CO*IR: A Greedy and Individually Fair Re-ranker

Seán Healy

ADAPT Centre (Dublin City University), Ireland

Abstract

An open-source and *greedy* approach to individually fair re-ranking is presented, evaluated and tested. Previous literature on individually fair search suggested greedy-style heuristics, but no such designs or implementations were introduced before this writing. We release our re-ranker as a zero-dependency utility compatible with all major operating systems.

We publish our re-ranker under a permissive software license (GPLv3). By explicitly considering individual fairness a post-processing (re-ranking) task, we implement the notion of individual fairness as a microservice, de-coupled from specific IR systems. Our software package works on commodity hardware, and finds application in small-to-medium-sized businesses, federated social networks, and under the broader open web search initiative.

Keywords

fairness, ranking, individual fairness, open source

1. Introduction

Individual fairness is a distributive justice consideration within information retrieval, first discussed in the context of sociotechnical systems by Dwork et al. [1] (*"similar individuals treated similarly"*). The idea follows: rather than collecting group labels for items, and optimising for equity with respect to these group labels, individual fairness aims to allocate exposure [2] (or perhaps impact/outcomes) equitably to *relevance* or utility, on the level of the individual. Biega notes that positional bias, stemming from limited screen space and user attention, means the only way to do this must involve multi-user repetition of equivalent information needs [3, p. 33]. Each user is served 'perturbed' rankings, $\tilde{\rho}_j$, as opposed to 'canonical' rankings (ρ), with a goal of equalising *exposure : relevance* among items (Figure 1). This method has been applied in both individual and group-fair scenarios. However, the methods so far have often assumed significant computational power, or alternately, reliable access to group labels, leading to another injustice in the form of *barrier to entry and centralisation.* The original proposal for the world wide web emphasised non-centralisation:

> "Information systems start small and grow. They also start isolated and then merge. A new system must allow existing systems to be linked together without requiring any central control or coordination." [4]

In assuming an abundance of accurate group label data, or an abundance of centralised computing power, we inherently support a *winner takes all* view of the WWW [5], where isolated systems have ultimately merged into monopolistic ones with access to highly centralised computing resources and accurate group labels for subjects and searchers. Within fairness research, we often assume this state-of-affairs, in an effort to reduce *winner takes all* effects among search subjects.

Ignoring computational constraints and data access issues limits scope, and focuses research effort on the development and validation of theoretical methods. However, we argue that it is possible to re-imagine these fairness processes for a *non-centralised* web, with only a negligible

AIMMES'24: Workshop on AI bias: Measurements, Mitigation, Explanation Strategies, March 20, 2024, Amsterdam
sean.healy@adaptcentre.ie (S. Healy)

impact on fairness outcomes. We demonstrate this through the lens of individual fairness, leaving group fairness for future work or additional re-ranking steps placed beforehand or afterwards within a ranking pipeline.

A major issue facing fairness-aware search adoption on the open web relates to compatibility and required expertise. With the exception of FairSearch [6], most fairness solutions are published in the form of Python experiments, without implementations that work across varied computer systems. Additionally, researching *greedy* fairness methods has been proposed as future work [2, p. 409], but as of this writing, the *greedy* strand of fair ranking research has received little explicit attention, despite efficient (greedy) solutions being vital for widespread feasibility of fair search.

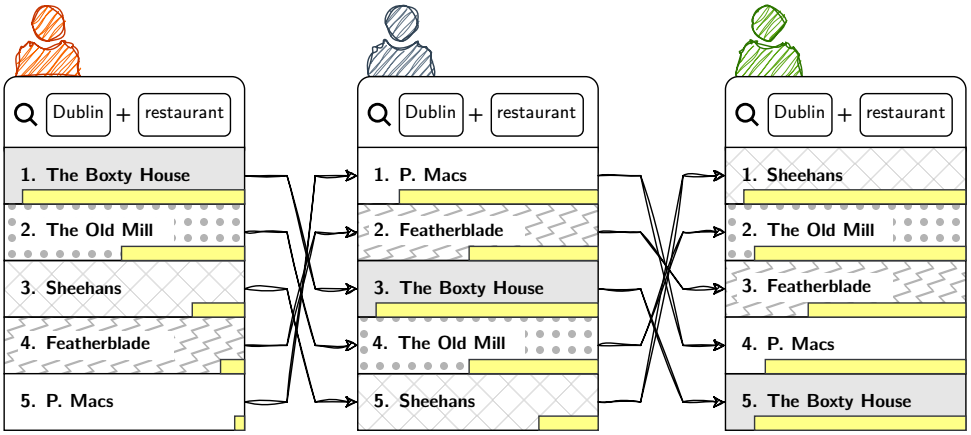

Figure 1: Individual amortized fairness operating in a ranking system across duplicate queries (the yellow bars indicate each document's relative *share* of exposure over time.

2. Background

Though some computationally efficient fairness techniques have been provided [7, 8], most others rely on intense optimisation strategies, necessitating a GPU at the minimum. This creates a high barrier to entry, thus excluding the open web as a consideration. There are also ecological downsides to this approach. In the case of Biega et al.'s technique [2, p. 411], integer linear programming (ILP) was used via a proprietary cloud-based system[1]. Singh and Joachims [9] concurrently suggested an approach equivalent to amortized fairness; their method uses Birkhoff-von-Neumann decomposition (BvN decomp.), and they applied it to groups as opposed to individuals, noting their method could apply to individuals by reducing groups to size one. Again, BvN is a computationally expensive procedure. To our knowledge, Morik et al. [7] were the first to publish work addressing bias in what is known as a *greedy* solution.

Definition 1. *Greedy algorithm. "An algorithm that always takes the best immediate, or local, solution while finding an answer." [10]*

Greedy algorithms can efficiently produce approximate solutions to NP-complete problems, including NP-complete fair ranking problems (Section 3). Though Morik et al. don't directly mention greedy algorithms, their proposed method is undoubtedly *greedy* (Definition 1), particularly given that it is observed to produce oscillating behaviour when a certain λ parameter is not tuned correctly [7, p. 434]. Though this oscillation decreases with λ-tuning, results remain approximate, not optimal. The authors apply their *P-controller* method [7, 11] to group fairness, and don't include quality threshold in their scope (e.g. Biega et al.'s θ threshold [2,

[1] https://www.gurobi.com

Table 1
Comparing some existing fairness techniques in IR: computational costs, quality thresholds, and implementation status.

