COL 764 Assignment 4 - Document Reranking using LLMs

2024 - Version 2.1

Version History

Revision	Date	Author(s)	Description
1.0	6 Nov 2024	Instructor	Initial Version.
2.0	10 Nov 2024	Instructor	Updated with the nDCG based marks assignment.
2.1	12 Nov 2024	Instructor	Fixed the BM-25 and LM based ranking confusion.

Deadline

Deadline (**absolute**) for final submission of the complete implementation, and the report on the algorithms as well as performance tuning: November 25th 2024, 09:00 PM. **Note that there is no delayed submission option for this assignment.**

Weightage

This assignment is evaluated against 90 marks. The tentative breakup of marks is given in the end of the document.

Instructions

1. This programming assignment is to be done by each student individually. Do not collaborate -either by sharing code, algorithm, and any other pertinent details- with each other.

2. All programs have to be written using Python. Anything else requires explicit prior permission from the instructor. The machine we use to evaluate your submissions has **Python version 3.12.0**. We recommend you to use same versions to avoid the use of deprecated/unsupported functions or take care to ensure required compatibility.

3. A single tar/zip of all the submission files has to be submitted. The zip/tar file should be structured such that

- upon deflating all submission files should be under a directory with the student's registration number. E.g., if a student's registration number is `20XXCSXX999` then the zip submission should be named 20XXCSXX999.zip and upon deflating **all contained files** should be under a directory named ./20XXCSXX999 only (names should be in uppercase). Your submission might be rejected and not be evaluated if you do not adhere to these specifications. **Note that we will use only Ubuntu Linux machines to build and run your assignments.** So take care that your file names, paths, argument handling etc. are compatible.

4. You *should not* submit data files, index files, dictionaries etc. If you are planning to use any other "special" library, please talk to the instructor first (or post on Teams).

5. **Note that there will be no deadline extensions. Apart from the usual "please start early" advise, I must warn you that this assignment requires significant amount of implementation effort, as well as some 'manual' tuning of parameters to get good speed up and performance. Do not wait till the end.**

1 Assignment Description

In this assignment the goal is to evaluate the performance of large language models (LLMs) when used in conjunction with small set of "context" query-document pairs from a pseudo-relevance set. The idea is to augment the query (or "expand" the query) with a LLM-generated relevant passage and use it in classical retrieval strategy. We will use a well-known TREC benchmark for this purpose.

1.1 Data Description

Document Collection: There are $3,213,835$ unique documents in the collection given to you in the form of a tab-separated file consisting of 4 fields: `doc_id`, `url`, `title` and `body`.

Queries: We will consider 45 topics from the TREC track. Each topic is indexed by a `query_id` contains a keyword `text` that is to be treated as the initial query fired by the user.

Pseudo-relevance Collection: Unlike previous assignment where you were supposed to build an index and a retrieval system, in this task you are only required to *rerank the top-100 documents* that are already retrieved through an initial model from the collection. **For each query, you will be provided upto 100 top-ranked documents that were retrieved.**

No Inverted Indexes Please!!

In this assignment we are not looking for an implementation of index or an end-to-end retrieval system. You only need to submit your implementation of "reranking" of the given top-100 results for each query. For this purpose, should you need to process the document collection (to collect some statistics), then you can compute it using any method and use it.

Filename	Size	Records	Description
docs.tsv.gz	7.9GB	3,213,835	**tsv**: doc_id, url, title, body
example-queries.tsv	999 Bytes	24	**tsv**: query_id, text
test-queries.tsv	968 Bytes	21	**tsv**: query_id, text
top100docs.tsv	112,695 Bytes	4,500	**tsv**: query_id, doc_id, score
qrels.tsv	182,852 Bytes	9,098	**tsv**: query_id, doc_id, relevance, iteration

Table 2: Details of the files released for this assignment. All files are made available in the following directory on HPC: `/scratch/cse/faculty/srikanta/COL764-A4-2024`, **except** the docs.tsv.gz which is available from `/scratch/cse/faculty/srikanta/COL764-A2-2024`.

2 Task

The goal is to rerank documents (similar to A2). However, you will use the Llama family of LLMs available on Groq [1] to expand the query before reranking the documents. To expand the given query, we will follow the idea proposed by query2doc [Wang et al., 2023]. The idea is to first generate *short* pseudo-documents by zero-/few-shot prompting of LLMs, and then expand the query with terms from the generated pseudo-documents. The assumption is that the pseudo-documents thus generated are based on the vast background information assimilated by the LLMs, and are thus consist of contextually relevant terms.

The way to elicit response from LLMs is to prompt the LLM by specifying the task, and then providing the necessary information for the LLM to perform the task on the given input. Often, you can provide good set of example context and responses to the LLM at the beginning to help LLM to adapt its response. Note that the quality of response from the LLM depends on two crucial factors (among many others):

(a) the specification of the task in the prompt (we will denote it as PROMPT in the rest of the document).

(b) the quality and the number examples that are provided as guidance to the LLM.

In this assignment, you are expected to experiment with both of these factors in order to improve the retrieval quality.

