between field length in terms of characters and tokens and between TF and IDF and TF*IDF. It is possible that including many similar features limited the ability of the feature selection methods to distinguish useful features from each other.

Another issue is that the feature selection algorithms we chose are *learned* algorithms just like LM. We trained and tested MPT and MSD on `corp2020train`. We used `corp2020train_split` as the training set and `corp2020val_split` as the test set. However, as we identified in Table 4.3a, the dataset is noisy, with some relevant documents being labeled as non-relevant and vice-versa. This noisiness makes it more difficult for MPT and MSD to select good features. In addition, `corp2020train` consists of only 200 queries, so just like with LM at large it is possible that MPT and MSD suffer from underfitting.

All-in-all, using feature selection algorithms did not remedy `CLM_features`. It would be interesting to see whether a larger or cleaner dataset would allow MPT or MSD to select better features. It may also be useful to add more information to the dataset to provide more features for the algorithms to choose from, e.g. in the form of inferred gender or country of affiliation as done for another submission to FAIR-TREC 2020 [24]. With the dataset being as it is, a more fruitful approach may be to manually construct or select more effective features.

5.2.3. Improving relevance through choice of effectiveness measure

LM can optimize for different effectiveness measures. The standard option is NDCG, which may be a sub-optimal choice for our context. We identified this as cause `CLM_measure`.

As we see in Section 2.3.2, our target measure EEL is built upon ERR. We use ERR to measure how much exposure a document receives, which we then use to determine whether the exposure is distributed fairly. The relevance EEL-R in turn is derived from EEL. Burges et al. [10] show that it is possible to optimize LM for ERR as well. Since the two metrics are related, by optimizing for ERR we may be able to improve EEL-R.

`pyltr` implements ERR as one of its optimization measure out of the box. We refer to the improvement of using ERR as the optimization measure as `ILM_measure`.

Results

We evaluate the performance when using ERR as the optimization measure on `corp2020val_split`. The levels of the factor `measure` are `ndcg` and `err`. The mean performance of each system is given in Table B.8. We compare the performance of the systems with two paired two-sided t-test with DV EEL($\mathcal{G}_i, \Pi$) and EEL($\mathcal{G}_h, \Pi$). The results of the t-tests are given in Table B.9. Finally, Figure B.3 shows the mean performances of `ndcg` and `err` with 95%CI computed as in Equation (5.3).

The effects of using ERR as the effectiveness metric are *not* significant. Therefore, we do not assess the performance on `corp2020test`.

Discussion

The goal of `ILM_measure` was to improve the relevance of LM by using a more appropriate effectiveness measure for optimization.

We compared the baseline optimization measure NDCG with the performance of ERR. We see that the base version performs better on both EEL($\mathcal{G}_i, \Pi$) and EEL($\mathcal{G}_h, \Pi$) as well as the sub-measures for relevance and disparity, although the difference is not statistically significant.

We note two possible reasons for the lack of improvement due to using ERR. Firstly, the version of ERR as it is used in the library we use for LM, `pyltr`, is not fully compatible with the implementation of ERR in the evaluation tools of FAIR-TREC 2020. There are two parts to the ERR measure that can be changed: the utility at rank j $\phi(j)$ and the probability of the user being satisfied by the document

d at j, $R(d)$ Chapelle et al. [14]. `pyltr` and `trec_eval_tools` differ both in how they define ϕ and how they define $R(j)$.

In `pyltr` ϕ is defined as described in Chapelle et al. [14, Sec. 4],

$$\phi(j) = \frac{1}{j}, \tag{5.10}$$

and $R(j)$ is given by

$$R(d) = \max(0, e^{rel(d)} - \frac{1}{2})^{4}. \tag{5.11}$$

However, in the FAIR-TREC 2020 evaluation tools, ϕ is as in Chapelle et al. [14, Sec. 7.2]:

$$\phi(j) = \gamma^{1-j}, \tag{5.12}$$

where γ is a patience parameter indicating how long the user searches before giving up, and

$$R(j) = \kappa \cdot rel(d), \tag{5.13}$$

where κ determines how heavily the relevance grade of the document should weigh.

The result of this difference is that LM is not optimized for the form of ERR underpinning EEL and as such does not increase relevance.

Additionally, Burges [11] find that on their specific dataset *training* LM for ERR results in a lower ERR on the *test* data, although they do not report whether their results are statistically significant or not. They find that optimizing for NDCG yields a better ERR instead. It is unclear why this effect occurs, but it is possible that by optimizing for ERR we are *harming* relevance rather helping it. Although the results from our t-tests were not significant, we do see in Figure B.3 that disparity is higher for `err` than for `ndcg`, while relevance is roughly the same, supporting this idea.

All-in-all, optimizing for ERR rather than NDCG does not seem likely to yield better results, even if we used the same versions of ERR to optimize and to measure. However, it would be interesting to see if it is possible to optimize for EEL directly.

5.2.4. Unaddressed causes

We identified three causes relating to the dataset, `CLM_noisy`, `CLM_ambiguous`, and `CLM_size`. We were unable to address these in this thesis, for the following reasons.

Noisy data `CLM_noisy` is caused by the process by which the data was gathered. We neither have access to the original click data from which the relevance labels were derived, nor does the time allotted for this thesis permit us to re-assess

[4] `pyltr/pyltr/metrics/err.py` at https://github.com/jma127/pyltr

all annotated documents manually. As such, we do not address `CLM_noisy` ourselves. Later editions of FAIR-TREC use datasets annotated by NIST assessors, see e.g. Ekstrand et al. [23, Sec. 3.4]. It would be interesting to see if a similar treatment of the dataset for FAIR-TREC 2020 would improve the usefulness of the dataset.

Ambiguous queries The ambiguity of queries we identified in `CLM_ambiguous` is a well known challenge in IR. Numerous techniques to combat this have been proposed over time. One of the most successful techniques is that of relevance feedback. In this process, users provide feedback on an initial set of results. The system then uses this feedback to update its knowledge of the relevance of documents and return better results. This process can be repeated a number of times [30, Sec. 9.1].

We did not investigate relevance feedback or related techniques both because of the high amount of resources it requires and because we wanted to focus on fairness rather than relevance. It would be interesting to see whether relevance feedback can be adapted in such a way that users can contribute feedback not only on the effectiveness of results, but on the perceived fairness as well. We leave this as a recommendation for future work.

Small dataset The last unaddressed cause is `CLM_size`. The small number of queries and annotated documents, especially in combination with the noisyness of the labels, makes it hard for LtR-based systems to be trained effectively. A larger number of annotated documents and queries would make it easier to improve rankers for the FAIR-TREC 2020 setting.

5.2.5. A note on query difficulty

Throughout the preceding experiments, we note that the `qid` factor is consistently significant and that the effect size is large compared to that of any of the stand-ins for the `system` factor. This is most likely due to the small sample size. `corp2020val_split` contains only 20 queries so if any one of the queries is more difficult than another this impacts the overall score. As such, it is difficult to say whether any significant effects we detected will generalize to larger datasets. Conversely, effects that were not significant on `corp2020val_split` might be significant on a larger dataset.

5.3. Improvements to Advantage Controller

We propose and evaluate improvements for the causes of failure of AC listed in Table 4.3a. Both failure causes affect all aspects of EEL. We propose three remedies for `CAC_no_author`: (i) adding dummy author information, and (ii) using a different scoring function h. We propose one remedy for `CAC_rel`: using a better set of estimated probability of relevance values.

We list the improvements in Table 5.1. We now describe each improvement in more detail.

5.3.1. Remedying missing data with dummy authors

The first point of intervention for `CAC_no_author` is the data itself. Documents without authors are advantaged by AC because for these documents $A_d(d,t) = 0$. If we can add author information to documents that lack it, this removes the disproportional advantage authorless documents receive.

One way in which we could add author information is by supplementing the data retrieved through the official instructions for FAIR-TREC 2020 with data from other sources. For example, Feng et al. [24] supplement `corp2020`with information on the country of origin of each author by searching for authors on Google Scholar. We could do something similar by looking for the titles of documents and retrieving the authors for each document. However, there are two limitations to this approach.

Firstly, AC works by looking at the author *id* rather than the author *name* to determine advantage. This is because the author name can be ambiguous, e.g. multiple people can have the same name, while the id is unique. However, the id is particular to the SSOC dataset and as such cannot be retrieved from a different source. We could re-generate ids for all authors, but then we would be unable to distinguish between authors sharing the same name.

Secondly, some documents do not have an author simply because the author is unknown or anonymous. There is no way to remedy this by looking in external sources.

Since we cannot retrieve the author information from external sources we generate *dummy authors* instead. There are two options: Either we treat all authorless documents as if they have the *same* author, or we treat all authorless documents as if they have *different* authors. In the first case, we assign all documents to the same dummy author. In the second case, we use the document id as a proxy for the author id. In both cases, we assume a document has *one* author.

We refer to this improvement as `IAC_dummy`.

Results

We evaluate the performance of AC with dummy author information. The levels of the `dummy` factor are `none` for the baseline, `one` for a single dummy author for all authorless documents, and `ind` for a different dummy author for each aughorless document. We create runs for each level on `corp2020val_split`. The performance of each level is summarized in Table B.10.

We perform two ANOVAs, one for EEL($\mathcal{G}_i, \Pi$) and one for EEL($\mathcal{G}_h, \Pi$) as dependent variable. The results of the ANOVAs are given in Table B.11. Since `dummy` is significant for both dependent variables, we perform additional Tukey's HSD test, the results of which are shown in Table B.12.

Finally, Figure B.4 shows the mean performances of the three systems with 95% CI using the MS_{error} from Table B.11.

We see from Table B.12 that the effects of using dummy authors are not significant. As such, we do not assess the performance on `corp2020test`.

Discussion

The goal of `IAC_dummy` was to prevent AC from unfairly advantaging documents without an author by assigning dummy authors to authorless documents. We see

that neither of our strategies for assigning dummy authors yields a significant difference in performance. Even so, if we look at Figure B.4, we do see that both `ind` and `one` lead to a lower EEL($\mathcal{G}_i, \Pi$) and EEL($\mathcal{G}_h, \Pi$) due to a lower disparity. This indicates that assigning dummy information may help combat the unfair advantaging of documents without authors after all. This is confirmed if we again plot the occurrences of each document for query 31412. Compare Figure 5.1 with Figure 4.4. We see that for both `one` and `ind` the document that previously spiked above the other documents no longer does so.

One reason that the effect does not show up as significant may be due to the small size of `corp2020val_split`. Due to this small size the differences in performance need to be larger for them to be significant. As such, it would be interesting to see whether this effect is more pronounced on a larger dataset.

5.3.2. Remedying unequal advantages with the scoring function

The second point of intervention is the scoring function h. Recall the formula for h as written in equation Equation (3.1):

$$h(d, t, q) = \theta\rho(d, q) - (1 - \theta)A(d, t).$$

The value of h is smaller for documents with a larger A, e.g. documents that previously have received more exposure than other documents, and vice versa for documents that previously have received less exposure than other documents. Since for authorless documents $A = 0$ at all times, the value of h only depends on the $\rho(d, q)$.

To equalize the influence of A between documents with and without authors we adapt h. We want to make sure that the documents are treated the same regardless of whether they have an author or not. One way to do so is to make sure that documents *with* authors cannot receive positive A values, just like authorless documents. In other words, the final score can only be larger than their estimated probability of relevance, not smaller. To this end, we define a new scoring function:

$$h_{min}(d, t, q) = \theta\rho(d, q) - (1 - \theta)\min(0, A(d, t)). \tag{5.14}$$

The term $\min(0, A(d, t))$ makes it so that each document has an advantage of 0 or less.