Method	Comp. Cost	Quality threshold	Production-ready
FA*IR [8]	Low	✗	✓ [6]
P-Controller [7]	Low	✗	✗
DELTR [12]	Low-medium	✗	✓ [6]
ILP [2]	Medium	✓	✗
LP (BvN decomp.) [9]	High	✗	✗

p. 409]). We later present a similarly *greedy* method that accounts for quality threshold, θ, while retaining low computational cost. Like the method of Morik et al., ours is not optimal. We define optimality in terms of the ILP solution [2], which is optimal by definition, given that the ILP solver finds the permutation of valid rankings that minimises the difference between attention $[A_i]$ and relevance $[r_i]$ (scaled by m per ranking count) (1), given a quality constraint (θ) on result pages of length k (2). A major caveat is that this approach only performs within a reasonable duration when lists of length ≈ 100 are being re-ranked [2]. Queries to a web search engine, however, might instead return tens of thousands of similarly ranked results. The ILP problem is defined as follows:

$$\text{minimise} \qquad \sum_{i=1}^{m} |A_i - m \cdot r_i| \tag{1}$$

$$\text{subject to} \qquad \text{NDCG}_k(\rho, \tilde{\rho}_j) \geq \theta, \qquad j \in [1, n] \tag{2}$$

Biega et al. propose (in addition) an online variant of (1), which optimises for a solution ($\tilde{\rho}$) at each ranking as opposed to assuming knowledge of how many rankings (n) are expected.

Some previous fairness work has reached production-ready implementation, notably the work of Zehlike and her colleagues [8, 12, 13], who provide Python/Java libraries, and Elasticsearch plugins for both an in-processing technique (DELTR [12]) and a post-processing technique (FA*IR[8]). We categorise the DELTR approach as low-medium cost, given its reliance on an LTR method (in-processing fairness). Organisations may already have complex LTR pipelines, and integrating this approach might be similarly complex, involving specialised expertise. For companies without an LTR pipeline, relying on traditional *no-data* IR metrics alone (e.g. BM25), this approach would not apply.

The FA*IR approach [8], a post-processing technique, assures ranking prefixes carry predefined proportions across protected groups. The implementation [6] is efficient and requires little additional expertise for deployment. However, it has seen limited adoption in industry (7 'GitHub stars' from industry practitioners, 13 in total as of January 2024). This could be due to a lack of group labels available to businesses (FA*IR is designed as a group-fair method), or perhaps the absence of a quality threshold on rankings produced by FA*IR. We summarise our analysis of these prior works in Table 1.

3. Method

We now propose an individual fairness method that is low-cost and incorporates a quality threshold; we later introduce a reference `C++` implementation (Section 3.2). Our approach is inspired by the combination of characteristics reviewed in Table 1, with an aim to introduce a fair re-ranking approach that can be immediately deployed in many varied computing environments, focusing mainly on open web search and scenarios of group membership ambiguity.

Quality control. Central to the field of information retrieval is quality. Al-Maskari et al. [14] state, *"User satisfaction is a subjective variable and is influenced by several factors (e.g. system effectiveness, user effectiveness, user effort, and user characteristics)."* Two popular metrics for ensuring quality or *system effectiveness* are expected reciprocal rank [ERR] (4) [15] and discounted cumulative gain [DCG]. We adopt a definition of the latter from Burges et al. [16, p. 95] (3).

$$DCG_k = \sum_{i=1}^{k} \frac{2^{\text{rel}(i)} - 1}{\log_2(i+1)} \tag{3}$$

$$ERR_k = \sum_{i=1}^{k} \frac{r_i}{i} \prod_{j=1}^{i-1} [1 - \text{rel}(i)] \tag{4}$$

In the above formulae, $rel(i)$ denotes the relevance of the document at position i in a ranking ρ [or re-ranking $\tilde{\rho}$]. NDCG is most often used as a quality metric during the development of relevance estimation strategies (including LTR). However, within fair re-ranking the formula is instead used in order to retain a certain level of quality; i.e. we define NDCG based on relevance probabilities as revealed to the fair re-ranker from an earlier point in the ranking pipeline. In fair re-ranking, these relevance scores do not arise directly from annotated data, and are generally the output from a ranking function itself. This new usage of the formula prompted Biega et al. to use differing notation for NDCG: NDCG-quality@$k(\rho, \tilde{\rho})$ = DCG@$k(\rho)$/DCG@$k(\tilde{\rho})$ [2], where ρ is a ranking without any amortized fairness applied, and $\tilde{\rho}$ is a ranking with amortized fairness applied. For simplicity, we instead refer to this as NDCG_k in the remainder of this paper.

Our method begins with quality control. Therein we bypass the more difficult question: *"Subject to $NDCG_k \geq \theta$, what is the best re-ranking ($\tilde{\rho}$) to distribute attention A?"* . Instead, we ask a series of easier questions: *"What is the fairest result to return first, subject to $NDCG_k \geq \theta$?"* Then, *"What is fairest to return next?"* etc. This is a greedy approach (Definition 1).

Before we go further, we must introduce what is meant by *exposure*, alternately referred to as *attention* by some authors. These are intuitive terms, especially to those familiar with search engine optimisation. The terms have numeric definitions in fair ranking research. The simplest practical model defines attention as the *marginal probability of examination* [7] in reference to earlier position-based models [17] which found that rank alone can predict exposure with high enough levels of accuracy for practical purposes [7, 18, 19]. In this sense attention would be defined similarly to one of the summand denominators in DCG: $A_i = log(i+1)^{-1}$ [9]. Another model for attention is *geometric* [2], i.e. a special case of the cascade model using equal click probability p for each item: $A_i = (1-p)^{i-1}p$. We suggest a possible amendment to this model: removing the final p multiplicand, given that attention maps to examination of items (not clicks), whereas the p multiplicand in the cascade model specifically models a click. In addition to the geometric model, the more general cascade model can be used to predict attention per-item in a ranking. This would differ from geometric in that each item receives varying click probability. The choice of attention model is important, though not the central scope of this work. We assume a geometric model with $p = 0.4$ in our experiments (Section 4). Returning to the greedy approach, our task is as follows: a) choose an item at each rank which adds the most individual fairness, b) Ensure chosen items are good enough to prevent the overall ranking from dipping below the quality threshold (θ). For the purposes of their ILP-based approach, Biega et al. use the following definition: $\text{unfairness}(\rho_1, \dots, \rho_n) = \sum_{i=1}^{n} |A_i - m \cdot r_i| = \sum_{i=1}^{m} \left| \sum_{j=1}^{m} a_i^j - m \cdot r_i \right|$.

a_i^j is the attention obtained by document i in ranking j, and r_i is the relevance of the document to the outer information need. We use an equivalent minimisation target (5) which aligns more closely with the method we will later outline.

$$\text{unfairness}(\rho_1, \dots, \rho_m) = \max_{i \in [1,n]} \left[\frac{A_i}{\text{rel}(i)} \right] - \min_{i \in [1,n]} \left[\frac{A_i}{\text{rel}(i)} \right]. \tag{5}$$

Proof of equivalence. When Biega et al.'s unfairness is minimised: $A_i - m \cdot r_i = 0, \forall i$. Put differently: $\frac{A_i}{\text{rel}(i)} = m, \forall i$. Therefore, unfairness$(\rho_1, \ldots, \rho_m)$ would be minimised too (5), given that $\max(m) = \min(m), \forall$ constants m.

