SI 650 / EECS 549 F23 Homework 5
Bias and Diversity

Due: See Canvas for official deadline

1 Homework Overview

Throughout the course, we've discussed how different IR model can unintentionally introduce disparities in search behavior and results—we even tried looking for these in class with our audits. In Homework 5, you'll extend our IR system in small ways to potentially improve its fairness.

The homework is scoped as two small extensions and then an open-ended task and reflection. Fairness in IR is an on-going topic and the subject of much research so we are (sadly) not fixing the issues you have seen in class and in your previous homework. However, this homework will give you some better intuition on how you might address the problem and how you might measure fairness, though how to measure is still an open question too!

This assignment has the following learning goals:

1. Learn how to increase diversity in the IR results
2. Better understand how fairness can be measured
3. Deepen understanding of algorithmic changes that can affect fairness in IR
4. Gain skills in development, debugging, performance optimization, and plotting
5. Practice reading equations and turning them into code.
6. Practice explaining technical content to an audience

2 Important Notes

Unlike previous homeworks, this homework should not cause an increase the memory requirements for running the core IR system due to needing to keep more network information around. If your Homework 4 code ran fine, it should still work just fine for Homework 5. You can continue to use the Great Lakes cluster as you need however, so free to request time on computers with larger memory. You should have access through the course account `si650f23_class`. If the IR system works fine on your machine, you do not need to use Great Lakes. You do not need to use GPUs for this assignment; we can use all the precomputed vectors and queries from previous assignments.

Homework 5 includes a few small extensions to the core search engine implementation and design.

1. The `query` method now has an optional `user_id` argument that indicates which user is issuing the query.

2. Learn how to personalize search results based on a user's prior interests
3. The method signature for `score` function has been changed slightly so that instead of taking in a list of query terms, the method now takes in a dictionary of term to count.[1]

Things to keep in mind when working on this assignment:

1. Please start early
2. You should compute the personalized page rank scores separately from the main IR system (like you did for the non-personalized scores)

To reduce ambiguity in this assignment, the following notation is included here and through all future assignments.

1. An arbitrary word is w_i where i refers to which word this is within the vocabulary V. A word can be a single token or a multi-word expression.[2]
2. An individual document is d, which may be denoted as d_i to indicate an arbitrary document in the collection. The length of the document in words is denoted as $|d|$, which includes stop words (before they are filtered).
3. The set of all documents in the collection is D and $|D|$ denotes the number of documents in the collection
4. The set of all documents' titles is T and $|T|$ denotes the number of documents' titles in the collection. In practice, this is the same as the number of documents but we use T to make the distinction clearer in the equations of whether we're referring to the document's main text (D) or its title (T).
5. $c_d(w_i)$ is the count of how many times w_i appears in the main text of document d (not including any other metadata/title associated with the document)
6. $c_t(w_i)$ is the count of how many times w_i appears in the title t (this assumes that all documents have a title)
7. $df_{w_i}^D$ is the number of documents in D that contain w_i in the body text (not title or metadata). $df_{w_i}^T$ is the analogous value for the number of titles that a word occurs in.
8. $avdl$ is the average document length, i.e., $\frac{1}{|D|}\sum_i |d_i|$
9. D_r is the set of documents that a user has marked as relevant; D_n is the set of documents that a user has marked as not-relevant. We are not using D_n in practice for this homework.
10. The *seed set* is the set of documents that a user has indicated explicitly or implicitly are related to their interests. These could be the bookmarks they have saved in their browser, for example. We refer to this set as R
11. $|R|$ is the number of seed documents
12. r_i is the number of documents in the seed set that have word w_i

[1] Nearly all implementations were already calculating and using this data structure in the function already, so this change is relatively minor. We have provided code on how to quickly convert your method in the code README.

[2] Note that for Homework 2, there are no multi-word expressions.

3 Measuring Fairness in Ranking

In the previous homeworks, we looked at how the rankings for generic queries (e.g., "person") varied with respect to pages that were associated with protected attributes. To get a sense of how "fair" the results were, you simply calculated the average rank by group. However, that metric is potentially overly simple for a few reasons. First, the average rank ignores how important differences are at different parts of the ranking; differences in rank at the top of the list should matter more. In comparison, we use a discount in NDCG to indicate this preference when evaluating quality. Second, the average rank ignores how common pages with each attribute are; when some attributes are more present in the data, perhaps the score is getting skewed more than we think!

This first part of Homework 5 will have you implement *one* way of quantifying fairness. How to measure fairness is an open question and we are not going to implement all metrics or more advanced metrics here, though you're encouraged to read more if you're interested. Specifically, we'll be implementing the Normalized Fairness of Retrieval Results (NFaiRR) metric from [2], which is a little like NDCG in that it compares the actual ranking versus an idealized ranking.

You'll start by implementing Fairness of Retrieval Results (FaiRR) as follows. We'll first define a function ω that calculates how biased a document is towards one particular attribute from a specific protected class (e.g,. Gender); it doesn't matter which attribute the page has for this particular metric. Given that we have known attributes from Wikidata, we can defined ω as follows: for some document d, let $\omega(d) = 0$ if the document is associated with a protected class and $\omega(d) = 1$ if it does not.