We also try a second variation. Since $A(d, t) = 0$ for authorless documents and $\rho \geq 0$, h cannot be below 0 for documents without authors. To make this the same for documents *with* authors, we define a second variation on h:

$$h_{max}(d, t, q) = \max(0, \theta\rho(d, q) - (1 - \theta)A(d, t)). \tag{5.15}$$

We refer to the full improvement as `IAC_hfunc`.

Results

We evaluate the performance of AC with different scoring functions. The levels of the `hfunc` factor are `linear` for the baseline with h, `max` for scoring with h_{max}, and

`min` for scoring with h_{min}. We create runs for each level on `corp2020val_split`. The performance of each level is summarized in Table B.13.

We perform two ANOVAs, one for EEL($\mathcal{G}_i, \Pi$) and one for EEL($\mathcal{G}_h, \Pi$) as dependent variable. The results of the ANOVAs are given in Table B.14.

Since `hfunc` is significant for both EEL($\mathcal{G}_i, \Pi$) we perform an additional Tukey's HSD test, the results of which is shown in Table B.12.

Finally, Figure B.5 shows the mean performances of the three systems with 95% CI using the MS_{error} from Table B.14.

Although we see from Tables B.14 and B.15 that the effects of using in particular h_{min} is significant, the performance is *worse* than for the baseline option of h. As such, we do not assess the performance on `corp2020test`.

Discussion

The goal of `IAC_hfunc` was to equalize the influence of authorless documents receiving 0 advantage at all times by adapting the scoring function so that either (i) the maximum advantage *any* document can receive is 0, or (ii) the minimum overall score any document can receive is 0. We see that in terms of EEL($\mathcal{G}_i, \Pi$), h_{min} performs significantly *worse* on `corp2020val_split` than either other option. There is no significant in performance between the baseline scoring function h and h_{max}.

Figure 5.2 confirms that h_{max} does *not* remedy `CAC_no_author`. We see that the authorless document is still over-represented. This is probably because there were few documents that received a negative score in the first place, so using h_{max} caused little difference in the behavior.

Interestingly, h_{min} *does* seem to resolve the issue of the over-represented authorless documents. If we look at the occurrences in top positions for the documents for query 31412 in Figure 5.2b, we see that there is no longer a document without an author that occurs disproportionately often. However, the problem seems to have shifted to a new non-relevant document. This document has the highest ρ out of all documents for the query. In other words: while setting the maximum advantage for all documents to 0 prevents the over-representation of authorless documents, h_{min} over-emphasizes the estimated probability of relevance ρ.

Overall we can say that using h_{min} successfully addresses the specific cause of `CAC_no_author`, but at the cost of introducing a new flaw. In that sense it is less promising than the previous point of intervention.

5.3.3. Improving performance with more accurate relevance predictions

The second cause we identified for AC, `CAC_rel`, is the relatively low estimated probabilities of relevance for some of the relevant documents. Higher or more accurate estimated probabilities of relevance may improve the performance of AC.

The sets of estimated probability of relevance probabilities `rel_set_A` and `rel_set_B` were generously provided to us by the authors of the original AC paper. Since we did not reproduce the process through which these relevance probabilities

were generated, we cannot affect them. However, just as we used AC as a post-processing method for LM in Section 5.2.1, we can use LM to generate predict probabilities of relevance for AC.

We train LM with the features and hyper-parameter settings as described in Tables 3.2 and 3.3. In the prediction phase, LM computes a score for each document for each query. These scores can be positive or negative. To use the predicted scores with AC they need to lie in the range $[0, 1]$ so they can be used as parameters for a Poisson-Binomial distribution. Kletti and Renders [27] use Isotonic Regression for normalizing classification probabilities [50] to generate their estimated relevance probabilities, although the process is are unclear. For the sake of simplicity we simply normalize the predicted scores from LM to lie between 0 and 1.

We see in Table 4.2 that baseline LM achieves an EEL$(\mathcal{G}_i, \Pi)$ of 1.378 and an EEL$(\mathcal{G}_h, \Pi)$ of 0.762 on `corp2020val_split`. By contrast, if we create a ranking based solely on the estimated probabilities of relevance in `rel_set_A`, we achieve a score of EEL$(\mathcal{G}_i, \Pi) = 1.438$ and EEL$(\mathcal{G}_h, \Pi) = 0.862$. We see that LM performs better, implying that the relevance scores are more accurate. We refer to the improvement of using LM to generate estimated relevance values as `IAC_rels`.

Results

We evaluate the performance of AC with different sets of estimated probabilities of relevance. The levels of the `rels` factor are `rel_set_A` for the baseline and `rel_set_lm` for the normalized scores predicted by LM. We create runs for each level on `corp2020val_split`. The performance of each level is summarized in Table B.16.

We compare the performance of the systems with two paired two-sided t-test with DV EEL$(\mathcal{G}_i, \Pi)$ and EEL$(\mathcal{G}_h, \Pi)$. The results of the t-tests are given in Table B.18.

Finally, Figure B.6 shows the mean performances of `rel_set_A` and `rel_set_lm` with 95%CI computed as in Equation (5.3).

Because the performance of AC with the estimated probabilities of relevance from LM is significantly better in terms of EEL$(\mathcal{G}_i, \Pi)$ on `corp2020val_split` we create additional runs of `rel_set_A` and `rel_set_lm` on `corp2020test`. The performance of each run is summarized in Table B.17.

To assess whether the performance with `rel_set_A` or `rel_set_lm` is better we perform two-sided paired t-tests between `rel_set_A` and `rel_set_lm` with DV EEL$(\mathcal{G}_i, \Pi)$ and EEL$(\mathcal{G}_e, \Pi)$. The results of the t-tests are given in Table B.19.

Discussion

The goal of the improvement `IAC_rels` was to improve the performance of AC by providing more accurate estimated relevance probabilities as input. We see that on both `corp2020val_split` and `corp2020test` this method yielded significant improvement for one or both groupings, mainly due to an increase in relevance on `corp2020val_split` and both an increase in relevance and a decrease in disparity on `corp2020test`. However, this is not due to `rel_set_lm` being more accurate, indeed the mean difference between true and estimated probability of relevance is 0.03 for `rel_set_A` versus 0.30 for `rel_set_lm` for

corp2020val_split. On corp2020test those values are 0.01 and 0.36 respectively.

Instead, the effect can be explained by the fact that the estimated probabilities of relevance in rel_set_lm are simply *higher* than those for rel_set_A. The average estimated probability of relevance in rel_set_lm is 0.48 on corp2020val_split and 0.51 on corp2020test. For rel_set_A it is 0.16 on corp2020val_split and 0.16 on corp2020test. Higher overall estimated probabilities of relevance cause more emphasis on relevance in computing h-score. This indicates that AC currently over-emphasizes fairness to the detriment of relevance. A more balanced approach could lead to a lower overall EEL.

5.4. Summary

In this chapter, we proposed and evaluated improvements to LM and AC. We propose the following improvements:

- LM
  - ILM_post: LM generates a single ranking for each query, leading to a high EEL-D. We applied fairness-aware post-processing methods to generate different rankings. We tried two post-processing methods, both of which significantly improved the performance.
  - ILM_fsm: Selecting the correct features to use with a LtR algorithm like LM can improve the performance. We designed a number of new features for LM and used two feature selection algorithms to select the best features. We found that neither set of selected features yielded a significant improvement relative to the original features. Since we have a small dataset, using feature selection algorithms was likely an overly complicated solution for our problem in this case.
  - ILM_measure: LM is most commonly optimized for NDCG, but our fairness metric EEL is based on the ERR model. We optimized LM for ERR instead. We found that using ERR does not significantly improve the performance and in fact that there is an indication that it decreases the performance. This may be due to a mismatch in the specific form of ERR used in the the implementation of LM and the form used in FAIR-TREC 2020.
- AC
  - IAC_dummy: Some non-relevant documents did not have an author. These documents were unfairly advantaged because their advantage in Equation (3.12) was always set to 0. We tried two strategies of assigning dummy authors to these documents. Neither strategy yielded a significantly better performance than the baseline, but there is an indication that on a larger dataset the effect may be more pronounced.

- `IAC_hfunc`: The other remedy to the unfair advantage of documents without authors was to use different scoring functions that counteract the effect of some documents consistently receiving an advantage of 0. We tried two alternative scoring functions. Of these, one had a significant impact, but the effect was to make the overall performance worse. Documents without authors were no longer overly favored, but instead documents with a higer estimated probability of relevance became overrepresented.
- `IAC_rels`: The estimated probability of relevance was too low for some relevant documents, leading them to be ranked worse than they should. We trained LM and used it to predict scores for each document. We then normalized the scores to lie in the range $[0, 1]$ and used them as if they were estimated relevance probabilities. The probabilities generated in this way were higher on average than those in `rel_set_A` which we received from Kletti and Renders [27]. The performance with the LM probabilities was significantly better than with `rel_set_A` for the individual grouping, indicating that AC benefits from increased focus on relevance.

There were a number of causes we did not address because they required us to alter `corp2020` itself, e.g. in the form of re-assessing relevance labels.

While implementing improvements we encountered three limitations of the FAIR-TREC 2020 benchmark:

- Limited usefulness of groupings.

  We evaluated each improvement both with an individual grouping and either an H-level grouping on `corp2020val_split` or an Econ-level grouping on `corp2020test`. The goal was to gain insight into the effect of the chosen grouping on performance. However, we found that using groupings mostly confounded the analysis because it was hard to reason about *why* the groupings caused the observed effects. Additionally, the stated goal of using EEL as a fairness measure is to created rankers that are fair towards *arbitrary* groupings. We are unable to assess whether this goal is met by using *specific* groupings in our evaluation.

- Limited ability to improve specific aspects of a ranker in a targeted manner

  The EEL metric consists of sub-metric EEL-D, EEL-R, and EEL-C representing *disparity*, *relevance*, and a per-query constant value respectively. There is no one-to-one correspondence between these sub-metrics and traditional effectiveness measures such as NDCG. This makes it hard to gauge what the tradeoff is between achieving greater fairness and the effectiveness of a system.

- Limited analysis due to small sample size

  Some queries are inherently more difficult than others, e.g. an ambiguous query is inherently harder than a specific query. To separate the effect of query difficulty from that of the improvements, we treat query id as

a factor in our ANOVA analyses. We found that `qid` is consistently a significant factor with a large effect size. This is most likely due to the small size of the dataset we performed the analysis on, `corp2020val_split`. `corp2020val_split` consists of only 20 queries, meaning that any difference in difficulty has a large impact. As such, our analysis are less conclusive than they would have been with a larger dataset.

Number	Level	Function	Fields
1 - 7	query-document	BM25	`fields_text`
8 - 14		TF	
15 - 21		IDF	
22 - 28		TF*IDF	
29 - 35	document	# chars	`fields_text`
36 - 42		# tokens	
43 - 45	document	numerical value	`fields_numerical`
46	query	# chars	
47		# tokens	

(a) Features for use with the feature selection algorithms for `ILM_fsm`.