It is useful to label $\frac{A_*}{\text{rel}(*)}$ itself as the *relative compensation* received by document i (relative to merit or utility). In this sense, we define unfairness as a large gap between the highest relative compensation and the lowest.

3.1. Deriving minimal relevance (a-posteriori) at each rank via an optimistic strategy.

We use an optimistically greedy strategy, meaning we select the *next result* which minimises unfairness (5) so long as this selection is feasible under the assumption that strongly relevant items will fill the remainder of the $k-1$ results (the result page suffix). We refer to this as the *deus ex machina* strategy (DEM); regardless of how low the running NDCG_j becomes at each choice j, highly relevant results might arrive in the tail end, abruptly making up for the predicament.

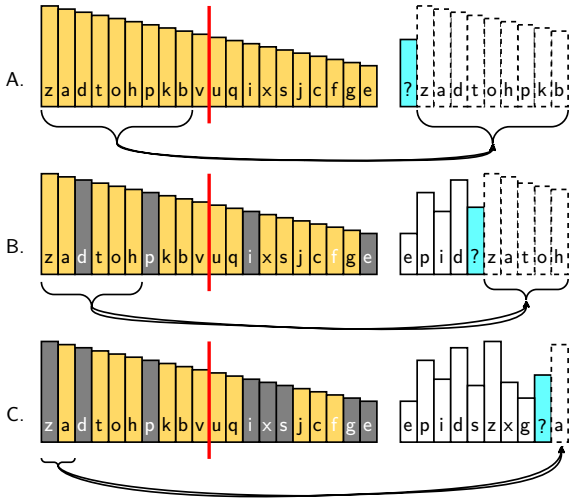

Figure 2: DEM selection; A. depicts the first rank being chosen; B. is fifth rank being chosen; C. is the ninth.

DEM selection is illustrated in three stages during the preparation of a single re-ranking (Figure 2). The left side of the diagram depicts *canonical* relevancies of items, as passed into the re-ranking system from an earlier ranking process. The red line represents the first page cutoff point ($k = 10$). The right side is the ongoing re-ranking of these items, with the sole coloured bar representing the current rank whose item is being decided. While choosing the first item (part A), the missing part of the ranking (dashed blocks) is assumed to be a prefix of the canonically ordered results, $\langle \text{z}, \text{a}, \text{d}, \text{t}, \text{o}, \text{h}, \text{p}, \text{b} \rangle$. Part B occurs after the prior selection of four items using DEM strategy. Notably, one of the items previously selected (d) was in the canonical prefix. We illustrate this by coloring its bar grey, meaning it can no longer be used during optimistic predictions about the remaining results. This is further illustrated in the dashed blocks (far right of part B), where the optimistic prefix excludes d: $\langle \text{z}, \text{a}, \text{t}, \text{o}, \text{h} \rangle$. This selection process continues (part C), and ultimately a re-ranking is produced within a given quality threshold. It might not end up that a re-ranking suffix (right) resembles a canonical prefix (left), but our overall strategy ensures we always remain within the quality threshold at each item choice. We now validate this strategy by considering the quality threshold as introduced earlier ($\text{NDCG}_k \geq \theta$). This can be rewritten as follows.

$$\text{DCG}_k(\tilde{\rho}) \geq \theta \cdot \text{DCG}_k(\rho) \tag{6}$$

Using the notation $\text{rel}(i)$ to represent the relevance of the document at position i in re-ranking $\tilde{\rho}$, and $\text{Rel}(i)$ for relevance at i in the more canonical ranking, ρ (e.g. left in Figure 2), we have:

$$\sum_{i=1}^{k} \frac{2^{\text{rel(i)}} - 1}{\log_2(i+1)} \geq \theta \sum_{i=1}^{k} \frac{2^{\text{Rel}(i)} - 1}{\log_2(i+1)} \tag{7}$$

Under DEM strategy, remaining item choices are assumed to have ideal relevance, and the item under the present rank (j) is picked accordingly. We represent this mathematically by substituting an ideal suffix into the above (7), while also accounting for the relevance of previous item choices:

$$\underbrace{\left(\sum_{i=1}^{j-1} \frac{2^{\text{rel(i)}} - 1}{\log_2(i+1)}\right)}_{\text{past}} + \underbrace{\left(\frac{2^{\text{rel(j)}} - 1}{\log_2(j+1)}\right)}_{\text{present}} + \underbrace{\left(\sum_{i=j+1}^{k} \frac{2^{\widetilde{\text{Rel}}(i)} - 1}{\log_2(i+1)}\right)}_{\text{future (ideal)}} \geq \theta \underbrace{\sum_{i=1}^{k} \frac{2^{\text{Rel}(i)} - 1}{\log_2(i+1)}}_{\text{ultimate ideal}} \tag{8}$$

The above can be re-organised to find a minimum relevance required at the present rank j. Below this relevance $[rel(j)]$, meeting the quality threshold would be impossible.

$$\begin{aligned} \text{rel}(j) &\geq \log_2 \left(1 + \theta \left[\sum_{i=1}^{k} \frac{2^{\text{Rel}(i)} - 1}{\log_{j+1}(i+1)}\right] - \left[\sum_{i=1}^{j-1} \frac{2^{\text{rel(i)}} - 1}{\log_{j+1}(i+1)}\right] - \left[\sum_{i=j+1}^{k} \frac{2^{\widetilde{\text{Rel}}(i)} - 1}{\log_{j+1}(i+1)}\right]\right) \\ &\geq \text{min\_rel}_j^{\text{DCG}} \qquad \text{... the label given for this minimum relevance at each rank} \end{aligned} \tag{9}$$

In the above equations, a distinction is made between canonical relevance $[\text{Rel}(i)]$, and canonical relevance with the previously chosen $j-1$ items removed $[\widetilde{\text{Rel}}(i)]$. This relates to the grey bars in Figure 2; $\text{Rel}(i)$ counts from the beginning, including these grey bars. On the other hand, $\widetilde{\text{Rel}}(i)$ excludes these bars while counting i.

