SI 650 / EECS 549: Homework 2 – Building a Document Ranker

Due: Wednesday, October 12, 11:59pm

1 Introduction

Do you ever have questions like "how long does coronavirus survive on surfaces," "how do i beat Gattuso in Tales of Vesparia" or "how exactly do apps not running "in the background" receive notifications"? If so, then this assignment is for *you*. In Homework 2, you'll be building a vertical search engine (of sorts) by designing document ranking functions and testing them in three different domains: covid-19, gaming, and android software development.

The goal of this assignment is to understand how ranking works in a real retrieval system using a vector space model (VSM) of retrieval. One of the major tasks in building an IR system is to implement a basic ranking algorithm that scores documents based on their relevance to the query. This assignment has two parts where you will design different VSM rankers. In the first, you'll re-implement the BM25 ranking function and Pivoted Length Normalization methods for ranking. These algorithms are simple but effective and should help you familiarize yourself with the basic features and data you'll likely use for part two. In part two, you'll implement *your own new ranking function*. Part two is where you will likely spend most of your time

To evaluate correctness, we'll be using Kaggle to have you submit your document rankings for each query. Your rankings will be evaluated using NDCG@10 or NDCG@5, depending on the dataset. An untuned BM25 implementation will serve as a simple-yet-effective method will serve as the baseline for how well retrieval could work.

To participate, you must join three Kaggle InClass competitions that are linked to each datasets (see links in each section for when/where to submit). Please be sure choose a username that we can identify as you or specify your Kaggle username in your homework submission itself.

This assignment uses the `pyserini`library,[1] which is a modern Information Retrieval library that supports a huge variety of different functionality—including deep learning. We recommend installing `pyserini`from pip (e.g., `pip install pyserini`). However, if this gives you issues, feel free to post on Piazza or run the assignment on Google colab (https://colab.research.google.com), which is particularly recommended if you're running this on Windows.

This assignment has the following learning goals:

1. Become familiar with a major information retrieval software package (`pyserini`)
2. Gain new software development and debugging skills
3. Improve the ability to read software library documentation

[1] https://github.com/castorini/pyserini

4. Learn how to translate equations into code
5. Understand the influence of hyperparameters on the performance of BM25 and Pivoted Length Normalization
6. Understand how vector space ranking functions work through exploring new formulations of the functions
7. Learn about issues in domain transfer when applying a tuned ranker to a new type of document collection

2 Provided Code and Kaggle Setup

We have provided several files to help you get started.

- `rankers.py` — You will implement your various rankers here as subclasses of `Ranker`, which is provided in this file as well.
- `main.py` — This is basic skeleton code for running your ranking function. You can modify this code to have it write results or explore what kind of ranking your model gives you.

Ultimately, this code is just a starting point and you are welcome to rewrite any part of it, provided that we can clearly identify the three ranking methods we ask for (described next). When submitted, we'll ask for all the code you used (even if you left some of these files unmodified) so that we can check that things work.

3 Re-implement Two VSM Ranking Functions (50 points)

Your first task will be to re-implement two famous ranking functions we've talked about in class. Implementing these functions will mostly require you to figure out how to translate the equations into `pyserini`operations.

In general, you need to write a function that, for a query, computes a relevance score for every document by aggregating the weight of every word that occurs in both the document and the query:

$$s(q,d) = g[f(w_1,q,d) + ... + f(w_N,q,d), q, d],$$

where $w_1, w_2, ..., w_N$ are terms in the query q that appeared in the documents d. That is, the score of a document d given a query q is a function g of the accumulated weight f for each matched term.

Task 1: Index the documents into `pyserini`

Before you can rank, you need some way of representing the query and documents in the same vector space. The first step is then to index the data using `pyserini`'s API and write this index to a file. We recommend reviewing `pyserini`'s documentation on indexing to get started https://github.com/castorini/pyserini#how-do-i-index-and-search-my-own-documents.

We'll be using three document collections for this assignment that are provided on the Kaggle InClass competitions: You will use your code index each of the collections *separately* to create different index files.

NOTE: The indexing itself is actually very easy with `pyserini`(can be done in a single line of code), so if your method seems complicated, something has gone amiss.

Task 2: Implement Pivoted Length Normalization

Once you have the data indexed, your second task will be to implement Pivoted Length Normalization as a ranking function. We have provided some skeleton code for getting you started. You can find the formulas for Pivoted Length Normalization in the lecture slides or in the textbooks.