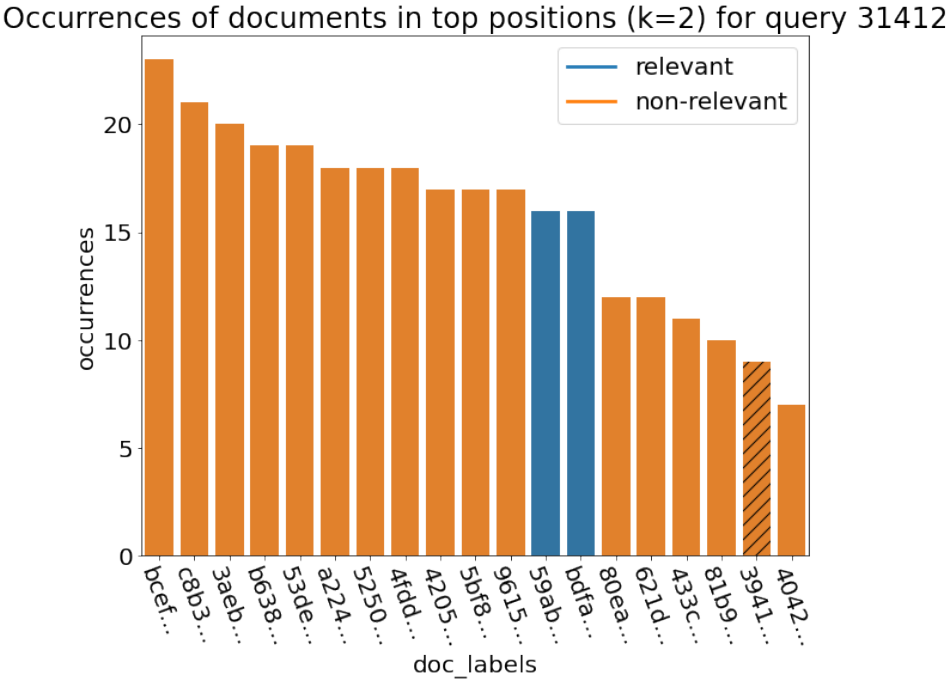
`fields_text`		`fields_numerical`	
Numbers	Field	Numbers	Field
1,8,15,22,29,36	title	43	year
2,9,16,23,30,37	abstract	44	inCitations
3,10,17,24,31,38	venue	45	outCitations
4,11,18,25,32,39	journal		
5,12,19,26,33,40	author names		
6,13,20,27,34,41	sources		
7,14,21,28,35,42	fields of study		

(b) Fields for features 1-45.

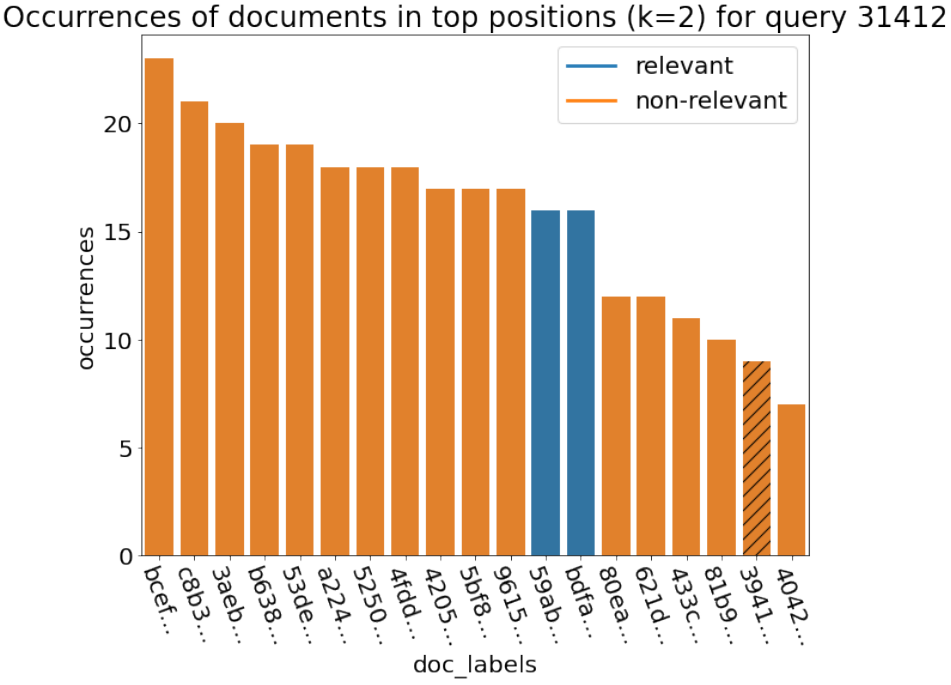
MPT		MSD	
Number	Feature	Number	Feature
1	BM25 title	1	BM25 title
2	BM25 abstract	2	BM25 abstract
30	# chars abstract	8	TF title
33	# chars author names	23	TF*IDF abstract
35	# chars fields of study	29	# chars title
36	# tokens title	30	# chars abstract
39	# tokens journal	37	# tokens abstract
40	# tokens author names	43	year
43	year	44	# inCitations
44	# inCitations	45	# outCitations

(c) Features selected by MPT and MSD.

Table 5.2: The features for use with `ILM_fsm`. The features themselves are listed in the table on the left. The fields for features 1-42 are listed in the upper table on the right, the fields for features 43 table to the right.

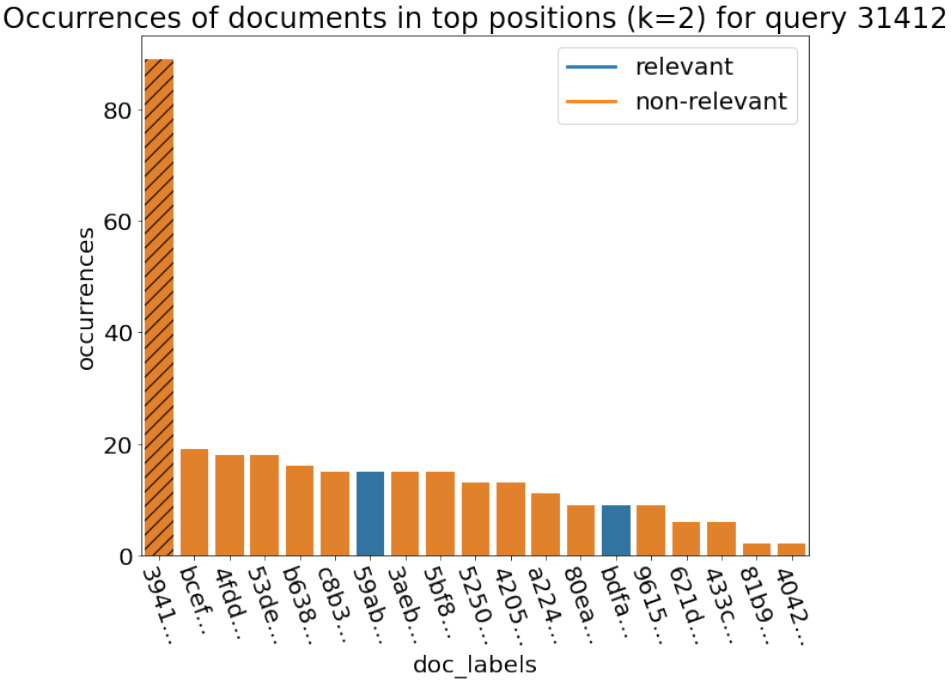


(a) `IAC_dummy` variation: `ind`

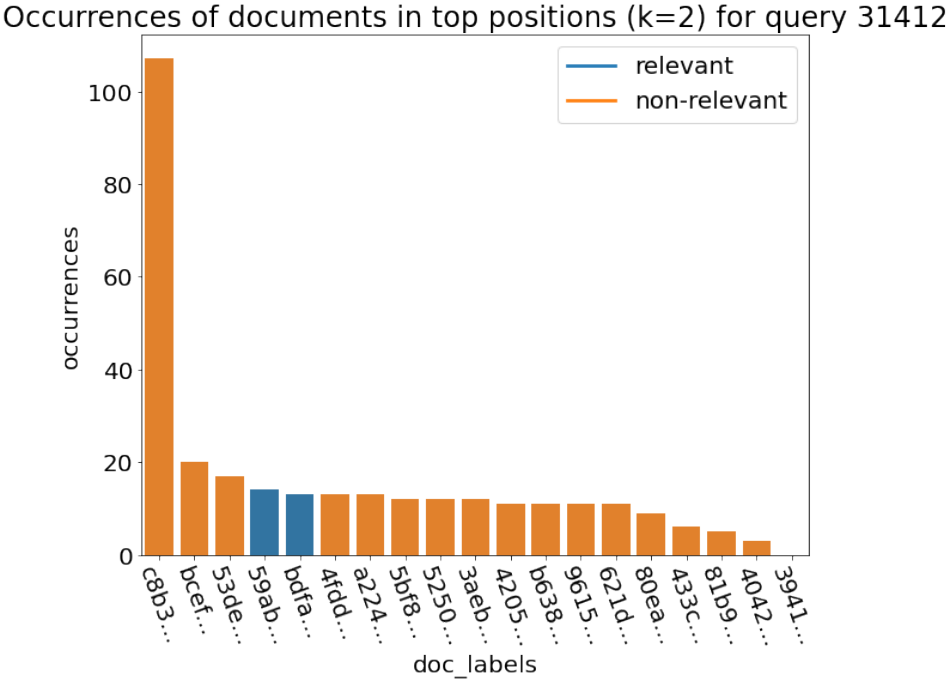


(b) `IAC_dummy` variation: `one`

Figure 5.1: The number of times each document appears in a top position for query `31312` when ranked by AC with `rel_set_A` and $\theta = 0.9$ and two different approaches to assigning dummy authors to documents.



(a) `IAC_hfunc` variation: `max`



(b) `IAC_hfunc` variation: `min`

Figure 5.2: The number of times each document appears in a top position for query `31312` when ranked by AC with `rel_set_A` and θ = 0.9 and two different scoring functions.

6

Conclusion

The goal of this thesis was to contribute to the developing field of fair IR by improving existing rankers and identifying strength and limitations of the FAIR-TREC benchmark, in particular the 2020 edition. We present our conclusions pertaining to the factors affecting the fairness of our rankers in Section 6.1 and the conclusions pertaining to FAIR-TREC 2020 in Section 6.2. We summarize the limitations of our approach in Section 6.3 and finish with recommendations for future work in Section 6.4.

6.1. Factors affecting the fairness of amortized fair rankers

We analyzed the performance of two rankers, LM and AC in terms of the fairness measure EEL. These rankers represent two radically different approaches. LM has been a highly effective and popular LtR method since its invention in 2011 and is commonly used as a baseline. However, LM does not mediate disparity in any way.

AC was the top-performing submission to the FAIR-TREC 2020 track. It explicitly takes into account the amount of exposure different documents have received across rankings and balances the amount of exposure each document receives. While AC *does* take relevance into account, reranking documents in a way that decreases disparity inevitably decreases relevance as well.

For each ranker, we found one factor that significantly affected the fairness.

For LM, we found that the most successful intervention was to apply a fairness-oriented post-processing method. We tried two post-processing methods, AC and RFRE. Both methods reduced disparity by around a factor 5 on validation data and by a factor 4 on test data. AC was the most successful overall, reducing EEL by a factor 3 on validation data and a factor 2.5 on test data.

AC takes a set of estimated probabilities of relevance for each document for each query as input. We found that the most successful intervention was to increase the magnitude of the estimated probabilities of relevance. This allowed AC to more

strongly favor accuracy over fairness which in this case led to a lower overall EEL. Improving the accuracy of the estimated probabilities of relevance improved the fairness by a factor 1.1 on validation data and a factor 1.1 on test data.

Interestingly, the factors that significantly affect the performance of each ranker are complementary to the focus of the rankers themselves, ie. LM focuses on relevance and the significant factor is to reduce disparity, AC focuses on disparity and the significant factor is to improve relevance. In other words, it seems as if the most significant factors are those that target the main weak point of the ranker.

We conclude the following. The main factor affecting relevance-based rankers such as LM is a lack of re-ranking. Applying a post-processing method is an easy way to improve the fairness. We hope that our work encourages future work into post-processing in combination with common baselines like LM.