Applying the maths. Our proposed method for greedy individual fairness involves sorting results by relative compensation. We then scan over this list (lowest relative compensation to highest), stopping at the first result with high enough relevance $[\text{rel}_j \geq \text{min\_rel}_j^{\text{DCG}}]$ (9). The scan could be lengthy, depending on the aggressiveness of the quality threshold (high θ), or depending on result corpus quantity and size. Calculating $\text{min\_rel}_j^{\text{DCG}}$ values in advance minimises computer operations during the scan.

Applying DEM selection with ERR quality metrics. Though θ was first used with DCG [2], it can also be used with any ranking quality metric, including ERR [introduced earlier (4)]. This is another advantage of the greedy approach over the ILP approach; ILP requires a highly efficient quality metric, whereas the greedy method does not.

$$\begin{aligned} \sum_{i=1}^{k} \frac{r_i}{i} \prod_{m=1}^{i-1} [1 - \text{rel}(i)] &\geq \theta \sum_{i=1}^{k} \frac{r_i}{i} \prod_{m=1}^{i-1} [1 - \text{Rel}(i)] \\ &\geq \theta \cdot \text{IERR}_k \qquad \text{...we use IERR}_k \text{ (\textit{ideal} ERR) as shorthand.} \end{aligned} \tag{10}$$

Expressing $\text{rel}(i)$ in terms of everything else (in the case of ERR), assuming a DEM strategy (Figure 2), is more complicated than with DCG, the latter being a much more *stateless* metric. For ERR, we have:

$$\underbrace{\left(\sum_{i=1}^{j-1} \frac{\text{rel}(i)}{i} \prod_{m=1}^{i-1} [1 - \text{rel}(m)]\right)}_{\text{past}} + \underbrace{\left(\frac{\text{rel}(j)}{j} \prod_{m=1}^{j-1} [1 - \text{rel}(m)]\right)}_{\text{present}} + \underbrace{\text{Future}}_{\text{" (ideal)}} \geq \theta \cdot \underbrace{\text{IERR}_k}_{\text{ultimate ideal}} \tag{11}$$

$$\text{with...} \qquad \text{Future} = \sum_{i=j+1}^{k} \left[\frac{\widetilde{\text{Rel}}(i-j)}{i} \left(\prod_{m=1}^{j-1} [1 - \text{rel}(m)] \right) [1 - \text{rel}(j)] \prod_{n=j+1}^{i-1} [1 - \widetilde{\text{Rel}}(n-j)] \right]$$

The above can be rewritten in the following intermediate form:

$$\frac{\text{rel}(j)}{j} + \text{X} \geq \frac{\theta \cdot \text{IERR}_k - \sum_{i=1}^{j-1} \frac{\text{rel}(i)}{i} \prod_{m=1}^{i-1} [1 - \text{rel}(m)]}{\prod_{m=1}^{j-1} [1 - \text{rel}(m)]} \tag{12}$$

$$\text{with...} \qquad \text{X} = [1 - \text{rel}(j)] \sum_{i=j+1}^{k} \frac{\widetilde{\text{Rel}}(i-j)}{i} \prod_{n=j+1}^{i-1} [1 - \widetilde{\text{Rel}}(n-j)]$$

The completed rewrite, focusing on expressing $\text{rel}(j)$ in terms of everything else, follows.

$$\begin{aligned} \text{rel}(j) &\geq \frac{\frac{\theta \cdot \text{IERR}_k - \sum_{i=1}^{j-1} \text{rel}(i) \prod_{m=1}^{i-1} [1 - \text{rel}(m)]}{\prod_{m=1}^{j-1} [1 - \text{rel}(m)]} - \sum_{i=j+1}^{k} \frac{\widetilde{\text{Rel}}(i-j)}{i} \prod_{n=j+1}^{i-1} [1 - \widetilde{\text{Rel}}(n-j)]}{\frac{1}{j} - \sum_{i=j+1}^{k} \frac{\widetilde{\text{Rel}}(i-j)}{i} \prod_{n=j+1}^{i-1} [1 - \widetilde{\text{Rel}}(n-j)]} \\ &\geq \text{min\_rel}_j^{\text{ERR}} \end{aligned} \tag{13}$$

Though the above formula involves a lot of summed products, previous work has provided optimised $\text{O}(k)$ techniques for calculating these parts [15]. Furthermore, k in our web search use-case will generally be quite low ($\ll 75$), and min_rel_j^* need only be calculated once per item returned to the user.

Minimum relevance a-priori. Above, we presented means of reducing computer operations during the linear scan of items ordered by relative compensation $\left[\frac{A_*}{\text{rel}(*)}\right]$. Another optimisation for reducing the cost of this linear scan is to pre-calculate a-priori minimum relevance (given quality threshold θ) at each rank j. Consider a scenario where almost all items in the re-ranking $\tilde{\rho}$ are ideal (relative to ρ), just not the item at rank j. The minimum relevance necessary at j in order to make (14) or (15) hold true is thus defined by substituting $\text{Rel}(\cdot)$ for all instances of $\widetilde{\text{Rel}}(\cdot)$ and $\text{rel}(\cdot \neq j)$ in (9) and (13) respectively. Any items below this relevance can be filtered out of a candidate list per-rank from the very start. This results in a shorter linear scan. We denote this cutoff relevance as $\text{min\_rel\_priori}_j^*$, and present its derived expression below, first for DCG and then for ERR.

$$\begin{aligned} &\text{min\_rel\_priori}_j^{\text{DCG}} = \\ &\log_2 \left(1 + [\theta - 1] \left[\sum_{i=1}^{j-1} \frac{2^{\text{Rel}(i)} - 1}{\log_{j+1}(i+1)} \right] + \theta \left[\sum_{i=j}^{k} \frac{2^{\text{Rel}(i)} - 1}{\log_{j+1}(i+1)} \right] - \sum_{i=j}^{k-1} \frac{2^{\text{Rel}(i)} - 1}{\log_{j+1}(i+2)} \right) \end{aligned} \tag{14}$$

$$\text{min\_rel\_priori}_j^{\text{ERR}} = \frac{\frac{\theta \cdot \text{IERR}_k - \sum_{i=1}^{j-1} \text{Rel}(i) \prod_{m=1}^{i-1} [1 - \text{Rel}(m)]}{\prod_{m=1}^{j-1} [1 - \text{Rel}(m)]} - \sum_{i=j+1}^{k} \frac{\text{Rel}(i-j)}{i} \prod_{n=j+1}^{i-1} [1 - \text{Rel}(n-j)]}{\frac{1}{j} - \sum_{i=j+1}^{k} \frac{\text{Rel}(i-j)}{i} \prod_{n=j+1}^{i-1} [1 - \text{Rel}(n-j)]} \tag{15}$$

Below, we outline at a high-level how our method works. Following this we discuss an edge-case. D streams into the re-ranking system ordered by relevance to the information need.