Given some ranked list of pages L for query q,

$$\text{FaiRR}_q = \sum_{i}^{t} \omega(L_i^q) p(i)$$

where L_i^q is the document at position i in the ranked list of results for query q and $p(i)$ is the discount factor for that position. The t is the ranking cut-off for evaluating fairness; documents retrieved at a rank below t are not considered. We define the discount function for position i the same as in NDCG:

$$p(i) = \frac{1}{\log_2(i+1)}$$

The above equations will produce an estimate of how fair a system's ranking is according to the metric. However, how fair could the most-fair ranking be in practice? To calculate this, we'll calculate FaiRR_q by first sorting the ranking of the top t documents by their ω scores in descending order. This measures the Ideal FaiRR (IFaiRR).

For a specific protected class (e.g., Race), to calculate the final NFaiRR score, we compute

$$\text{NFaiRR} = \sum_{q \in Q} \frac{\text{FaiRR}_q}{\text{IFaiRR}_q}$$

Note that you would get difference NFaiRR scores for each class, since you would need to redefine ω according to which pages have attributes for that class.

■ **Problem 1.** (20 points) Implement NFaiRR

■ **Problem 2.** (10 points) What exactly is NFaiRR measuring and is it fairness? Let's try a thought exercise. Consider a few hypothetical rankings where pages associated with different attributes are placed. What happens if all the pages associated with one attributes are ranked at the top and the rest are ranked at the bottom (but still above t). What if no pages with attributes appear above t? In a paragraph describe what you think NFaiRR is measuring and whether it is accurately capturing fairness and *why*. Describe in your own words what kinds of rankings might be most fair, as measured by NFaiRR.

■ **Problem 3.** (10 points) Based on your answer to the previous question, how might we improve or extend NFaiRR to address any issues you spotted? Be specific with respect to the equations and definitions. You do not need to implement anything for this answer, nor do you need to prove your fix works. However, you do need to explain your reasoning.

■ ***Optional* Problem 1.** (0 points) Measuring fairness is hard! Can you come up with a better way? One idea is to try to take into account the expected distribution of results with respect to the protected classes. For example, if you have two groups A and B and they show up in 20% of the data and 5% of the data, you would expect some slice of the search results to have 20% of A and 5% of B (assuming that A and B are unrelated to relevance). Design a new fairness metric that compares the expected distribution of some protected attributes with the observed distribution. You can also just describe how you'd do this if you want to. Would this approach work better than NFaiRR and why?

4 Increasing Diversity in Ranking

Our current ranking methods only take relevance into account. However, users often care about many things when searching, one of them being how diverse are the results. For example, consider how satisfied you might be if the top 10 search results were all the essentially same page. One early way to introduce more diversity in the search results is to use Maximal Marginal Relevance (MMR) [1], which is a re-ordering of the search results based on how similar the documents are. In this part of the homework, we'll implement this strategy to see if adding diversity increases fairness.

Re-ranking with MMR works by starting from an initial ranking to iteratively add documents to the final result ranking based on their similarity to already-selected documents. For similarity, we'll use the bi-encoder vectors to measure two documents' cosine similarity (not the dot score here).

We'll start with an *initial* ranked list of results L_q for some query q and a set of documents S that indicates which documents have already been selected to return.[3] To select the next document to include in the ranking and add to S, we compute[4]

$$MMR = \text{argmax}_{d_i \in L \ S} \left[\lambda \text{relevance}(d_i, q) - (1 - \lambda) \left(\max_{d_j \in S} \text{sim}(d_i, d_j) \right) \right]$$

where relevance is your IR system's relevance score, sim is the cosine similarity of the two documents bi-encoder vectors and λ is an interpolation parameter.

[3] Note that in set notation, $L \ S$ means everything in L except what is in S.

[4] Note that there is a typo in the original paper due a misplaced parenthesis, so the equation there is different from the correct equation in this homework.

Typically, we add documents to S using MMR to select the next document until $|S|$ is meets threshold t. All remaining relevant documents not in S are then included in order of their original relevance (but after the documents in S). Note that the MMR-reordering is done at the very last step prior to the user seeing the results, so for L2R, this happens *after* the L2R model produces its ranking.

■ **Problem 4.** (20 points) Implement MMR in your `l2r.py` file and use a default threshold of $t = 100$.

■ **Problem 5.** (20 points) How does increasing the diversity of the search results impact fairness? Let's use our old average-rank metric and our new NFaiRR metric to assess the impact. For this problem, you will *not* be using any pseudofeedback like you did with Homework 4.

Once again, load the `person-attributes.csv` file, which contains data for all the people-related documents and the reported demographic attributes of those people in four categories. A person will have one attribute label for a category (e.g., "Lutheranism" for the "Religious_Affiliation" category). You'll want to calculate what are the 10 most common labels in each category and keep those handy for plotting.