The main factor affecting the fairness-oriented AC ranker is the relevance of its input probabilities. This shows that while AC can significantly improve fairness of relevance-based rankers, it is affected strongly by the output of those rankers. It would be interesting to see whether other post-processing methods are similarly affected.

6.2. Strengths and limitations of the FAIR-TREC 2020 benchmark

The FAIR-TREC 2020 benchmark is an important first step in providing a solid groundwork for fair IR research. The availability of a corpus, annotated training and evaluation data, and a predefined fairness measure makes it possible to compare different rankers with each other while eliminating the variance that comes from everyone choosing their own datasets and defining their own measures. This unification makes results more comparable and scientifically sound [51].

Even so, we encountered a number of limitations of the FAIR-TREC 2020 benchmark. We found limitations of the corpus and of the fairness measure.

The corpus is gathered semi-automatically from click logs. As a result, the relevance labels are noisy. This affects the performance of LtR rankers as they are unable to be trained properly. More thorough annotation of documents could improve the usefulness of the benchmark. Later versions of FAIR-TREC use other datasets, but the SSOC should not be forgotten.

Another downside is the size of the training and test sets `corp2020train` and `corp2020test`. There were a number of improvements we applied to the rankers that did not yield a significant effect even though we expected they would. For example, we saw that missing authors caused some documents to be severely over-exposed. While adding missing author information remedied this problem, the effect on the overall performance was not significant. While it is possible that the improvement truly did not affect the performance in a meaningful way, another potential reason is that the dataset was not large enough properly distinguish any variance caused by the difficulty of the querys from the variance caused by the improvements.

The fairness measure EEL has some limitations. Firstly, EEL combines both

disparity and relevance into a single measure. This makes it difficult to target either disparity or relevance since any intervention will invariably affect each aspect of the measure. Additionally, this makes it difficult to optimize for EEL directly. A combination between a fairness measure and an explicit effectiveness measure may be more useful here.

Using a separate effectiveness measure also makes it easier to gauge what the trade-off is between fairness and effectiveness. The sub-measure EEL-R does not translate one-to-one to utility for the user, so it is hard to say whether any intervention negatively affects the user experience.

Lastly, EEL is computed over a particular grouping. But the stated goal of FAIR-TREC 2020 is to achieve fairness to *arbitrary* groupings. It is not possible to measure this with EEL and a single grouping alone.

6.3. Limitations of our approach

Any approach to fairness can inherently only cover a small part of this wide field of research. Since we focus specifically on fairness as it is defined for FAIR-TREC 2020 our conclusions are not necessarily applicable to other forms of fairness. In addition, we analyzed only 2 out of 20 algorithms for which submissions were made to FAIR-TREC 2019 or 2020. While we endeavored to choose interesting algorithms, we cannot say for sure that our conclusions generalize to other algorithms as well.

Aside from this, the main limitation we encountered is with our modified failure analysis method. We ran into a number of issues we did not foresee in advance. Most important among these are the limited use of EEL as a measure of difficulty and the limited ability of our method to detect complex causes of failure.

Following the failure analysis method from Buckley [9], we ranked queries by their difficulty to determine which queries would be most interesting to analyse. We chose EEL as our measure of difficulty. However, because EEL includes the component EEL-C we cannot compare the performance of two queries with each other one to one, since EEL-C changes per query. Therefore, saying that a query with a higher EEL is inherently more difficult than a query with a lower EEL is not necessarily true. As a result, it is possible that we missed failures that we could otherwise have detected.

The complexity issue is that the questions in the modified failure analysis template focus on the number of occurrences in top and bottom positions of (non)-relevant documents. These questions do not allow us to detect failures that depend on the interaction between different documents. For example, the advantage term in Equation (3.1) depends on all documents in a ranking that share an author. It is possible that there were failures related to the interaction between document position and shared authorship. We were unable to find any such failures, but we also could also not exclude the possibility of such failures with any certainty.

Even with these limitations, we hope that our modified method can function as a point of departure for other people who want to rigorously investigate their fair rankers, and that our approach may help others design improved failure analysis methods for amortized rankers.

6.4. Future work

There are a number of ideas we were unable to explore further in the interest of time.

Firstly, in the interest of time we investigated only two rankers. However, there were many interesting submissions to FAIR-TREC 2020 with different approaches. We cannot say for certain that the factors we identified are applicable to those rankers as well.

For LM, we only investigate a small number of potential accuracy improvements. There has been a lot of research throughout the years on improving the accuracy of LtR algorithms like LM. Since the focus on this thesis is on fairness, we only investigated a couple to see if accuracy improvements would increase the overall EEL. But one other obvious accuracy improvement would be to apply stemming and tokenization; the standard Elasticsearch analyzer doesn't do stemming and only basic tokenization.

Additionally, we saw that the improvements we tried did not improve the EEL-R, but we did not evaluate LM's performance in terms of traditional evaluation measures such as NDCG. We did not look at NDCG since we wanted to improve EEL, and there is no one-to-one interpretation of EEL-R as a user utility measure. Even so, it would be interesting to see whether a higher score on a user utility measure yields a better EEL as well.

Lastly, the post-processing methods we applied are similar to each other in their approach. They both re-compute a score for each document that balances expected exposure based on estimated probabilities of relevance with relevance or utility to the user. This allows us to compare them more easily, but it also means that we cannot see how they compare to a method that takes a different approach. Additionally, they are sensitive to the same issues.

One option for another post-processing method to investigate is Equity of Attention [4]. This method also looks at fairness across multiple rankings, in this case on an individual level. Their method is solving a linear programming problem that optimizes a certain fairness measure under utility constraints. It would be interesting to see how this method performs with rankings from LM as input.

For AC, when computing the expected target expected exposure $\hat{\mathcal{E}}^*$ (Equation (3.2)), Kletti and Renders [27] assume that each true relevance for a document $rel(d)$ is the realization of a draw from a Bernouilli distribution with as its parameter $\rho(d)$, the estimated probability of relevance. It would be interesting to investigate how well this assumption holds up, and whether other ways of computing the expected target expected exposure would yield better results.

As mentioned before, EEL is computed with respect to a specific grouping. The exact influence of the groups' composition is not fully clear. Different queries may be difficult according to different groupings. A more rigorous investigation of the influence of grouping on the outcome of EEL is left for future work. One option would be to generate many different random groupings and compute the performance over all of those [45].

The last point of future work pertains to our modified failure analysis. As mentioned in Section 6.3, our method is unable to detect more complex classes of

failures. One possible remedy may be to look at rankings from a more probabilistic point of view. Rather than assessing which documents end up in top positions most often in the absolute sense, it may be more useful to determine which factors increase the *probability* of documents to be ranked highly. This matches better with the kind of fairness measured by EEL.

References

[1] Waleed Ammar et al. "Construction of the Literature Graph in Semantic Scholar". In: *NAACL*. 2018. doi: 10.18653/v1/N18-3011.

[2] Marco Angelini et al. "VIRTUE: A Visual Tool for Information Retrieval Performance Evaluation and Failure Analysis". In: *Journal of Visual Languages & Computing* 25.4 (Aug. 1, 2014), pp. 394–413. issn: 1045-926X. doi: 10.1016/j.jvlc.2013.12.003. url: https://www.sciencedirect.com/science/article/pii/S1045926X13001006 (visited on 07/05/2022).

[3] B. Wayne Bequette. *Process Control: Modeling, Design, and Simulation*. Prentice Hall Professional, 2003. 804 pp. isbn: 978-0-13-353640-9. Google Books: PdjHYm5e9d4C.

[4] Asia J. Biega, Krishna P. Gummadi, and Gerhard Weikum. "Equity of Attention: Amortizing Individual Fairness in Rankings". In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. SIGIR '18. New York, NY, USA: Association for Computing Machinery, June 27, 2018, pp. 405–414. isbn: 978-1-4503-5657-2. doi: 10.1145/3209978.3210063. url: https://doi.org/10.1145/3209978.3210063 (visited on 05/12/2021).

[5] Asia J. Biega et al. "Overview of the TREC 2019 Fair Ranking Track". Mar. 25, 2020. arXiv: 2003.11650 [cs]. url: http://arxiv.org/abs/2003.11650 (visited on 04/14/2021).

[6] Asia J. Biega et al. "Overview of the TREC 2020 Fair Ranking Track". Aug. 11, 2021. arXiv: 2108.05135 [cs]. url: http://arxiv.org/abs/2108.05135 (visited on 11/15/2021).

[7] Malte Bonart. "Fair Ranking in Academic Search". In: *The Twenty-Eighth Text REtrieval Conference Proceedings (TREC 2019)* (), p. 3.

[8] Sebastian Bruch et al. "A Stochastic Treatment of Learning to Rank Scoring Functions". In: *Proceedings of the 13th International Conference on Web Search and Data Mining*. WSDM '20. New York, NY, USA: Association for Computing Machinery, Jan. 20, 2020, pp. 61–69. isbn: 978-1-4503-6822-3. doi: 10.1145/3336191.3371844. url: https://doi.org/10.1145/3336191.3371844 (visited on 05/12/2021).

[9] Chris Buckley. *Reliable Information Access Final Workshop Report*. Center, 2004.

[10] Christopher Burges et al. "Learning to Rank Using an Ensemble of Lambda-Gradient Models". In: *Proceedings of the Learning to Rank Challenge*. Proceedings of the Learning to Rank Challenge. PMLR, Jan. 26, 2011, pp. 25–35. url: https://proceedings.mlr.press/v14/burges11a.html (visited on 04/25/2022).

[11] Christopher J C Burges. *From RankNet to LambdaRank to LambdaMART: An Overview*. MSR-TR-2010-82. June 2010, p. 19. url: https://www.microsoft.com/en-us/research/publication/from-ranknet-to-lambdarank-to-lambdamart-an-overview/.

[12] Robin Burke. "Multisided Fairness for Recommendation". July 8, 2017. arXiv: 1707.00093 [cs]. url: http://arxiv.org/abs/1707.00093 (visited on 05/27/2021).

[13] Ben Carterette. "Statistical Significance Testing In Theory and In Practice" (Delaware). url: http://ir.cis.udel.edu/ICTIR13tutorial/slides.pdf (visited on 07/08/2022).

[14] Olivier Chapelle et al. "Expected Reciprocal Rank for Graded Relevance". In: *Proceeding of the 18th ACM Conference on Information and Knowledge Management - CIKM '09*. Proceeding of the 18th ACM Conference. Hong Kong, China: ACM Press, 2009, p. 621. isbn: 978-1-60558-512-3. doi: 10.1145/1645953.1646033. url: http://portal.acm.org/citation.cfm?doid=1645953.1646033 (visited on 01/25/2022).

[15] Jacob Cohen. *Statistical Power Analysis for the Behavioral Sciences*. 2nd ed. New York: Routledge, July 1, 1988. 567 pp. isbn: 978-0-203-77158-7. doi: 10.4324/9780203771587.

[16] Nick Craswell et al. "An Experimental Comparison of Click Position-Bias Models". In: *Proceedings of the 2008 International Conference on Web Search and Data Mining*. WSDM '08. New York, NY, USA: Association for Computing Machinery, Feb. 11, 2008, pp. 87–94. isbn: 978-1-59593-927-2. doi: 10.1145/1341531.1341545. url: https://doi.org/10.1145/1341531.1341545 (visited on 06/28/2021).

[17] W. Bruce Croft, Donald Metzler, and Trevor Strohman. *Search Engines - Information Retrieval in Practice*. 2015.

[18] Jeffrey Dastin. *Amazon Scraps Secret AI Recruiting Tool That Showed Bias against Women*. Reuters. Oct. 11, 2018. url: https://www.reuters.com/article/us-amazon-com-jobs-automation-insight-idUSKCN1MK08G (visited on 07/05/2022).