(A) Serve top k items as the first ranking, $\tilde{\rho}_1 = \rho$, updating A_* accordingly.

(B) Copy $(k-1)$ top items from D into I (left of red line, Figure 2).

(C) Calculate $\text{min\_rel\_priori}_j^*$ using I and θ (14 & 15), $\forall j \in [1, k]$

(D) Remove items i from D where $\text{rel}(i) < \text{min\_rel\_priori}_k^*$ (14 & 15).

(E) Create candidate lists per-rank, $j \in [1, k]$, st. list item relevances $\geq min\_rel\_priori^*_j$.

(F) Sort candidate lists by by relative compensation $\left[\frac{A_*}{\text{rel}(*)}\right]$, low $\rightarrow$ high.

(G) Initiate a new re-ranking, $\tilde{\rho}_*$.

(H) For j between 1 and k:

(H)(1) Calculate min a-posteriori relevance at j (min_rel^*_j), using I, θ and $\tilde{\rho}_*$ (9 & 13).

(H)(2) Scan over candidates, until an item i with $\text{rel}(i) < \text{min\_rel}^*_j$ is found. Add it to $\tilde{\rho}_*$.

(H)(3) Mask said item in all candidate lists until the loop completes (no duplicate selections).

(I) Serve $\tilde{\rho}_*$ as the next ranking, update A_*, and return to Step (F).

An edge-case. One caveat excluded from the overview (for brevity) is that sometimes the item selected at a given rank (coloured bars, right hand side, Figure 2) will be part of the *optimistic tail* (dashed blocks, Figure 2). In many cases, this will not be an issue, given that the a-posteriori *past + present + future* quality score (9 & 13) is likely to remain within threshold. However, this does constitute *double-counting*, and there will be some edge-cases where this becomes an issue. See Figure 3.

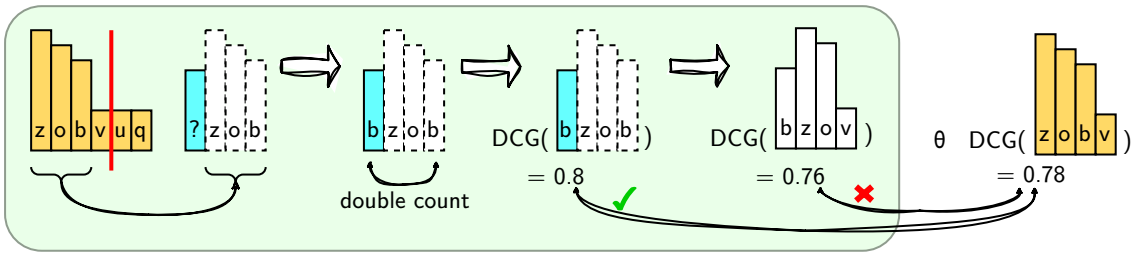

Figure 3: An edge-case during DEM selection; b is selected, and DEM counts it both in the *present* and *future* parts of (8). Expected (ultimate) quality, on completion of the ranking, is therefore estimated inaccurately.

An item b is chosen at the first rank. In reality, only the first two items, z and o, would result in a ranking that meets the quality threshold of $\theta \cdot IDCG_4$. The presence of b in the optimistic tail (dashed blocks) causes it to be *double counted*.

Rather than addressing this issue before it arises, we instead prioritise linear scan speed, and *initially let the error happen*, only checking after a candidate item is found whether or not it is part of the *optimistic tail* (dashed boxes in Figure 2), and furthermore, whether the corrected quality of the ranking falls below the threshold.

To address this edge-case, we insert the below steps in the algorithm outline, displacing the original Steps (H)(1)$\rightarrow$(H)(3) to (H)(4)$\rightarrow$(H)(6). The new *Step (H)(3)* might seem controversial, given that it scans upwards through I, a much shorter list than D. However, it can be proven that when the edge-case occurs, scanning anything outside this upper-I part would be fruitless.

(H)

(H)(1) Check if previously selected document i is in I [skip to ((H)(4)) if not].

(H)(2) If so, mask it in I, re-calculate min_rel^*_{j-1}; check θ was truly respected (8 & 11).

(H)(3) If it was not, unmask i and scan up through I until a document i' fulfils the threshold. Continue backwards (to very beginning of I), finding i'' with lowest $\frac{A_*}{\text{rel}(*)}$. Choose and subsequently mask i'' instead of i.

(H)(4) … business as usual …

Proof for sufficiency of upward scan [Step (H)(3)]. We further illustrate the edge-case double count problem via Figure 4, where a document x is chosen, but this document happens to be the *optimistic tail* (dashed blocks). Given the left scenario in Figure 4 meets quality standards, but the middle (accounting for the double count correctly) does not, we wish to know if this indicates that the right side scenario will also not meet quality standards. In the right hand side, rather than selecting x, another document beyond the end of the *optimistic tail* is chosen. All items after x in the canonical ordering are assumed to have equal relevance, $t < \text{rel}(j_x)$, and the re-occurrence of x in the *optimistic tail* occurs at offset m into that tail.