Most of your effort in this task will be spent learning how to use `pyserini`and access the kind of data you need to calculated the scoring function. The code itself is relatively simple so one goal is to have you familiarize yourself with their documentation and, in general, practice how to learn a new complex software library.

Once you have finished your implementation, please upload your predictions to the following Kaggle InClass for the **trec-covid** dataset:
https://www.kaggle.com/t/c9b042ef090f49f4bd70fa18d5fb4b3f. You are welcome to tune the hyper parameters of the scoring function, but that is not needed for Task 2 (but will be for Task 3!). Full credit will depend on a correct implementation.

Task 3: Implement and Tune BM25

The third task asks you to implement another slightly more complicated ranking function, BM25. You can find the formulas for BM25 in the lecture slides. You should use the definitions in the course slides, as other definitions may be different (e.g., Wikipedia's BM25 is different and missing the k_3 term).

Once you have finished your implementation, submit your rankings to the KaggleInClass for the **gaming** dataset https://www.kaggle.com/t/992a05c211b940088942205b5f3b4e5a. You will be graded based on your best performing submission. You'll get full credit if your retrieval function can beat the provided baseline in the dataset, which is an untuned BM25, which means you'll need to do some hyperparameter tuning.

4 Design Your Own Scoring Function [50 points]

The second half of the assignment will have you implement at least one retrieval function *different* from BM25, Dirichlet Prior, and Pivoted Length Normalization (and any variations of these). You

will be graded based on your best performing function. You'll get full credit if your retrieval function can beat the provided baseline in the dataset, which is an untuned BM25. By "beat," we mean that your implemented function and your choice of parameters should reach higher NDCG@5 than the baseline on Kaggle for our dataset, which you can check at any time.

Most of your time in this part of the assignment will be spent trying to see what kinds of adjustments you can make to a ranking function improve. In past years, nearly all students have managed to outperform BM25 so it is very much possible. You are welcomed to be creative in your approaches as well. The intent of this part is to help you understand how different parts of the function help or hurt ranking, which often is a very empirical and hands-on process when trying to fine-tune a ranker for a particular dataset.

In your submission, please include the code to implement the retrieval function, the parameters you used that achieved the best performance, and the best performance. In addition, ***explain*** **what your function does, how you designed in, and why you decide to choose those hyperparameter values**. Your explanation should be at least a few sentences, if not longer. You will lose points if you cannot explain why your function can reach a higher performance. You can include your explanations in the end of the submitted notebook. You should aim to explain your design in at least three sentences.

Note: Simply varying the value of parameters in Okapi/BM25, Dirichlet Prior or Pivoted Length Normalization does not count as a new retrieval function. The variations of any of the algorithms listed in the slides or in the textbooks also do not count as new functions.

Once you have completed your function, upload your results to the following Kaggle InClass, which uses the **android** dataset:
https://www.kaggle.com/t/4cd177bc66ba411192984e81466d9518. Full credit for this function depends on whether it can surpass an untuned BM25 ranking of the same dataset.

5 What to submit?

You need to submit three things:

1. Please submit your code in a runnable format to Canvas; .ipynb files are acceptable.
2. Please submit the rankings for all three ranking functions to their respective Kaggle InClass sites. Be sure the join using the link above. **Please be sure to include your Kaggle username in this file** so we can figure out which score is yours.
3. Please submit a text (pdf/Word) file that describes your choices and implementation for Part 2 on designing your own ranking function. We will need to understand this to make sense of your code.

Everything needs to be submitted to Canvas. Please do *not* submit a zip; submitting each file separately is strongly preferred.

6 Late Policy

Throughout the semester, you have three free late days total. These are counted as whole days, so 1 minute past deadline result sin 1 late day used (Canvas tracks this so it's easier/fair). However,

if you have known issues (interviews, conference, etc.) let us know at least 24 hours in advance and we can work something out. **Special Covid Times™ Policy**: If you are dealing with Big Life Stuff®, let the instructor know and we'll figure out a path forward (family/health should take priority over this course). Once the late days are used up, the homework cannot be submitted for credit, though speak with the instructor if you think this is actually a possibility before actually not submitting.

7 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. 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.

Please be aware, that we know that many implementations of BM25 and Pivoted Length Normalization exist out there—including those internal to `pyserini`. We have collected many (many) of these and are able to check your implementation against theirs (which is robust to *many* code transformations), as well as against other students. Please do your own work and do not submit code you found online (or anywhere else).