[19] Fernando Diaz. "Fairness Track Overview". url: https://trec.nist.gov/pubs/trec29/videos/Mon5.1%20Fairness%20Track%20Overview%20(Fernando%20DIaz).mp4 (visited on 01/31/2022).

[20] Fernando Diaz et al. "Evaluating Stochastic Rankings with Expected Exposure". In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. CIKM '20. New York, NY, USA: Association for Computing Machinery, Oct. 19, 2020, pp. 275–284. isbn: 978-1-4503-6859-9. doi: 10.1145/3340531.3411962. url: https://doi.org/10.1145/3340531.3411962 (visited on 05/03/2021).

[21] Kaweh Djafari Naini and Ismail Sengor Altingovde. "Exploiting Result Diversification Methods for Feature Selection in Learning to Rank". In: *Advances in Information Retrieval*. Ed. by Maarten de Rijke et al. Vol. 8416. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014, pp. 455–461. url: http://link.springer.com/10.1007/978-3-319-06028-6_41 (visited on 04/20/2022).

[22] Pinar Donmez, Krysta M. Svore, and Christopher J.C. Burges. "On the Local Optimality of LambdaRank". In: *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '09*. The 32nd International ACM SIGIR Conference. Boston, MA, USA: ACM Press, 2009, p. 460. isbn: 978-1-60558-483-6. doi: 10.1145/1571941.1572021. url: http://portal.acm.org/citation.cfm?doid=1571941.1572021 (visited on 07/13/2022).

[23] Michael D Ekstrand et al. "Overview of the TREC 2021 Fair Ranking Track". In: (), p. 42.

[24] Yunhe Feng et al. *University of Washington at TREC 2020 Fairness Ranking Track*. 2020.

[25] Andres Ferraro, Lorenzo Porcaro, and Xavier Serra. "Balancing Exposure and Relevance in Academic Search". In: (), p. 2.

[26] Xiubo Geng et al. "Feature Selection for Ranking". In: *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '07*. The 30th Annual International ACM SIGIR Conference. Amsterdam, The Netherlands: ACM Press, 2007, p. 407. isbn: 978-1-59593-597-7. doi: 10.1145/1277741.1277811. url: http://portal.acm.org/citation.cfm?doid=1277741.1277811 (visited on 04/20/2022).

[27] Till Kletti and Jean-Michel Renders. "Naver Labs Europe at TREC 2020 Fair Ranking Track". In: (), p. 4.

[28] *Listings of WHO's Response to COVID-19*. Jan. 29, 2021. url: https://www.who.int/news/item/29-06-2020-covidtimeline (visited on 07/14/2022).

[29] Tie-Yan Liu. "Learning to Rank for Information Retrieval". In: *Foundations and Trends® in Information Retrieval* 3.3 (June 26, 2009), pp. 225–331. issn: 1554-0669, 1554-0677. doi: 10.1561/1500000016. url: https://www.nowpublishers.com/article/Details/INR-016 (visited on 02/08/2022).

[30] Christopher Manning, Prabhakar Raghavan, and Hinrich Schuetze. "Introduction to Information Retrieval". In: (2009), p. 581.

[31] M. E. Maron and J. L. Kuhns. "On Relevance, Probabilistic Indexing and Information Retrieval". In: *Journal of the ACM* 7.3 (July 1, 1960), pp. 216–244. issn: 0004-5411. doi: 10.1145/321033.321035. url: https://doi.org/10.1145/321033.321035 (visited on 07/05/2022).

[32] Marco Morik et al. "Controlling Fairness and Bias in Dynamic Learning-to-Rank". In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '20. New York, NY, USA: Association for Computing Machinery, July 25, 2020, pp. 429–438. isbn: 978-1-4503-8016-4. doi: 10.1145/3397271.3401100. url: https://doi.org/10.1145/3397271.3401100 (visited on 05/20/2021).

[33] Harikrishna Narasimhan et al. "Pairwise Fairness for Ranking and Regression". Jan. 7, 2020. arXiv: 1906.05330 [cs, stat]. url: http://arxiv.org/abs/1906.05330 (visited on 07/13/2021).

[34] Stephen Olejnik and James Algina. "Measures of Effect Size for Comparative Studies: Applications, Interpretations, and Limitations". In: *Contemporary Educational Psychology* 25.3 (July 2000), pp. 241–286. issn: 0361476X. doi: 10.1006/ceps.2000.1040. url: https://linkinghub.elsevier.com/retrieve/pii/S0361476X00910403 (visited on 06/01/2022).

[35] Tao Qin and Tie-Yan Liu. *Introducing LETOR 4.0 Datasets*. June 9, 2013. doi: 10.48550/arXiv.1306.2597. arXiv: 1306.2597 [cs]. url: http://arxiv.org/abs/1306.2597 (visited on 07/13/2022).

[36] Jean-Michel Renders. *Code to Reproduce Your TREC 2020 Fair Ranking Track Runs*. In collab. with Maaike Visser. E-mail. Jan. 17, 2022.

[37] S.E. Robertson. "THE PROBABILITY RANKING PRINCIPLE IN IR". In: *Journal of Documentation* 33.4 (Apr. 1, 1977), pp. 294–304. issn: 0022-0418. doi: 10.1108/eb026647. url: https://www.emerald.com/insight/content/doi/10.1108/eb026647/full/html (visited on 07/04/2022).

[38] Tetsuya Sakai. "Statistical Reform in Information Retrieval?" In: *ACM SIGIR Forum* 48.1 (June 26, 2014), pp. 3–12. issn: 0163-5840. doi: 10.1145/2641383.2641385. url: https://dl.acm.org/doi/10.1145/2641383.2641385 (visited on 05/30/2022).

[39] Jacques Savoy. "Why Do Successful Search Systems Fail for Some Topics". In: *Proceedings of the 2007 ACM Symposium on Applied Computing - SAC '07*. The 2007 ACM Symposium. Seoul, Korea: ACM Press, 2007, p. 872. isbn: 978-1-59593-480-2. doi: 10.1145/1244002.1244193. url: http://portal.acm.org/citation.cfm?doid=1244002.1244193 (visited on 07/05/2022).

[40] Mahmoud F Sayed and Douglas W Oard. "The University of Maryland at the TREC 2020 Fair Ranking Track". In: (), p. 4.

[41] Ashudeep Singh and Thorsten Joachims. "Equality of Opportunity in Rankings". In: (Dec. 2017), p. 3.

[42] Ashudeep Singh and Thorsten Joachims. "Fairness of Exposure in Rankings". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD '18. New York, NY, USA: Association for Computing Machinery, July 19, 2018, pp. 2219–2228. isbn: 978-1-4503-5552-0. doi: 10.1145/3219819.3220088. url: https://doi.org/10.1145/3219819.3220088 (visited on 05/14/2021).

[43] Student. "The probable error of a mean". In: *Biometrika* (1908), pp. 1–25.

[44] Niek Tax, Sander Bockting, and Djoerd Hiemstra. "A Cross-Benchmark Comparison of 87 Learning to Rank Methods". In: *Information Processing & Management* 51.6 (Nov. 1, 2015), pp. 757–772. issn: 0306-4573. doi: 10.1016/j.ipm.2015.07.002. url: https://www.sciencedirect.com/science/article/pii/S0306457315000862 (visited on 07/26/2022).

[45] Thibaut Thonet and Jean-Michel Renders. "Multi-Grouping Robust Fair Ranking". In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '20. New York, NY, USA: Association for Computing Machinery, July 25, 2020, pp. 2077–2080. isbn: 978-1-4503-8016-4. doi: 10.1145/3397271.3401292. url: https://doi.org/10.1145/3397271.3401292 (visited on 04/28/2021).

[46] Julián Urbano, Harlley Lima, and Alan Hanjalic. *Statistical Significance Testing in Information Retrieval: An Empirical Analysis of Type I, Type II and Type III Errors*. June 5, 2019. doi: 10.1145/3331184.3331259. arXiv: 1905.11096 [cs, stat]. url: http://arxiv.org/abs/1905.11096 (visited on 05/30/2022).

[47] E. Voorhees, D. K. Harman, and National Institute of Standards and Technology (U.S.), eds. *TREC: Experiment and Evaluation in Information Retrieval*. Digital Libraries and Electronic Publishing. Cambridge, Mass: MIT Press, 2005. 462 pp. isbn: 978-0-262-22073-6.

[48] Ellen M. Voorhees. "The TREC-8 Question Answering Track Report". In: ().

[49] Xuanhui Wang et al. "The LambdaLoss Framework for Ranking Metric Optimization". In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. CIKM '18: The 27th ACM International Conference on Information and Knowledge Management. Torino Italy: ACM, Oct. 17, 2018, pp. 1313–1322. isbn: 978-1-4503-6014-2. doi: 10.1145/3269206.3271784. url: https://dl.acm.org/doi/10.1145/3269206.3271784 (visited on 07/11/2022).

[50] Bianca Zadrozny and Charles Elkan. "Transforming Classifier Scores into Accurate Multiclass Probability Estimates". In: (), p. 6.

[51] Meike Zehlike, Ke Yang, and Julia Stoyanovich. "Fairness in Ranking: A Survey". May 12, 2021. arXiv: 2103.14000 [cs]. url: http://arxiv.org/abs/2103.14000 (visited on 05/19/2021).

[52] Meike Zehlike et al. "FA*IR: A Fair Top-k Ranking Algorithm". In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. CIKM '17. New York, NY, USA: Association for Computing Machinery, Nov. 6, 2017, pp. 1569–1578. isbn: 978-1-4503-4918-5. doi: 10.1145/3132847.3132938. url: https://doi.org/10.1145/3132847.3132938 (visited on 05/03/2021).

A

Changes relative to the offical FAIR-TREC instructions

A.1. A note on retrieving the 2019 corpus

The corpora for FAIR-TREC2019 and FAIR-TREC2020 are both snapshots of the Semantic Scholar (S2) Open Corpus from the Allen Institute for Artificial Intelligence [1]. Versions of the corpus can be downloaded through api.semanticscholar.org/corpus/download. However, at the time of writing the version of the corpus as it must have been at the time that FAIR-TREC2019 was running was no longer available on that site.

To obtain the corpus snapshot, we used the Wayback Machine[1] to go back to the snapshot of 16 Aug 2019. Then we ran

```
aws s3 cp --no-sign-request --recursive s3://ai2-s2-research
    ↪ -public/open-corpus/<date>/manifest.txt .
```

for each date counting back from 16-08-2019 until a download resolved. This meant that we had found the most recently available version at the time of the 2019 track, which was the version of the corpus on 31-01-2019. We then downloaded the rest of the files with

```
aws s3 cp --no-sign-request --recursive s3://ai2-s2-research
    ↪ -public/open-corpus/2019-01-31/ .
```

A.2. Changes to evaluation procedure

FAIR-TREC2020 was accompanied by a repository with tools to evaluate system runs [2]. To reproduce the values as reported in [6] we make a number of changes with respect to the provided code and instructions.

[1] archive.org/web/

[2] github.com/fair-trec/fair-trec-tools/tree/master/eval

- The evaluation script requires each document to be mapped to a numerical group identifier. We originally interpreted this to mean that a "Mixed" document should have two group identifiers, one for "Developing" and one for "Advanced". However, we found that to reproduce the reported results, "Mixed" papers had to be mapped to a separate group identifier altogether. We created the group mapping based on `merged-annotations.json`[3] as indicated by the organisers[4].
- We modified the last line in expeval.sh from `./expeval.py -I runfile.tsv -R qrels.tsv -G -C` to `./expeval.py qrels.tsv runfile.tsv -G -C -U`.
- We set `sqrt=False` in the `distance` method.