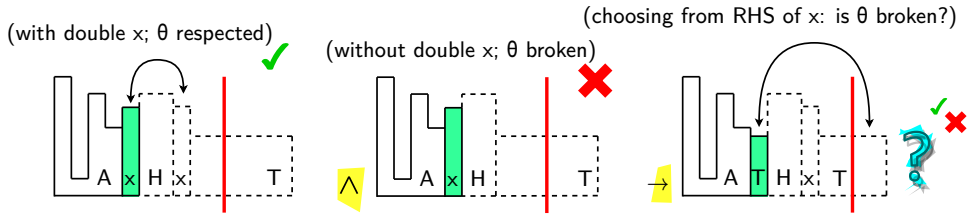

Figure 4: DEM edge-case optimisation (support diagram for proof)

The diagram illustrates the formulae below. (16) corresponds to the middle portion of Figure 4, and (17) corresponds to the right portion. We use a symbol in (17) representing the unknown operator, $[\![\oplus]\!] \in \{[\![<]\!], [\![\geq]\!]\}$. We seek to prove that $[\![\oplus]\!] = [\![<]\!]$.

$$\underbrace{\sum_{i=1}^{j-1} \frac{2^{rel(i)}-1}{\log_2(i+1)}}_{\text{A}} + \underbrace{\frac{2^{rel(j_x)}-1}{\log_2(j+1)}}_{\text{x (middle)}} + \underbrace{\sum_{i=1}^{m-1} \frac{2^{\widetilde{Rel}(i)}-1}{\log_2(j+i+1)}}_{\text{H (middle)}} + \underbrace{\sum_{i=m+1}^{k-j} \frac{2^{t}-1}{\log_2(j+i)}}_{\text{T (middle)}} < Y \tag{16}$$

... with $Y = \theta \cdot IDCG_k$

$$\underbrace{\sum_{i=1}^{j-1} \frac{2^{rel(i)}-1}{\log_2(i+1)}}_{\text{A}} + \underbrace{\frac{2^{t}-1}{\log_2(j+1)}}_{\text{T (right; first)}} + \underbrace{\sum_{i=1}^{m-1} \frac{2^{\widetilde{Rel}(i)}-1}{\log_2(j+i+1)}}_{\text{H (right)}} + \underbrace{\frac{2^{rel(j_x)}-1}{\log_2(j+m)}}_{\text{x (right)}} + \underbrace{\sum_{i=m+2}^{k-j} \frac{2^{t}-1}{\log_2(j+i)}}_{\text{T (right; second)}} \oplus Y \tag{17}$$

Shifting some common parts from (16 & 17) [i.e. A, H] to the RHS, '$Y -$ ShiftedParts' can be replaced with another constant, C, and we now have a simpler problem. If we can prove that $[\![\oplus]\!] = [\![<]\!]$ in (19), then we will have proven the same in (17).

$$\underbrace{\frac{2^{rel(j_x)}-1}{\log_2(j+1)}}_{\text{x (middle)}} + \underbrace{\frac{2^{t}-1}{\log_2(j+m+1)} + \sum_{i=m+2}^{k-j} \frac{2^{t}-1}{\log_2(j+i)}}_{\text{T (middle)}} < C \tag{18}$$

$$\underbrace{\frac{2^{t}-1}{\log_2(j+1)}}_{\text{T (right; first)}} + \underbrace{\frac{2^{rel(j_x)}-1}{\log_2(j+m+1)}}_{\text{x (right)}} + \underbrace{\sum_{i=m+2}^{k-j} \frac{2^{t}-1}{\log_2(j+i)}}_{\text{T (right; second)}} \oplus C \tag{19}$$

Given that the LHS of (19) is clearly smaller than the LHS of (18), by transitivity $[\![\oplus]\!] = [\![<]\!]$. Ultimately, this is why we only need to scan upwards in Step (H)(3). With descending (non-uniform) T (Figure 4), this observation would only be exacerbated.

3.2. Implementation and Usage

In our implementation, we use a candidate list per rank (Figure 5, pre-populated with feasible items. This reduces the length of the linear scans, and creates a strong basis for future hybrid

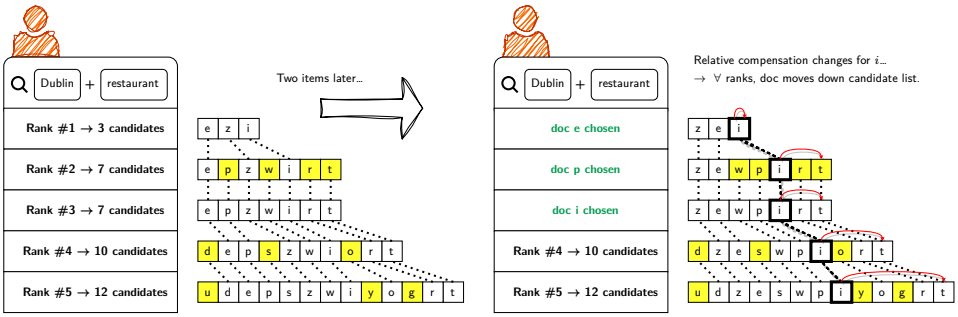

Figure 5: Candidate lists per-rank, as determined by the θ threshold. Candidate list length ascends with rank, and relative-order is preserved in lesser ranked candidate lists (no overlapping dashed lines).

greedy/shallow graph-search approaches. The methods and optimisations described above can be implemented in any programming language, but we provide a sample GPLv3-licensed implementation written in standard `C++`[†]. This choice focuses on the open web, given the ubiquity of `C++` compilers for all modern operating systems, and the performance benefits associated with the low-level programming paradigm. The implementation offers a built-in CLI interface with further integrations to follow. The underlying re-ranker runs as a daemon, reachable by different processes. With the optimisations described in Section 3, particularly the DEM selection strategy, individually fair re-rankings are placed within reach for any organisation limited to commodity hardware. Usage instructions can be found within the project website.

4. Evaluation

RQ1 Can we address quality-constrained individual fairness via greedy *item-by-item* selection?

- RQ1a Is the greedy method comparable to best-case scenarios (ILP, Graph search, BvN decomp.) in terms of optimal fairness per-ranking?
- RQ1b Is the greedy method fast enough for general-purpose usage?
- RQ1c Is the installation process easy enough to support a non-centralised ecosystem?

The first research question (RQ1) is more-so a mathematical one, with an answer etched out in Section 3, as well as in the subsequent implementation (Section 3.2). However, without analysing how closely the fairness results of this approach align with the best case (RQ1a), it would be unclear under what scenarios each method is better.

RQ1a Optimality We were unable to obtain funding for Gurobi's cloud-based solution. However, we were able to ascertain some optimally fair re-rankings using exhaustive graph search in SWI-Prolog. We then used these optimal solutions as a reference point to estimate fairness loss when using our greedy method. The Prolog solution space search relies on an observation from Figure 5: given a high enough quality threshold, low ranks will have small numbers of feasible candidates, and higher ranks will have higher numbers of feasible candidates. This a-priori observation considerably reduces the search space of feasible rankings. With a small enough corpus size (≈ 30), and proper re-calculation of feasible items (DEM strategy, Section 3.1), the search space becomes small enough to iterate over each valid ranking. It is thereby possible to determine how many *excess rankings* our method requires, as compared to a re-ranker that finds the optimally fair re-ranking at each query instance. We investigate this for various θ values, with the initial and trivial observation that when $\theta = 0$, the greedy method produces optimal rankings. Likewise, when $\theta = 1$, greedy re-rankings are optimal.

