FairSearch: A Programming Library for Fair Search Results

Extended Abstract

Meike Zehlike
Technische Universität Berlin
Berlin, Germany
meike.zehlike@tu-berlin.de

Carlos Castillo
Universitat Pompeu Fabra
Barcelona, Catalunya, Spain
chato@acm.org

Ivan Kitanovski
Faculty of Computer Science and Engineering
Skopje, Macedonia
ivan.kitanovski@finki.ukim.mk

1 INTRODUCTION

In the last decade, the online world has evolved well beyond being just a platform for trivial content findings, personal updates and traveling blogs. Social media, e-commerce, professional, political, educational, and dating sites, to mention just a few, determine our possibilities and success as individuals, consumers, employees, voters, students, and lovers. Ranked search results, news feeds, and recommendations, have become the main mechanism by which we find content, products, places, and people online. Indeed, there are very few large websites without a search function.

It is therefore of societal and ethical importance to ask whether search algorithms produce results that can demote, marginalize, or exclude individuals of unprivileged groups (e.g., racial or gender discrimination) or promote products with undesired features (e.g., gendered books).

Search results are almost always ranked in descending order of some relative quality measure of the items. There are, however, reasons to depart from a simple ranking by relevance or utility, and also consider the utility of those being searched, particularly when the items represent people or businesses.

Due to its high importance and impact, our aim is *to develop the first fair open source search API*, FairSearch. Our goal for FairSearch is to provide *pre-, in- and post-processing* approaches for fair ranking algorithms. It currently provides a post-processing method, FA*IR [1], which can enforce ranked group fairness, ensuring that all prefixes of the ranking have a fair share of items across the groups of interest, and ranked individual fairness, reducing the number of cases in which a less qualified or lower scoring item is placed above a more qualified or higher scoring item. By usage of in-processing methods such as DELTR [2] web and search engine developers will be able to tune their applications during search ranking computation for reduction of disparate exposure across groups within the results with great flexibility.

We build FairSearch by extending a popular, well-tested open source search engine: ElasticSearch.[1] Taking a long-term view, we believe the use of this tool will be an important step towards achieving equality of opportunity and reducing inequality and discrimination in the online world, and consequently in society as a whole.

With this paper we want to present details about the implementation and architecture of FairSearch, how to use it as a developer and future ideas for its extension.[2][3]

[1] https://www.elastic.co/
[2] Funded by Data Transparency Lab https://datatransparencylab.org/grantees-dtl-2017/
[3] Code available on https://github.com/fair-search

DSSGW'19, May 2019, San Francisco, USA
2019. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

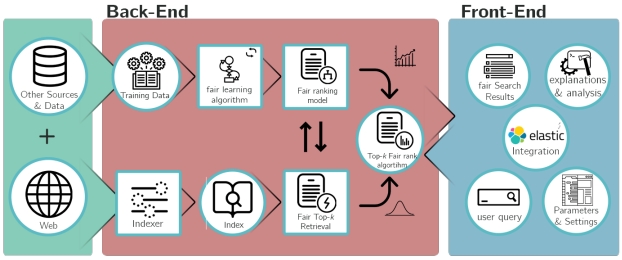

Figure 1: Tool structure and main components

2 ARCHITECTURE

Figure 1 shows the architecture and main components of the proposed tool as an extension of the Elasticsearch platform. Elasticsearch is an open source enterprise search system written in Java, used by thousands of online services including Netflix, Facebook, Ticketmaster, and Instagram. Its major features include full-text search, hit highlighting, faceted search, real-time indexing, dynamic clustering, database integration, NoSQL features [4] and rich document (e.g., Word, PDF) handling and is designed for scalability and fault tolerance. At each stages of the Elasticsearch

Front-end. The front-end will contain a UI system administrators for to enter parameters for all included methods (such as the minimum fraction p of items that have a protected attribute for FA*IR), while defaults will be preset in a configuration file.

Back-end. The main back-end component are different modules to generate the fair top-k search results with respect to the user-provided input parameters. Each module component will consist of one "fair" search algorithm. This will allow users of FairSearch to choose from different fairness and distributive equality frameworks and such adjust their application to their specific needs of non-discrimination and equal opportunity. While DELTR works *in-processing* and will be used during search index computation such that the resulting ranking models reduce disparate exposure across groups, the *post-processing* algorithm FA*IR reorders the result ranking itself to satisfy the fairness constraints defined in [1]. FA*IR is additionally provided as a stand-alone Java library in case developers want to use it in a different ranking environment than Elasticsearch.

REFERENCES

[1] Meike Zehlike, Francesco Bonchi, Carlos Castillo, Sara Hajian, Mohamed Megahed, and Ricardo Baeza-Yates. 2017. FA*IR: A fair top-k ranking algorithm. In *Proc. of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 1569–1578.
[2] Meike Zehlike and Carlos Castillo. 2018. Reducing Disparate Exposure in Ranking: A Learning To Rank Approach. *arXiv preprint arXiv:1805.08716* (2018).

[4] http://nosql-database.org/