[3] Click "TREC 2020 Fair Ranking Track evaluation data" on `fair-trec.github.io/2020/`
[4] `groups.google.com/g/fair-trec/c/OQPkqXxKiIw`

B

Statistical significance test results

In this appendix we give the results for the various statistical significant tests we performed to assess the performance of the improvements in in Chapter 5. The structure of this chapter follows that of Chapter 5.

B.1. Improvements to LambdaMART

B.1.1. Improving disparity with post-processing

	$\mathcal{G}_i$			$\mathcal{G}_h$		
Level	EEL	EEL-D	EEL-R	EEL	EEL-D	EEL-R
`none`*	1.378	1.239	**0.208**	0.762	**2.154**	**1.243**
`ac`	**0.459**	**0.233**	0.165	**0.409**	**1.615**	1.150
`rfre`	0.554	0.301	0.151	0.495	1.656	1.127

Table B.1: Mean EEL($\mathcal{G}_i, \Pi$) and EEL($\mathcal{G}_h, \Pi$) of each level of factor `post` corresponding to LM improvement `ILM_post` on `corp2020val_split`. Best performances are bolded. The baseline level is marked with $*$.

Level	$\mathcal{G}_i$ EEL	$\mathcal{G}_i$ EEL-D	$\mathcal{G}_i$ EEL-R	$\mathcal{G}_e$ EEL	$\mathcal{G}_e$ EEL-D	$\mathcal{G}_e$ EEL-R
none*	1.422	1.248	**0.215**	0.855	2.209	**1.297**
ac	**0.577**	**0.336**	0.181	**0.437**	**1.863**	1.334
rfre	0.612	**0.336**	0.164	0.450	1.872	1.332

Table B.2: Mean EEL($\mathcal{G}_i, \Pi$) and EEL($\mathcal{G}_e, \Pi$) of each level of factor `post` corresponding to LM improvement `ILM_post` on `corp2020test`. Best performances are bolded. The baseline level is marked with $*$.

Factor	df	SS	MS	F	p	ω^2	$\omega^2_{partial}$
post†	2	10	5.1	53	0.0	0.54	0.84
qid†	19	4.7	0.25	2.6	0.0059	0.16	0.91
error	38	3.6	0.095	-	-	-	-

(a) DV: EEL($\mathcal{G}_i, \Pi$)

Factor	df	SS	MS	F	p	ω^2	$\omega^2_{partial}$
post†	2	1.35	0.68	9.56	0.0	0.05	0.46
qid†	19	18.16	0.96	13.49	0.0	0.76	0.99
error	38	2.69	0.07	-	-	-	-

(b) DV: EEL($\mathcal{G}_h, \Pi$)

Table B.3: Results of a two-way ANOVA with factors `post` (corresponding to LM improvement `ILM_post`) and `qid`, dependent variables EEL($\mathcal{G}_i, \Pi$) and EEL($\mathcal{G}_h, \Pi$), and dataset `corp2020val_split`. Factors marked with † are statistically significant.

Level 1	Level 2	$\bar{d}$	p	d_c
none*	ac	−0.92	0.0†	3.0
none*	rfre	−0.82	0.0†	2.7
ac	rfre	0.095	0.71	0.31

(a) DV: EEL($\mathcal{G}_i, \Pi$)

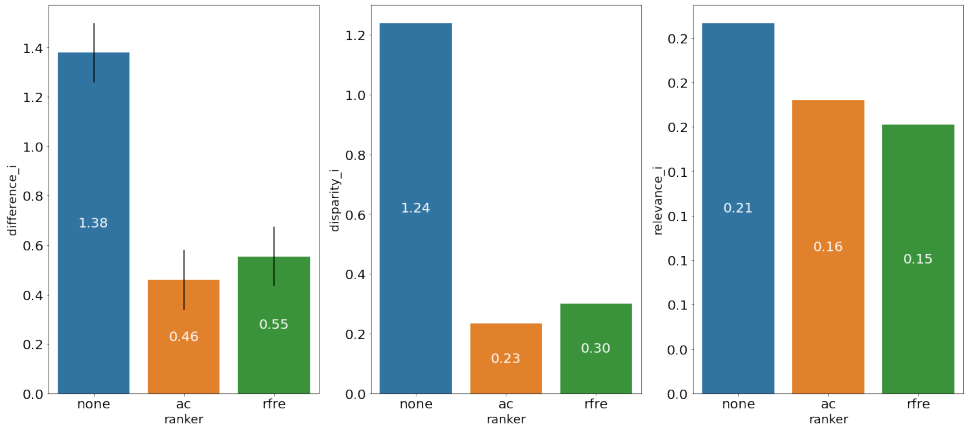
Level 1	Level 2	$\bar{d}$	p	d_c
none*	ac	−0.35	0.16	1.3
none*	rfre	−0.27	0.35	1.0
ac	rfre	0.086	0.89	0.32

(b) DV: EEL($\mathcal{G}_h, \Pi$)

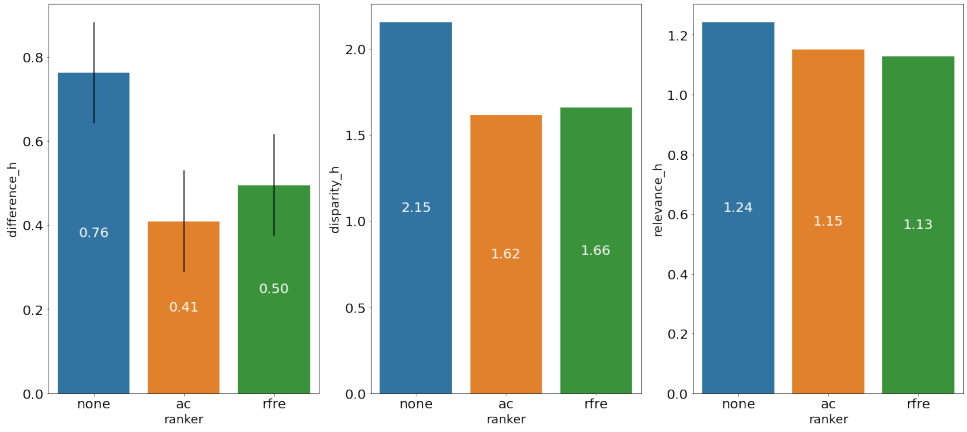
Table B.4: Results of a Tukey's HSD test for the levels of factor `post` corresponding to LM improvement `ILM_post`, dependent variables EEL($\mathcal{G}_i, \Pi$) and EEL($\mathcal{G}_h, \Pi$), and dataset `corp2020val_split`. p-values marked with † are statistically significant. The baseline level is marked with $*$.

Level	DV	df	$\bar{d}$	V	T	p	CI	ES_t
ac	EEL($\mathcal{G}_i, \Pi$)	199	−0.85	0.22	25	0.0†	[−0.91, −0.78]	1.79
	EEL($\mathcal{G}_e, \Pi$)	199	−0.42	0.53	8.1	0.0†	[−0.52, −0.32]	0.57
rfre	EEL($\mathcal{G}_i, \Pi$)	199	−0.81	0.27	22	0.0†	[−0.88, −0.74]	1.55
	EEL($\mathcal{G}_e, \Pi$)	199	−0.41	0.70	6.9	0.0†	[−0.52, −0.29]	0.49

Table B.5: Results of t-tests between levels `ac` and `rfre` of factor `post` (corresponding to LM improvement `ILM_post`) and baseline level `none` with dependent variables EEL($\mathcal{G}_i, \Pi$) and EEL($\mathcal{G}_e, \Pi$) and dataset `corp2020test`. p-values marked with † are statistically significant.



(a) DV: EEL($\mathcal{G}_i, \Pi$)



(b) DV: EEL($\mathcal{G}_h, \Pi$)

Figure B.1: Mean EEL($\mathcal{G}_i, \Pi$) and EEL($\mathcal{G}_h, \Pi$) of each level of factor `post` corresponding to LM improvement `ILM_post` on `corp2020val_split`. Error bars are computed as in Equation (5.1).

B.1.2. Improving relevance with feature selection

	$\mathcal{G}_i$			$\mathcal{G}_h$		
Level	EEL	EEL-D	EEL-R	EEL	EEL-D	EEL-R
none*	1.378	1.239	0.208	0.762	2.154	1.243
mpt	**1.286**	**1.234**	**0.252**	**0.696**	**1.902**	1.149
msd	1.371	1.251	0.218	0.840	2.287	**1.270**

Table B.6: Mean EEL($\mathcal{G}_i, \Pi$) and EEL($\mathcal{G}_h, \Pi$) of each level of factor `fsm` corresponding to LM improvement `ILM_fsm` on `corp2020val_split`. Best performances are bolded. The baseline level is marked with $*$.

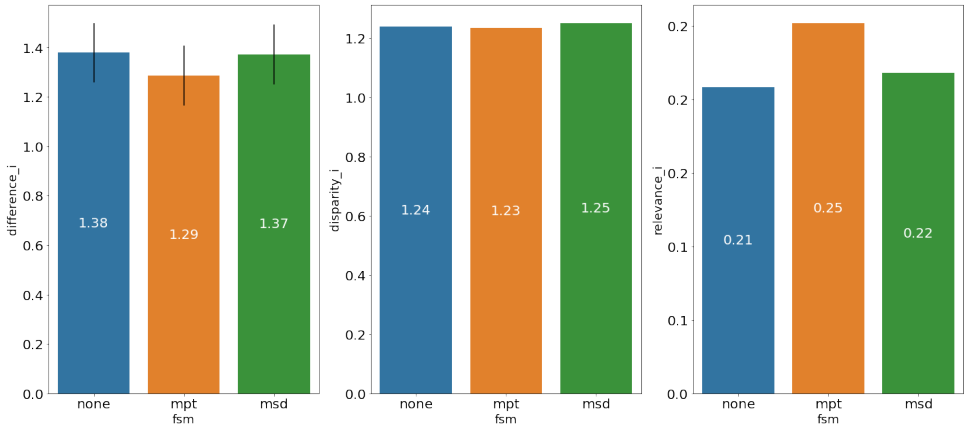
Factor	df	SS	MS	F	p	ω^2	$\omega^2_{partial}$
`fs`	2	0.11	0.05	0.28	0.76	−0.02	−0.08
`qid`†	19	9.7	0.51	2.7	0.0	0.36	0.91
`error`	38	7.2	0.19	-	-	-	-

(a) DV: EEL($\mathcal{G}_i, \Pi$)

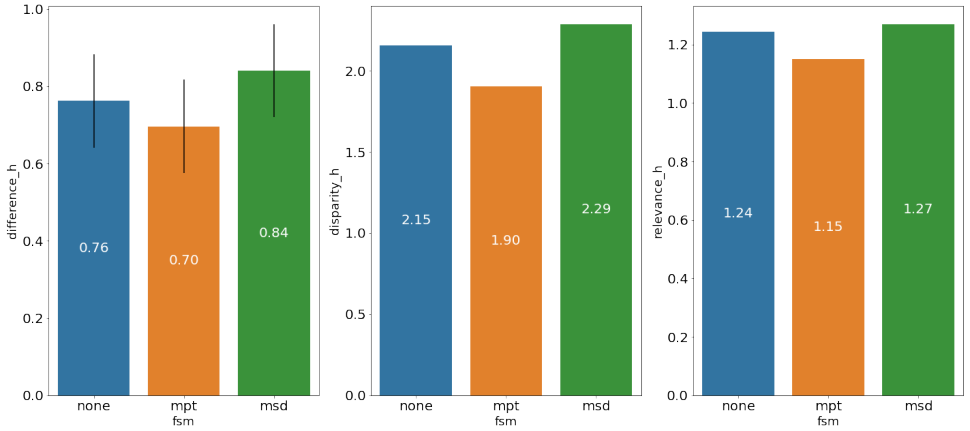
Factor	df	SS	MS	F	p	ω^2	$\omega^2_{partial}$
`fs`	2	0.21	0.10	0.36	0.70	−0.01	−0.07
`qid`†	19	16	0.86	3.0	0.0	0.4	0.93
`error`	38	11	0.29	-	-	-	-

(b) DV: EEL($\mathcal{G}_h, \Pi$)

Table B.7: Results of a two-way ANOVA with factors `fs` (corresponding to LM improvement `ILM_fsm`) and `qid`, dependent variables EEL($\mathcal{G}_i, \Pi$) and EEL($\mathcal{G}_h, \Pi$), and dataset `corp2020val_split`. Factors marked with † are statistically significant.



