COL 764 Assignment 2 - Document Reranking Task

2023 - Version 1.2

Version History

Revision	**Date**	**Author(s)**	**Description**
1.0	28 Sep 2023	Instructor	Initial Version.
1.1	03 Oct 2023	Instructor	Fixed the instructions for LM and W2V tasks.
1.2	11 Oct 2023	TA	Added URL.

Deadline

Deadline for final submission of the complete implementation, and the report on the algorithms as well as performance tuning: October 18th 2023, 11:59 PM.

NOTE: The deadline has been changed keeping in mind the HPC downtime. It is very likely that Assignment 3 will be released before the submission deadline of this assignment. Please plan accordingly.

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 either using Python/Java/C/C++ programming languages only. Anything else requires explicit prior permission from the instructor. The machine we use to evaluate your submissions has the following versions:
   - **C**, **C++**: gcc12;
   - **Java**: java 11 (we will not use OpenJDK);
   - **Python:** version 3.7.

   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 the source code 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.

- apart from source files, the submission zip file should contain a build mechanism *if needed* (allowed build systems are Maven and Ant for Java, Makefile for C/C++). It is the responsibility of each student to ensure that it compiles and generates the necessary executable as specified. **Please include an empty file in case building is not required** *(e.g. if you are using Python)*. **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 different models of pseudo-relevance feedback –including query expansion and reranking– aimed at improving the precision and recall of results. We will use data collection and queries from TREC COVID track. Specifically, this assignment will use the CORD19 data release from 16th July 2020. It is available for download from `https://csciitd-my.sharepoint.com/:f:/g/personal/csz208507_iitd_ac_in/Eh9EIEwDC75EmWQKXO4sBqABZcJ-PAC2AmZayCLYq3t9KQ?e=0o2exj`. **Do not use later/older versions since they are significantly different.**[1]

TREC COVID19

TREC COVID track ran for multiple years, and unlike the usual TREC tracks had multiple rounds within a year keeping in mind the urgency of developing techniques to search through the COVID related mountain(s) of research articles that were being produced. Thus it played (and continues to play) an important role in the global effort to deal with this pandemic.

1.1 Data Description

Document Collection: There are $191,175$ unique CORD identifiers in the collection, with each CORD id corresponding to one or more full-text files that are extracted from article publisher website or from PubMed Central's (PMC) curated text site of the article. Metadata corresponding to all CORD ids are listed in a file called `metadata.csv`. The format and the description of this file is given at `https://github.com/allenai/cord19/blob/master/README.md`.

Queries: We will consider only 40 topics from the TREC COVID19 track. Each topic –indexed by a `topic number` consists of the following three fields: `query`, `question`, and `narrative`.

Which Field to Use?

You are free to use any one of these fields in your assignment – but note that it should any ONE field.

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 on the full-text index of the collection. Thus no stop-words were removed from the documents or the queries. **For each topic, you will be provided upto 100 top-ranked documents.**

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.

[1]We can download the dataset and make it accessible on HPC once it comes back up

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.

2 Task

Starting from the top-100 documents for each query, you are supposed to implement the following methods for reranking them to get higher-relevant documents in the higher ranks.

How did we get these top-100?

It is immaterial how these top-100 candidates were generated (i.e., which specific retrieval model was used). Also, you are free to use any form of document preprocessing – i.e., tokenization, lemmatization, stop-word elimination, entity recognition, etc.– as long as they provide better retrieval quality.

2.1 Task 1: Relevance Model based Language Modeling

Use Lavrenko and Croft's relevance models with an Unigram model with Dirichlet smoothing. In particular, you should recall that Lavrenko and Croft presented two models, we will call them **RM1** and **RM2**, for incorporating relevance information into the language modeling. In RM1 they use i.i.d. sampling, and in RM2 they suggest the use of conditional sampling. You are required to submit your implementation of **RM1** only.

Consider the top-100 documents as the candidate pseudo-relevance models. Again, you are free to tune μ hyperparameter (and any other parameter) to achieve higher retrieval quality. Please report **every** parameter value used in your implementation.

2.2 Task 2: Use of Local Embeddings for Query Expansion

In [Diaz et al., 2016], authors introduced the idea of using *local embeddings* (i.e., locally computed word embeddings) to expand the query and then do the reranking using these embeddings. Specifically, their approach can summarized as follows:

1. Train Word2Vec embeddings based on the initially retrieved top-k documents (the pseudo-relevance set).
2. Use these embeddings to compute the nearest words to the original query terms, and use top$N$ expansion terms from that set.
3. Rerank the documents using the expanded query and present the results.

You can use the word2vec implementation from `https://code.google.com/archive/p/word2vec/`. Please refer to the paper for more details of the model and implementation details. You are limited to using not more than 20 expansion terms (you can use less, but not more).

2.3 Evaluation Metrics

The qrels for training topics we have shared are judged on a 3-point relevance grade: (0: non-relevant, 1: partially relevant, and 2: relevant). We will use **nDCG@**$\{5, 10, 50\}$ for evaluation.