[†]https://librecoir.com/

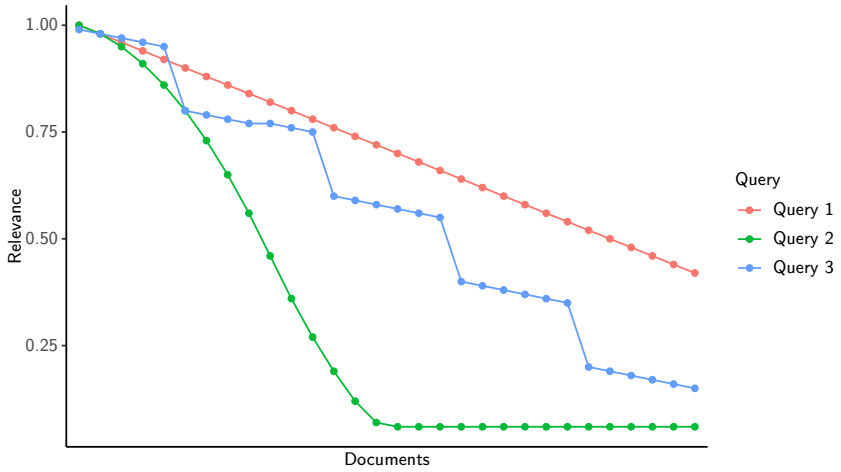

Figure 6: Three simulated queries, for comparing the performance of greedy fairness against optimal fairness.

Using an aggregated metric for *unfairness* (20), we conducted experiments with the three query types from Figure 6. This metric counts instances of negative discrimination (under-representation as opposed to over-representation). The attention afforded to each document is made relative by division with the sum of all attention. This value is then compared to the item's relative utility, $\text{rel}(i)/\sum \text{rel}(j)$.

$$\text{unfairness} = \sum_{i \in [1,n]} \max \left[0, \frac{rel(i)}{\sum_j \text{rel}(j)} - \frac{A_i}{\sum_j A_j} \right] \tag{20}$$

The results from this experiment indicated that fairness is much more attainable with some query types versus others. Query 2 in particular (the green peak in Figure 6) was able to approach fairness relatively quickly, even when θ was set high ($\theta \approx 0.88$). However, with excessively high θ, results are inevitably unfair. For example, when a quality threshold of $\theta = 0.98$ is used, barely any change in ordering is allowed near the top of the result page, and this results in a slowly rising unfairness metric. On the other hand, when quality threshold is low enough ($\theta \approx 0.78$), fairness is achieved in Query 2 very early on. Even with more difficult queries (Queries 1 and 2), a low enough θ threshold ($\theta < 0.5$) while using the greedy method leads to reasonable individual fairness. Though the comparison with optimal rankings was restricted to small levels of duplication, and thereby not statistically significant, there is inevitably a trade-off between algorithm efficiency and optimality. This is future research.

RQ1c Ease-of-use Evaluating a system's ease-of-use is a qualitative and highly subjective task, especially in the case of WWW software; a software suite or package built for the web could be easy to install one month, later rendered entirely unusable the next month due to dependency versioning or backwards compatibility issues. In the case of ML systems, the problem is made worse by widespread practices of publishing pre-trained models without background training sets. At times this is done for copyright reasons, other times because the dataset is too large. We aimed to bypass most compatibility issues by writing the CO*IR program in C++, purposely avoiding external dependencies. Additionally, we bypass issues around datasets by producing a fairness method that doesn't rely on data. Future work will amend CO*IR to allow for gradually acquired training data for tuning θ automatically per-query. Evaluation of the system's ease-of-use is ongoing, and we welcome feedback from all stakeholders, especially those in positions to implement fairness in product settings (software engineers, system admins, etc.)

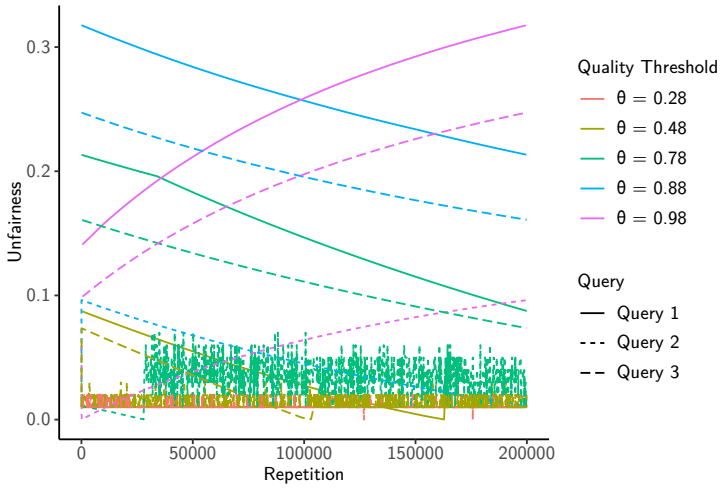

Figure 7: Results from experiments conducted with distinct simulated queries from Figure 6

Table 2

Computing time taken for bulk re-ranking jobs on a large corpus (10,000 items).

Load	Total computing time (i5 core)
500 re-rankings	0.221 seconds
2000 re-rankings	0.730 seconds
8000 re-rankings	2.925 seconds

RQ1b Efficiency One of the primary purposes of this work was the attempted introduction of individual fairness to web-scale search on commodity hardware. To evaluate this goal, the CO*IR system was applied to the task of re-ranking a query with 10,000 competing documents, and a gradually declining relevancy slope among documents (like Query 1 in Figure 6). CO*IR was able to produce 8,000 re-rankings in under 3 seconds on an i5 processor (2.30GHz). Results from this experiment are presented in Table 2.

5. Conclusion and Future Work

In this work we considered the problem of *barriers to entry* in fair re-ranking, particularly individually fair re-ranking. We designed and analysed a fair re-ranker for commodity hardware. Optimisations included: tackling the re-ranking problem in a greedy fashion, pre-eliminating infeasible (irrelevant) items, and exploiting edge-cases to allow for early exit from a linear scan. With these optimisations, applying individual fairness on a much higher quantity of results (as compared to prior work; $\gg 100$) is now within reach for smaller (as well as larger) organisations.