(a) DV: EEL($\mathcal{G}_i, \Pi$)



(b) DV: EEL($\mathcal{G}_h, \Pi$)

Figure B.2: Mean EEL($\mathcal{G}_i, \Pi$) and EEL($\mathcal{G}_h, \Pi$) of each level of factor `fs` corresponding to improvement `ILM_fsm` on `corp2020val_split`. Error bars are computed as in Equation (5.1).

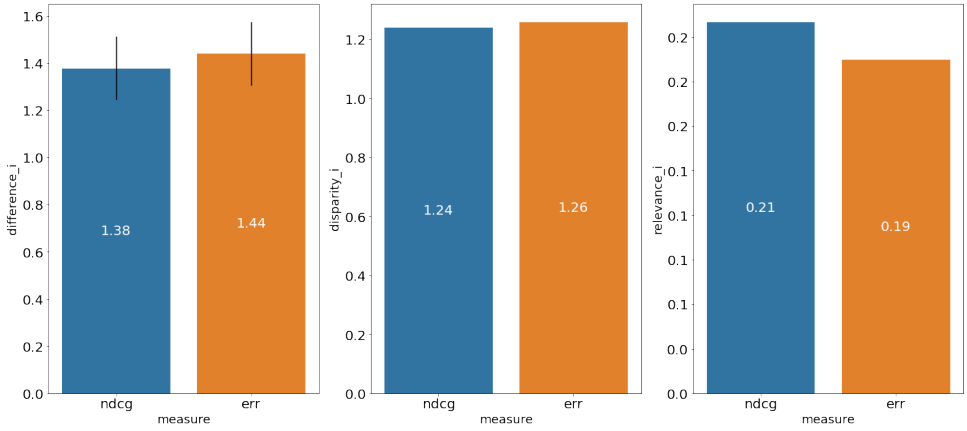
B.1.3. Improving relevance through choice of effectiveness measure

	$\mathcal{G}_i$			$\mathcal{G}_h$		
Level	EEL	EEL-D	EEL-R	EEL	EEL-D	EEL-R
`ndcg`*	**1.378**	**1.239**	**0.208**	**0.762**	**2.154**	**1.243**
`err`	1.440	1.259	0.187	0.901	2.313	1.253

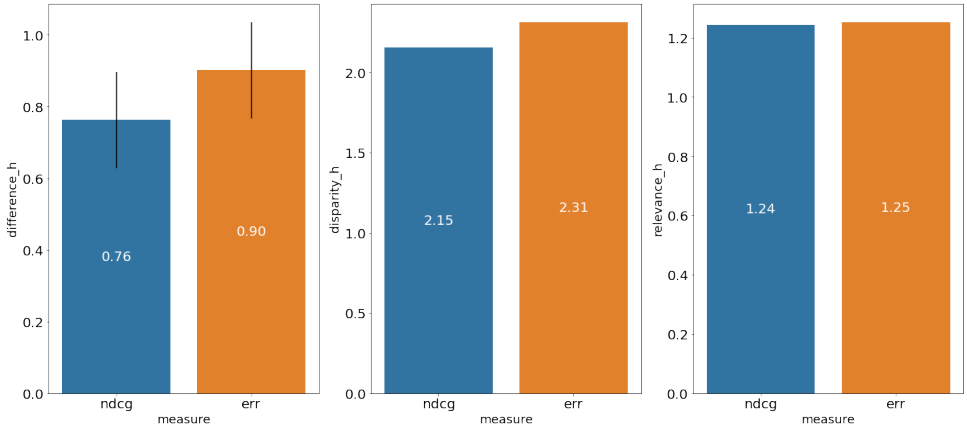
Table B.8: Mean EEL($\mathcal{G}_i, \Pi$) and EEL($\mathcal{G}_h, \Pi$) of each level of factor `measure` corresponding to LM improvement `ILM_measure` on `corp2020val_split`. Best performances are bolded. The baseline level is marked with $*$.

DV	df	$\bar{d}$	V	T	p	CI	ES_t
EEL$(\mathcal{G}_i, \Pi)$	19	0.062	0.090	-0.92	0.37	$[-0.08, 0.20]$	0.21
EEL$(\mathcal{G}_h, \Pi)$	19	0.14	0.16	-1.5	0.14	$[-0.05, 0.33]$	0.34

Table B.9: Results of t-tests between levels `ndcg` (baseline) and `err` of factor `measure` corresponding to LM improvement `ILM_measure` with dependent variables EEL$(\mathcal{G}_i, \Pi)$ and EEL$(\mathcal{G}_h, \Pi)$ and dataset `corp2020val_split`. p-values marked with † are statistically significant.



(a) DV: EEL$(\mathcal{G}_i, \Pi)$



(b) DV: EEL$(\mathcal{G}_h, \Pi)$

Figure B.3: Mean EEL$(\mathcal{G}_i, \Pi)$ and EEL$(\mathcal{G}_h, \Pi)$ of each level of factor `measure` corresponding to LM improvement `ILM_measure` on `corp2020val_split`. Error bars are computed as in Equation (5.3).

B.2. Improvements to Advantage Controller

B.2.1. Remedying missing data with dummy authors

	$\mathcal{G}_i$			$\mathcal{G}_h$		
Level	EEL	EEL-D	EEL-R	EEL	EEL-D	EEL-R
none*	0.515	0.238	0.139	0.498	1.653	**1.124**
ind	0.449	**0.186**	0.146	0.423	1.569	1.120
one	**0.448**	0.187	**0.147**	**0.418**	**1.565**	1.120

Table B.10: Mean EEL($\mathcal{G}_i, \Pi$) and EEL($\mathcal{G}_h, \Pi$) of each level of factor `dummy` corresponding to AC improvement `IAC_dummy` on `corp2020val_split`. Best performances are bolded. The baseline level is marked with $*$.

Factor	df	SS	MS	F	p	ω^2	$\omega^2_{partial}$
dummy†	2	0.060	0.030	7.3	0.0	0.020	0.39
qid†	19	2.9	0.15	38	0.0	0.91	1.0
error	38	0.15	0.0	-	-	-	-

(a) DV: EEL($\mathcal{G}_i, \Pi$)

Factor	df	SS	MS	F	p	ω^2	$\omega^2_{partial}$
dummy†	2	0.080	0.040	4.2	0.020	0.0	0.24
qid†	19	17	0.87	91	0.0	0.96	1.0
error	38	0.36	0.010	-	-	-	-

(b) DV: EEL($\mathcal{G}_h, \Pi$)

Table B.11: Results of a two-way ANOVA with factors `dummy` (corresponding to AC improvement `IAC_dummy`) and `qid`, dependent variables EEL($\mathcal{G}_i, \Pi$) and EEL($\mathcal{G}_h, \Pi$), and dataset `corp2020val_split`. Factors marked with † are statistically significant.

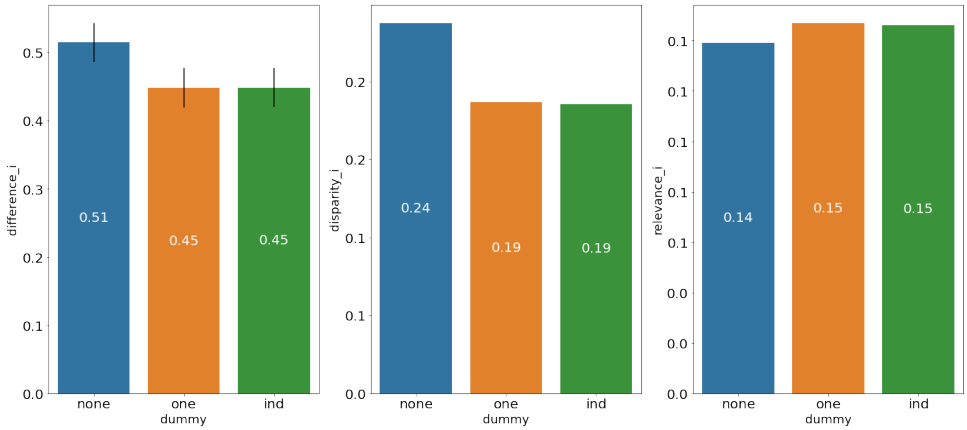
Level 1	Level 2	$\bar{d}$	p	d_c
none*	ind	−0.066	0.64	0.68
none*	one	−0.0665	0.64	0.68
ind	one	−0.00040	1.0	0.0044

(a) DV: EEL($\mathcal{G}_i, \Pi$)

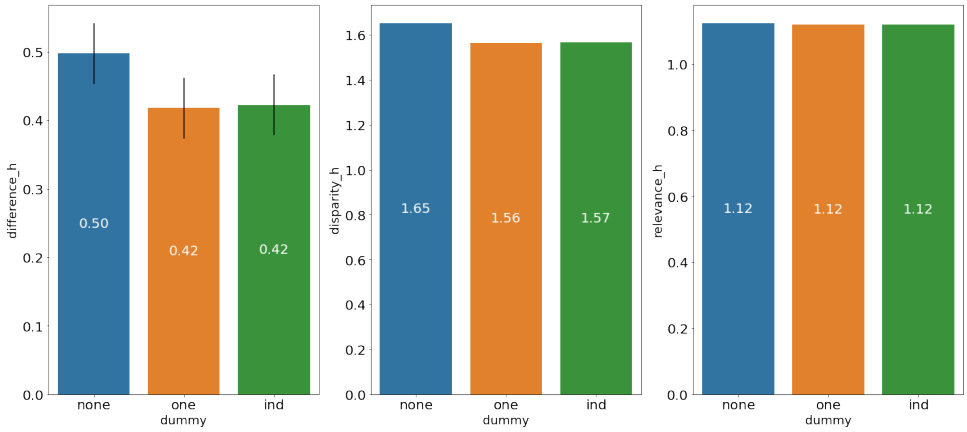
Level 1	Level 2	$\bar{d}$	p	d_c
none*	ind	−0.075	0.90	0.82
none*	one	−0.080	0.89	0.77
ind	one	−0.0045	1.00	0.046

(b) DV: EEL($\mathcal{G}_h, \Pi$)

Table B.12: Results of a Tukey's HSD test for the levels of factor `dummy` corresponding to AC improvement `IAC_dummy`, dependent variables EEL($\mathcal{G}_i, \Pi$) and EEL($\mathcal{G}_h, \Pi$), and dataset `corp2020val_split`. p-values marked with † are statistically significant. The baseline level is marked with $*$.



(a) DV: EEL($\mathcal{G}_i, \Pi$)



(b) DV: EEL($\mathcal{G}_h, \Pi$)

Figure B.4: Mean EEL($\mathcal{G}_i, \Pi$) and EEL($\mathcal{G}_h, \Pi$) of each level of factor `dummy` corresponding to AC improvement `IAC_dummy` on `corp2020val_split`. Error bars are computed as in Equation (5.1).