2.4 Program Structure

You are expected to submit two implementations:

1. Language Model -
   `lm_rerank.sh [query-file] [top-100-file] [collection-dir] [output-file]`

2. Word2Vec Model –
   `w2v_rerank.sh [query-file] [top-100-file] [collection-dir] [output-file] [expansions-file]`

where,

Scripts	
`lm_rerank`:	bash script file
`w2v_rerank`:	bash script file
INPUT files	
`query-file`:	file containing the queries in the same xml form as the training queries released
`top-100-file`:	a file containing the top100 documents in the same format as train and dev top100 files given, which need to be reranked
`collection-dir`:	directory containing the full document collection. Specifically, it will have `metadata.csv`, a subdirectory named `document_parses` which in turn contains subdirectories `pdf_json` and `pmc_json`.
OUTPUT files	
`output-file`:	file to which the output in the trec_eval format has to be written
`expansions-file`:	specifies the file to which the expansion terms used for each query should be output.

2.5 Output formats

We will make use of trec_eval tool for generating nDCG scores. Therefore, it is important that you follow the exact format specification as required by the tool. To reiterate – white space is used to separate columns. The width of the columns in the format is not important, but it is important to have exactly six columns per line with at least one space between the columns.

Format of Output-File:

Lines of results are of the following form

```
1 Q0 pid1    1 2.73 runid1
1 Q0 pid2    1 2.71 runid1
1 Q0 pid3    1 2.61 runid1
1 Q0 pid4    1 2.05 runid1
1 Q0 pid5    1 1.89 runid1
```

where:

- the first column is the topic (query) number.
- the second column is currently unused and should always be "Q0".
- the third column is the identifier of the retrieved document in context of document ranking task.
- the fourth column is the rank the passage/document is retrieved.
- the fifth column shows the score (integer or floating point) that generated the ranking. This score must be in descending (non-increasing) order.
- The sixth column is the ID of the run you are submitting.

2.5.1 Output format for expansions-file

We will use the following simple format to output the expansion terms used for each query:
`<topic number> :  <expansion term1>, ...  <expansion term20>`
The output file specified in `expansions-file` will consist of one such single line for each topic.

2.6 QRels for Tuning

In order for you to tune various hyper-parameters, we will release the benchmark query relevances as QRel file for all the queries we have released. You are allowed to use these true relevance judgements only to tune your hyper-parameters.

List of Files

Here is the complete list of all files that will be needed as part of the assignment. Those highlighted in yellow are part of the dataset download from CORD19 website (see Section 1). Remaining files can be downloaded from `http://www.cse.iitd.ac.in/~srikanta/course/col764-2021/col764-a2-release.tgz`

No.	Filename	Description	Size
1	metadata.csv	Contains the *golden* values of `cord_id`, `abstract`, `authors`, `file` etc. For details about this file, you should read the CORD19 website (see Section 1.1).	192, 510 rows (including header)
2	document_parses.tar.gz	Contains original benchmark JSON documents in two directories `pdf_json` and `pmc_json`. You should use the `pmc_json` contents when available, there may be some documents which have no file in `pmc_json` in which case you should use file from `pdf_json` – all of their text in both cases.	62, 736 files in `pmc_json`, and 84, 420 files in `pdf_json`.
3	covid19-topics.xml	Contains topics to be used in the assignment – `query`, `description` and `narrative` fields.	40 topics
4	t40-top-100.txt	top-100 cord_ids for each topic in the covid19-topics.xml above. Note that all the cord_ids have been verified and can be found in the metadata.csv file (and through that you can locate their corresponding JSON file)	4000
5	t40-qrels.txt	Qrels file containing the relevance judgements for the 40 queries. The format of each line (as with any standard TREC Qrels) is: `<qid> <run> <cord_id> <rel>`. You should ignore `<run>` entry. Relevance, as mentioned in Section 2.3, is judged as follows: judgment is 0 for not relevant, 1 for partially relevant, and 2 for fully relevant.	59, 746 rows for 40 queries.

2.7 Submission Plan

All your submissions should strictly adhere to the formatting requirements given above.

- Submission of your source code on 18th October, and the final version of results.
- You should also submit a README for running your code, and a PDF document containing the implementation details as well as the resulting nDCG scores with various parameters that have been specified (preferably as a table).
  Name them README.{txt|md} **and** 20XXCSXX999.pdf **respectively**.
- The set of commands for evaluation of your submission roughly follows -

Tentative Evaluation Steps (which will be used by us)

```
$ unzip 20XXCSXX999.zip
$ cd 20XXCSXX999
$ bash build.sh
$ bash lm_rerank.sh <arguments>
$ bash w2v_rerank.sh <arguments>
```

- Here is the link to TREC EVAL tool. `https://trec.nist.gov/trec_eval/`
- Only `BeautifulSoup`, `lxml`, `pandas`, `numpy/scipy`, `nltk`, `PorterStemmer` and libraries distributed with Python (e.g. `re`) will be available in the Python evaluation environment. You can also assume that NLTK libraries that are available through `''nltk.download('popular')''` are already downloaded. Include the source code of any imports in your submission for other languages (after getting prior permission from the instructor). No external downloads will be allowed for any language environment at runtime – i.e., there may not be internet connectivity during runtime, so tread carefully. If in doubt, please post it on the Teams channel.

2.8 Tentative breakup of marks assignment

In general, a submission qualifies for evaluation if and only if it adheres to the specifications given above (arguments, structure, use of external libraries, correct output format, input format adherence, etc.). Given this requirement, the marks assignment for correct implementation of:

Relevance LM	25
Word2Vec reranking	25
README and algorithmic details documentation	15
Report	25
Total	90

References

[Diaz et al., 2016] Diaz, F., Mitra, B., and Craswell, N. (2016). Query expansion with locally-trained word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.