Future work might aim to make the package more usable in web search scenarios (the preliminary version targets standalone rankings). Distributing the program's execution across a network of collaborating servers would also be advantageous for the open web. Integrating group fairness as an additional feature, alongside individual fairness, would also be necessary for a more holistically ethical ranking tool, as individual fairness would be inappropriate for the purposes of positive action intervention in rankings [20]. For group fairness, however, there already exists FairSearch [6], which we recommend. Optional multi-threading or GPU support would also increase our tool's efficiency in many scenarios. We didn't target this initially; *version one* was intended first-and-foremost to bring individually fair ranking into the range of possibility for small organisations partaking in an open, non-centralised, web.

Finally, we aim to introduce a modified version of the CO*IR algorithm, which uses the same heuristics as above (8 and 13), applying them in A* graph search [21]. With a pessimistic

heuristic, A* search produces *admissible* solutions [22]; and would therefore be equivalent to the ILP method used by Biega et al. [2]. Varying between pessimistic and optimistic heuristics, we hypothesise it is possible to have a fairness solution that uses system resources dynamically, i.e. *optimality* when the resources are there, sub-optimality otherwise, and many different grades of optimality in-between the two extremes.

Acknowledgements

This paper was in part funded by the Science Foundation Ireland. I must thank Maaike Visser for her ongoing support, sharing of ideas and guidance. Her Masters thesis [23] provided me an introduction to fairness evaluation metrics, which will help inform future work.

References

[1] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, R. Zemel, Fairness through awareness, in: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ITCS '12, Association for Computing Machinery, 2012, pp. 214–226.

[2] A. J. Biega, K. P. Gummadi, G. Weikum, Equity of attention: Amortizing individual fairness in rankings, in: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, 2018, pp. 405–414.

[3] J. Biega, Enhancing privacy and fairness in search systems, 2018.

[4] T. Berners-Lee, Information management: A proposal, https://web.archive.org/web/20100401051011/https://www.w3.org/History/1989/proposal.html, 1989. Accessed: March 24, 2024.

[5] C. Santesteban, S. Longpre, How big data confers market power to big tech: Leveraging the perspective of data science, The Antitrust Bulletin 65 (2020) 459–485. URL: https://doi.org/10.1177/0003603X20934212. doi:10.1177/0003603X20934212. arXiv:https://doi.org/10.1177/0003603X20934212.

[6] M. Zehlike, T. Sühr, C. Castillo, I. Kitanovski, Fairsearch: A tool for fairness in ranked search results, in: Companion Proceedings of the Web Conference 2020, WWW '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 172–175. URL: https://doi.org/10.1145/3366424.3383534. doi:10.1145/3366424.3383534.

[7] M. Morik, A. Singh, J. Hong, T. Joachims, 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, Association for Computing Machinery, New York, NY, USA, 2020, p. 429–438. URL: https://doi.org/10.1145/3397271.3401100. doi:10.1145/3397271.3401100.

[8] M. Zehlike, F. Bonchi, C. Castillo, S. Hajian, M. Megahed, R. Baeza-Yates, FA*IR: A fair top-k ranking algorithm, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM '17, Association for Computing Machinery, 2017, pp. 1569–1578.

[9] A. Singh, T. Joachims, Fairness of exposure in rankings, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '18, Association for Computing Machinery, New York, NY, USA, 2018, pp. 2219–2228.

[10] P. E. Black, Greedy algorithm, https://www.nist.gov/dads/HTML/greedyalgo.html, 2005. Accessed: 2024-02-12.

[11] B. W. Bequette, Process Control: Modeling, Design and Simulation, Prentice Hall, Rensselaer Polytechnic Institute, 2003.

[12] M. Zehlike, C. Castillo, Reducing disparate exposure in ranking: A learning to rank approach, in: Proceedings of The Web Conference 2020, WWW '20, Association for

Computing Machinery, New York, NY, USA, 2020, p. 2849–2855. URL: https://doi.org/10.1145/3366424.3380048. doi:10.1145/3366424.3380048.
[13] M. Zehlike, Fairness in Rankings (Dissertation), Ph.D. thesis, 2022.
[14] A. Al-Maskari, M. Sanderson, A review of factors influencing user satisfaction in information retrieval, Journal of the American Society for Information Science and Technology 61 (2010) 859–868. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/asi.21300. doi:https://doi.org/10.1002/asi.21300. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/asi.21300.
[15] O. Chapelle, D. Metlzer, Y. Zhang, P. Grinspan, Expected reciprocal rank for graded relevance, in: Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM '09, Association for Computing Machinery, New York, NY, USA, 2009, p. 621–630. URL: https://doi.org/10.1145/1645953.1646033. doi:10.1145/1645953.1646033.
[16] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, G. Hullender, Learning to rank using gradient descent, in: Proceedings of the 22nd International Conference on Machine Learning, ICML '05, Association for Computing Machinery, New York, NY, USA, 2005, p. 89–96. URL: https://doi.org/10.1145/1102351.1102363. doi:10.1145/1102351.1102363.
[17] N. Craswell, O. Zoeter, M. Taylor, B. Ramsey, An experimental comparison of click position-bias models, in: Proceedings of the 2008 International Conference on Web Search and Data Mining, WSDM '08, Association for Computing Machinery, New York, NY, USA, 2008, p. 87–94. URL: https://doi.org/10.1145/1341531.1341545. doi:10.1145/1341531.1341545.
[18] A. Agarwal, I. Zaitsev, X. Wang, C. Li, M. Najork, T. Joachims, Estimating position bias without intrusive interventions, in: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 474–482. URL: https://doi.org/10.1145/3289600.3291017. doi:10.1145/3289600.3291017.
[19] X. Wang, N. Golbandi, M. Bendersky, D. Metzler, M. Najork, Position bias estimation for unbiased learning to rank in personal search, in: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM '18, Association for Computing Machinery, New York, NY, USA, 2018, p. 610–618. URL: https://doi.org/10.1145/3159652.3159732. doi:10.1145/3159652.3159732.
[20] M. Zehlike, A. Loosley, H. Jonsson, E. Wiedemann, P. Hacker, Beyond incompatibility: Trade-offs between mutually exclusive fairness criteria in machine learning and law, 2022. arXiv:2212.00469.
[21] P. E. Hart, N. J. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, IEEE Transactions on Systems Science and Cybernetics 4 (1968) 100–107. doi:10.1109/TSSC.1968.300136.
[22] S. J. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, 4th ed., Pearson, Boston, 2018.
[23] M. Visser, Investigating Fair Rankers Under the Expected Exposure Framework, Master's thesis, 2022.