A typical prompt for this assignment will be as follows:

```
PROMPT

Query: <example_query>
Passage: <example_passage>
...
```

[1] Visit `https://console.groq.com/login` and create a free account. You will be able to create a API key, which can be used to invoke LLM through a Python program.

```
...
Query: <query>
Passage:
```

2.1 Selection of Example Query-Passage Pairs

For few-shot examples, you will utilize a small set of training queries and experiment with different approaches to passage selection. For each query, you will use $k = 0, 1, 2, 3, 4$ number of example passages to construct k-shot prompting.

We can preprocess the given set of training queries and their initial retrieval set for constructing a collection of query-passage pairs from which we can select the k examples to be included in the prompt.

2.1.1 Preprocessing

For each training query, you are required to generate **at least** 3 example passages, and experiment with the following approaches for forming the example passages:

1. **Random passage selection:** Select a random passage from any document that is marked as relevant for the given query. The length of the randomly selected passage can be limited to 1-4 sentences only.

2. **Diverse passage selection:** For the given query, select passages that are most diverse from each other. Diversity measured in terms of the cosine angle between the term-vector representation of passages — i.e., two passages a and b are more diverse than a and c if $\cos(a, b) > \cos(a, c)$. The first passage selected will be the first 3 sentences of the highest ranked document in the initial retrieved set.

Thus, at the end of this step, you have **two separate collections** of `(query, passage)` pairs for all training queries, with each training query associated in at least 3 distinct pairs. The next step will be described for one such example collection, similar steps to be followed separately for the other example collection.

2.1.2 Selection for the prompt

For each query, you will select $k = 0, 1, 2, 3, 4$ example pairs from one of the `(query, passage)` constructed above. The selection process is as follows:

1. **Random:** As the name suggests, randomly select the example pairs from the collection.

2. **Disjoint:** Consider examples from only those example queries which have no overlapping term with the given query or overlapping document in their top-100 documents of the given query.

2.2 Final Result Ranking

We will use the standard retrieval model with BM-25 as the underlying scoring model. You are free to tune the hyper-parameters of the model, but you must report them.

Note that you are not expected to implement the retrieval model that works on the entire corpus (so no need for full-fledged inverted indexing). The corpus is provided only for assistance in (a) getting actual document content from the document identifier in the top-100 documents file, (b) to estimate the term weight and other statistics that you may need for implementing the BM-25 retrieval model. You are only required to implement the retrieval model required for *reranking* the top-100 documents that are provided for each query. **You may assume that the `query-file` contains only those queries whose top-100 documents are available in the `top-100-file`** (see Section 2).

2.2.1 Query

Note that since the initial query is much shorter than expansion terms that would generated as part of the passage, we will concat the initial query 5 times along with all the expansion terms for forming the query which will be processed. In other words, if the original query was $q =$ `"who killed nicholas ii of russia"` then the query that will processed will be $Q =$`"who killed nicholas ii of russia who killed nicholas ii of russia who killed nicholas ii of russia who killed nicholas ii of russia who killed nicholas ii of russia"` $\odot$ `<expansion terms>`, where $\odot$ represents the concatenation operator. Your retrieval model must take into account repeated query terms appropriately.

2.3 Metrics

We will focus on determining the best choice for improving the following metrics — nDCG@5, 10, 50.

2.4 Submission Structure

You are required to submit the following:

Name	Description
report.pdf	A single PDF file that contains details of your implementation, hyper-parameter choices, best results on the test queries given to you under different preprocessing–prompt-selection strategy combinations.
rand_examples.csv	A comma-separated file containing `(query, passage)` example pairs constructed using the Random passage selection strategy.
div_examples.csv	A comma-separated file containing `(query, passage)` example pairs constructed using the Diverse passage selection strategy.
rand_prompt.csv	a comma-separated file containing `(query, prompt, result)` containing the query, the prompts constructed (with different values of k), the result passages generated by the LLM under Random prompt selection strategy. The first row for a query is for $k = 0$, thus the prompt field of that row will be empty.
disj_prompt.csv	a comma-separated file containing `(query, prompt, result)` containing the query, the prompts constructed (with different values of k), the result passages generated by the LLM under disjoint prompt selection strategy. The first row for a query is for $k = 0$, thus the prompt field of that row will be empty.

You are **NOT** required to submit the implementation files. However, you must retain them since we may randomly select some submissions to be demonstrated.

2.5 Tentative breakup of marks assignment

In general, a submission qualifies for evaluation if and only if it adheres to the specifications given above

report	30
random example pairs file	10
diverse example pairs file	20
random prompts file	15
disjoint prompts file	15

We will use ndcg@5, 10, 50 in the overall evaluation of the submissions as follows: In the report component, we will use 10 marks for identifying the best performing model submission. If you are the top-2 of ndcg@5, 10 or 50 (any one) you will be awarded full 10 marks, if you are in the next-5 you will be awarded 7 marks, next 5 will be awarded 4 marks, and rest will get 3 marks (if they are reported). Otherwise, you will get 0 on 10.

References

[Wang et al., 2023] Wang, L., Yang, N., and Wei, F. (2023). Query2doc: Query expansion with large language models. In Bouamor, H., Pino, J., and Bali, K., editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9414–9423, Singapore. Association for Computational Linguistics.