B.2.2. Remedying unequal advantages with the scoring function

	$\mathcal{G}_i$			$\mathcal{G}_h$		
Level	EEL	EEL-D	EEL-R	EEL	EEL-D	EEL-R
`linear`*	0.515	0.238	0.139	0.498	1.653	1.124
`max`	0.555	0.282	0.141	0.505	1.690	1.140
`min`	0.788	0.561	0.164	0.590	1.685	1.094

Table B.13: Mean EEL($\mathcal{G}_i, \Pi$) and EEL($\mathcal{G}_h, \Pi$) of each level of factor `hfunc` corresponding to AC improvement `IAC_hfunc` on `corp2020val_split`. Best performances are bolded. The baseline level is marked with $*$.

Factor	df	SS	MS	F	p	ω^2	$\omega^2_{partial}$
`hfunc`†	2	0.87	0.43	24	0.0	0.15	0.70
`qid`†	19	4.1	0.21	12	0.0	0.66	0.99
`error`	38	0.69	0.020	-	-	-	-

(a) DV: EEL($\mathcal{G}_i, \Pi$)

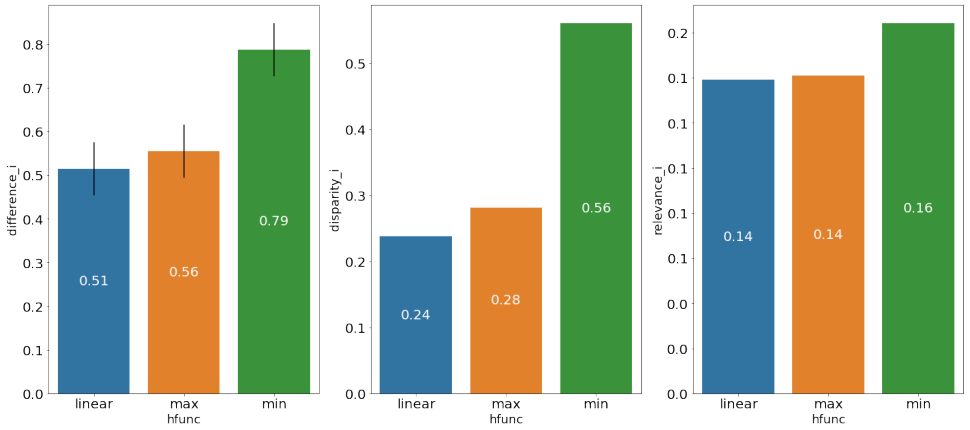
Factor	df	SS	MS	F	p	ω^2	$\omega^2_{partial}$
`hfunc`	2	0.11	0.050	1.2	0.31	0.0	0.020
`qid`†	19	18	1.0	22	0.0	0.87	0.99
`error`	38	1.7	0.040	-	-	-	-

(b) DV: EEL($\mathcal{G}_h, \Pi$)

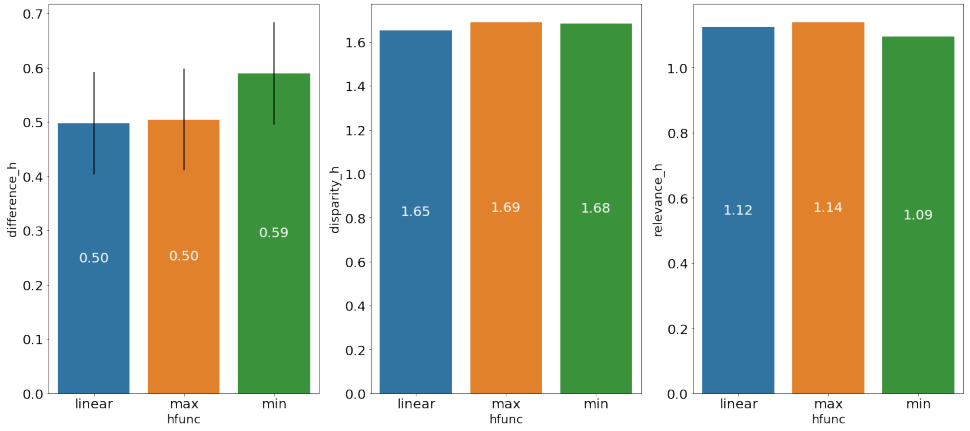
Table B.14: Results of a two-way ANOVA with factors `hfunc` (corresponding to AC improvement `IAC_hfunc`) and `qid`, dependent variables EEL($\mathcal{G}_i, \Pi$) and EEL($\mathcal{G}_h, \Pi$), and dataset `corp2020val_split`. Factors marked with † are statistically significant.

Level 1	Level 2	$\bar{d}$	p	d_c
`linear`*	`max`	0.040	0.90	0.30
`linear`*	`min`	0.27	0.011†	2.0
`max`	`min`	0.23	0.036†	1.7

Table B.15: Results of a Tukey's HSD test for the levels of factor `hfunc` corresponding to AC improvement `IAC_hfunc`, dependent variables EEL($\mathcal{G}_i, \Pi$) and EEL($\mathcal{G}_h, \Pi$), and dataset `corp2020val_split`. p-values marked with † are statistically significant. The baseline level is marked with $*$.



(a) DV: EEL($\mathcal{G}_i, \Pi$)



(b) DV: EEL($\mathcal{G}_h, \Pi$)

Figure B.5: Mean EEL($\mathcal{G}_i, \Pi$) and EEL($\mathcal{G}_h, \Pi$) of each level of factor `hfunc` corresponding to AC improvement `IAC_hfunc` on `corp2020val_split`. Error bars are computed as in Equation (5.1).

B.2.3. Improving performance with more accurate relevance predictions

Level	$\mathcal{G}_i$ EEL	$\mathcal{G}_i$ EEL-D	$\mathcal{G}_i$ EEL-R	$\mathcal{G}_h$ EEL	$\mathcal{G}_h$ EEL-D	$\mathcal{G}_h$ EEL-R
`rel_set_A`*	0.515	0.238	0.139	0.498	1.653	1.124
`rel_set_lm`	**0.459**	**0.233**	**0.165**	**0.409**	**1.615**	**1.150**

Table B.16: Mean EEL($\mathcal{G}_i, \Pi$) and EEL($\mathcal{G}_h, \Pi$) of each level of factor `rels` corresponding to AC improvement `IAC_rels` on `corp2020val_split`. Best performances are bolded. The baseline level is marked with $*$.

Level	$\mathcal{G}_i$ EEL	$\mathcal{G}_i$ EEL-D	$\mathcal{G}_i$ EEL-R	$\mathcal{G}_e$ EEL	$\mathcal{G}_e$ EEL-D	$\mathcal{G}_e$ EEL-R
rel_set_A*	0.572	0.270	0.151	0.429	1.901	**1.356**
rel_set_lm	**0.531**	**0.266**	**0.170**	**0.412**	**1.835**	1.332

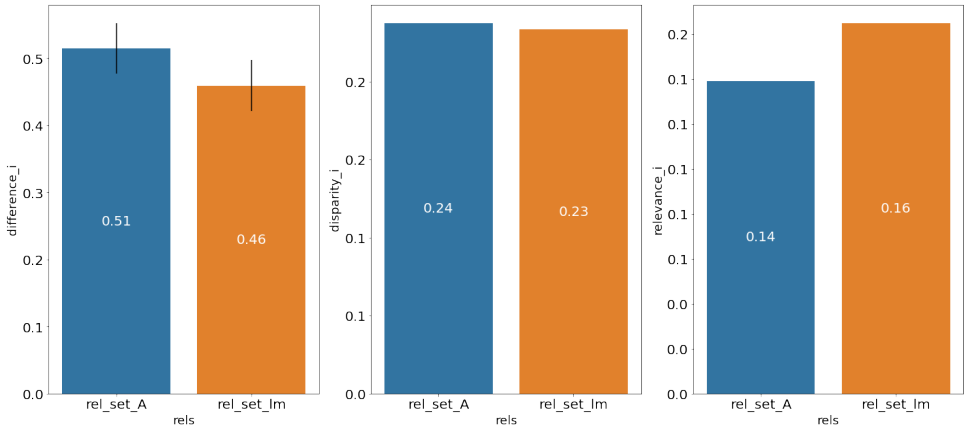
Table B.17: Mean EEL($\mathcal{G}_i, \Pi$) and EEL($\mathcal{G}_e, \Pi$) of each level of factor rels corresponding to AC improvement IAC_rels on corp2020test. Best performances are bolded. The baseline level is marked with $*$.

DV	df	$\bar{d}$	V	T	p	CI	ES_t
EEL($\mathcal{G}_i, \Pi$)	19	-0.056	0.013	2.2	$0.042^{\dagger}$	$[-0.11, 0.0]$	0.49
EEL($\mathcal{G}_h, \Pi$)	19	-0.089	0.085	1.4	0.19	$[-0.22, 0.050]$	0.31

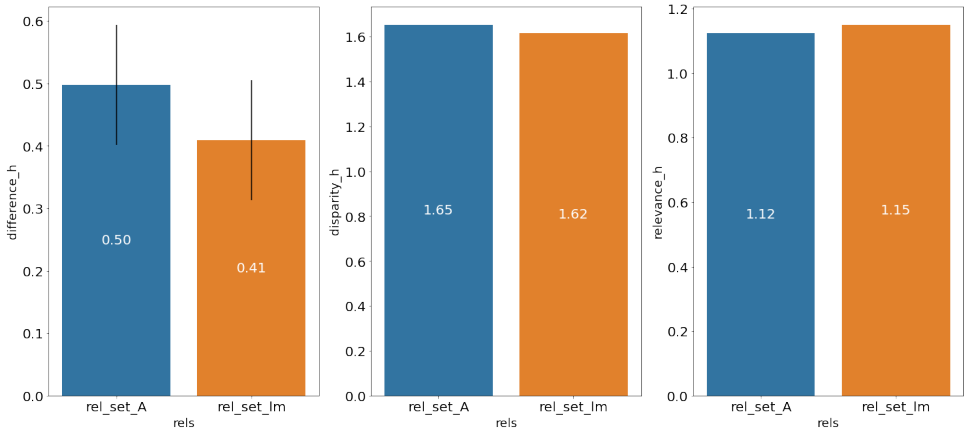
Table B.18: Results of t-tests between levels rel_set_A (baseline) and rel_set_lm of factor rels corresponding to AC improvement IAC_rels with dependent variables EEL($\mathcal{G}_i, \Pi$) and EEL($\mathcal{G}_h, \Pi$) and dataset corp2020val_split. p-values marked with † are statistically significant.

DV	df	$\bar{d}$	V	T	p	CI	ES_t
EEL($\mathcal{G}_i, \Pi$)	199	-0.85	0.22	25	$0.0^{\dagger}$	$[-0.91, -0.78]$	1.8
EEL($\mathcal{G}_e, \Pi$)	199	-0.42	0.54	8.0	$0.0^{\dagger}$	$[-0.52, -0.32]$	0.57

Table B.19: Results of t-tests between levels rel_set_A (baseline) and rel_set_lm of factor rels corresponding to AC improvement IAC_rels with dependent variables EEL($\mathcal{G}_i, \Pi$) and EEL($\mathcal{G}_h, \Pi$) and dataset corp2020test. p-values marked with † are statistically significant.



(a) DV: EEL($\mathcal{G}_i, \Pi$)



(b) DV: EEL($\mathcal{G}_h, \Pi$)

Figure B.6: Mean EEL($\mathcal{G}_i, \Pi$) and EEL($\mathcal{G}_h, \Pi$) of each level of factor `rels` corresponding to improvement `IAC_rels` on `corp2020val_split`. Error bars are computed as in Equation (5.3).