Rank all 200K documents for the same set of queries as in Homeworks 3 and 4: "person", "woman", "teacher", "role model", "professional". If you have the results from Homework 3 saved, you can use these. **New Part:** Calculate the ranks for all 200K documents for the same 5 queries. Generate these results for four systems:

1. A L2R system with `VectorRanker` (what you produced in Homework 4)
2. A L2R system with `VectorRanker` with MMR ordering where $\lambda = 0.3$
3. A L2R system with `VectorRanker` with MMR ordering where $\lambda = 0.5$
4. A L2R system with `VectorRanker` with MMR ordering where $\lambda = 0.7$

Save all these results somewhere for analysis. We'll want to compare the effect of MMR on each system, depending on which relevant scorer was used to initially rank.

For each query, let's look at how re-ordering by MMR affected the relative ranks of people in different demographic subgroups. For each query and each demographic category, make a bar plot showing the average ranking (with 95% confidence intervals) for pages with each of the top-10 labels in that category. *You will have four bars for each label*: one for each system and *lambda* value.

Your Writing Task: In two paragraphs, write about the following two prompts.

1. Describe what you see in the 20 plots. Does re-ordering to favor diversity help reduce disparity in the relative rankings?
2. Given what you see in the results, and your knowledge of how the IR system works, how would *you* change the ranking function to reduce disparity? Be as specific as you can with respect to what is changing in the implementation. You do not need to write any code to answer this question.

5 Evaluation

How does reranking for diversity with MMR affect our L2R-based system from Homework 3? We'll test a few settings and score the outputs using MAP and NDCG. We will then

compare against the IR models from previous homework: our baseline BM25 retrieval system, Homework 2 system, and the Homework 3 system, and the best performing Homework 4 system you had that used pseudofeedback. To save time, you can (and should) use the scores from the previous assignment for this one, rather than recomputing them.

■ **Problem 6.** (20 points) Compute the MAP@10 and NDCG@10 scores for each query using the following hyperparameter settings for MMR:

1. A L2R system with `VectorRanker` with MMR ordering where $\lambda = 0.3$
2. A L2R system with `VectorRanker` with MMR ordering where $\lambda = 0.5$
3. A L2R system with `VectorRanker` with MMR ordering where $\lambda = 0.7$

Note that running these does *not* require you to re-index or retrain. You are changing the query-time behavior and can simply re-issue each query, changing the MMR λ argument to get different behavior from the same system (you do not need to change the MMR threshold).

Plot the mean MAP@10 and NDCG@10 across all queries for each system, showing the 95% confidence interval. Use a bar plot. You should have 7 bars: 4 from the previous systems, and 3 for the new configurations. If you're curious, you can try a few more λ values too to get a sense of what effect (if any) the parameter has.

In a few sentences, describe what you see. What effect does adding adding diversity have on the performance, if any? Are there particular configurations that do better/worse? Which configuration would you choose?

■ ***Optional*** **Problem 2.** (0 points) What happens if you maximize diversity? Try setting $\lambda = 0$ to return the maximally-diverse ordering. How bad is this result? (or good?)

6 Grading and Points

Answers to the questions above constitute 60 points. A correct implementation of the code is worth 40 points. Partial credit will be given wherever possible, provided you show your work.

7 Late Policy

You have **seven** free late/flex days for this course. All late days are managed through the autograder. Any submission that is after the scheduled time on Canvas (or the autograder) will use one late day. You do not need to ask permission for free late days. We intend you to use them *first* before asking for any other forms of extensions, unless you experience a serious and unexpected life event.

8 What to Submit

Autograder Submission Procedure Your implementation of all the files should be uploaded to the Homework 4 assignment in https://autograder.io. You should already be added

as a student to the course but if you do not see the course there, please notify the instructional team immediately to be added. The autograder will tell you which files you have to upload.

1. Your code to the autograder
2. A PDF or Word document with your answers and plots to all questions

Academic Honesty Policy

Unless otherwise specified in an assignment all submitted work must be your own, original work. Any excerpts, statements, or phrases from the work of others must be clearly identified as a quotation, and a proper citation provided. Re-using worked examples of significant portions of your own code from another class or code from other people from places like online tutorials, Kaggle examples, or Stack Overflow will be treated as plagiarism. **The use of ChatGPT or Co-Pilot for generating solutions is prohibited; any solution using these tools will receive an automatic zero, regardless of how much of the code these tools were used for.** The instructors reserve the right to have you verbally describe the implementation of any part of your submission to assess whether you wrote it. If you are in doubt on any part of the policy, please contact one of the instructors to check.

Any violation of the University's policies on Academic and Professional Integrity may result in serious penalties, which might range from failing an assignment, to failing a course, to being expelled from the program. Violations of academic and professional integrity will be reported to Student Affairs. Consequences impacting assignment or course grades are determined by the faculty instructor; additional sanctions may be imposed.

References

[1] Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336, 1998.

[2] Navid Rekabsaz, Simone Kopeinik, and Markus Schedl. Societal biases in retrieved contents: Measurement framework and adversarial mitigation of bert rankers. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 306–316, 2021.