Assignment 1

2021 - Version 2.0

Deadlines

1. Deadline for final submission of the complete implementation, and the report on the algorithms as well as performance tuning: 10^{th} March, 2021, 11:59 PM
2. The form for submitting self-reported metrics on the training data will be opened from 28^{th} Feb 2021.

Weightage

This assignment is evaluated against 100 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.
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 tarball/zip submission should be named 20xxcsxx999.{tgz|zip} and upon deflating **all contained files** should be under a directory named ./20xxcsxx999 only. If this is not followed, then your submission will be rejected and will not be evaluated.
   - apart from source files the submission tarball/zip file should contain a build mechansim 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 executables as specified. **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. If you are planning to use any other "special" library, please talk to the instructor first (or post on Teams).
5. Note that your submission will be evaluated using larger collection. Only assumption you are allowed to make is that all documents will be in English. The overall format of documents will be same as given in your training. The collection directory will be given as an input (see below).
6. **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

Information retrieval (IR) has been an integral part of the search engines for ranking and recommendation. IR has overlapping areas in various fields such as Extreme Classification (link), NLP for Q/A pipeline. In this assignment we will be experimenting with algorithms from all these domains and apply them in information retrieval scenarios. For this assignment we will be using author's implementation of **Parabel** (link) from extreme classification, **Latent Dirichlet Allocation** (link) followed by **logistic regression** (link) which is a "starting point" for neural methods for NLP and **BM25** (link) from classical IR methods. Note you need not re-code entire project and you are free to use any other **open source** library for the purpose of this assignment but *cite them properly* in your report. Failing this can lead to severe actions as seem fit by the appropriate authority.

1.1 Task

We will be working with MS-Marco available at HPC on the following path:
`/scratch/cse/faculty/srikanta/MSMARCO-DocRanking`.
Task is to retrieve relevant passages for a question. Read more at this (link). For each algorithm task is

1. Build the dataset pipeline to feed the training as well as test dataset to the algorithms. Here are a few pointers:
   (a) Parabel uses TF-IDF weighted Bag of Words as the input vectors. Refer to dataset format at the extreme classification repository (link).
   (b) You can use any tokenizers of your choice but we recommend NLTK.
2. Pseudo relevance feedback (PRF): Using top-k predicted documents re-write the queries in order to improve evaluation metrics. You can use any expansion technique of your choice (***except Rocchio's***). Mention what technique you used in the report and what were your gains over the main method. Experiment with k =10, 50, 100 and 1000.

1.2 Program Structure

You are required to provide us with the following files:

1. **Requirements file** File should be named as `requirements.yml`. This will be used to setup the environment to run your code and environment name should be **InfoRetri**. After setting up your conda environment with the given name, use the following command to generate your `requirements.yml`
```
conda activate InfoRetri
conda env export > requirements.yml
```
2. **Setup** file. There should be another `setup.py` file which will be used to download all necessary files for your program to run. Here is how it should work.
```
python setup.py
```
3. **Run** file. There should be another `run.sh` file which will process the dataset and run the desired algorithm. It takes the following four arguments.

`<Algorithm>`	string of characters indicating the algorithms. Options are [`parabel`‖`lda`‖`bm25`]
`<data directory>`	path to dataset for eg. "`/data/MS-MARCO`". Notice no back slash at the end of the path
`<test file>`	test queries for eg. "`/data/MS-MARCO-test.tsv`".
`<result directory>`	path to store output results for eg. "`/results/MS-MARCO`". Notice no back slash at the end of the path
`top k`	Top k most relevant results to retrieve and re-rank using PRF. eg. 100
`<output file name>`	file name to store the final rankings. eg. `retrieved.txt`

Here is how your program should work.

```
./run.sh parabel /data/MS-MARCO /data/MS-MARCO-test.tsv /results/MS-MARCO 100 re-
trieved.txt
```

This should train parabel on MS-MARCO dataset and generate a text file consist of results in `retrieved.txt` for queries in "MS-MARCO-test.tsv" (format same as training) in the following format.

Note: You can have any number of files consisting of codes in your submission folder but we will need above three files to test your scripts.

Result Output Format: We will make use of trec_eval tool for generating nDCG@10 and Precision@10 scores. Therefore, it is **very** important that you follow the exact format that is required by the tool. The trec_eval tool is freely downloadable from `https://github.com/usnistgov/trec_eval`. The result file format instructions are as follows:

Lines of results_file are of the form

```
030 Q0 ZF08-175-870 0 4238 prise1
qid iter docno rank sim run_id
```

giving TREC document numbers (a string) retrieved by query qid (a string) with similarity sim (a float). The other fields are ignored, with the exception that the run_id field of the last line is kept and output. In particular, note that the rank field is ignored here; internally ranks are assigned by sorting by the sim field with ties broken deterministicly (using docno). Sim is assumed to be higher for the docs to be retrieved first. File may contain no NULL characters. Lines may contain fields after the run_id; they are ignored.

[taken from the comments in `format.c` file in the trec_eval repository].

You will be supplied with a test document collection, queries and qrels not later than 5^{th} March 2021.

1.3 Submission Plan

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

- Submission of your source code on **Moodle**, as well as 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 any tuning you may have done.

Leaderboard: We will have a form-based submission of Precision@10 and nDCG@10 scores starting from 28^{th} Feb 2021. You can submit it as many times as you want (a subset of these metrics as well), and the leader board of submissions will be continuously updated and shared to all.

1.4 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:

Parabel	15
Latent Dirichlet Allocation	15
BM25	15
Pseudo relevance feedback + re-ranking	10×3
README and algorithmic details documentation	10
Leaderboard gains	15
Total	100