COMP3009J – Information Retrieval
Programming Assignment

This assignment is worth **30% of the final grade** for the module.
Due Date: Sunday 29th May 2022 at 23:55 (i.e. beginning of Week 15)

This assignment asks you to write a simple Information Retrieval system. For this, two corpora have been provided on Brightspace. These are named ``COMP3009J-small-corpus.zip'' and ``COMP3009J-large-corpus.zip''. The larger corpus is also hosted on Weiyun, in case you have difficulty downloading from Brightspace.

When you extract these zip files you will find several files that you will need to complete this assignment. The README.md file in each describes the files contained in the archive and their format[1].

Both corpora are in the same format, **except for the relevance judgments**. The small corpus does not have unjudged documents.

You should first try the following tasks using the small corpus. If you can do this successfully, then you should move to the large corpus. The large corpus will require more efficient programs.

Part 1: BM25 Model

For this assignment you are required to implement the **BM25** of Information Retrieval. You must create a program (using Python) that can do the following.

1. **Extract the documents** contained in the document collections provided. You must divide the documents into terms by preprocessing in an appropriate way. The strategy must be documented in your source code comments.

2. Perform **stopword removal**. A list of stopwords to use is contained in the stopwords.txt file that is provided in the ``files'' directory.

3. Perform **stemming**. For this task, you may use the porter.py code provided in the ``files'' directory.

4. The first time your program runs, it should create an appropriate **index** so that IR using the BM25 method may be performed. Here, an index is any data structure that is suitable for performing retrieval later.

[1] This is a Markdown file. Although you can open and read it as plain text, a Markdown editor like Remarkable (https://remarkableapp.github.io/ - Windows or Linux) or MacDown (https://macdown.uranusjr.com/ - macOS) is recommended.

This will require you to calculate the appropriate weights and do as much pre-calculation as you can. This should be stored in an external file in some human-readable format. Do not use database systems (e.g. MySQL, SQL Server, etc.) for this.

5. If an index file has already been created, your program should load the index from this file, rather than processing the document collection again.

6. Your program should accept a query on the command line and return a list of the 15 most relevant documents, according to the BM25 IR Model, sorted beginning with the highest similarity score. The output should have three columns: the rank, the document's ID, and the similarity score. A sample run of the program is contained later in this document. The user should continue to be prompted to enter further queries until they type "QUIT".

It is **ESSENTIAL** that this can be run as a standalone program on the command line, without requiring an IDE such as IDLE, PyCharm, etc. A graphical user interface (GUI) is not required.

You can assume that your program will be run in the same directory as the README.md file (i.e. the current directory will have the ``documents'' and ``files'' directories in it). Do not use absolute paths in your code.

Non-standard libraries (other than the Porter stemmer provided) may not be used. You may use standard libraries (i.e. any libraries that are bundled with Python by default).

Part 2: Evaluation

For this part, your program should use the standard queries that are part of the corpus provided to evaluate the effectiveness of the BM25 approach. The user should be able to select this mode by changing how they run the program.

For example, if users want to enter queries manually, they should be able to run (assuming your program is called search.py):

```
python search.py -m manual
```

Or to run the evaluations:

```
python search.py -m evaluation
```

For the evaluations, the standard queries should be read from the ``queries.txt'' file (in the ``files'' directory). This file has a query on each line, beginning with its query ID.

An output file should be created (named ``output.txt''). The format is described in the README.md files.

After creating this file, your program should calculate and print the following evaluation metrics (based on the relevance judgments contained in the ``qrels.txt'' file in the ``files'' directory): For each metric, your program should output the average score for all the queries provided.

- Precision
- Recall
- P@10
- R-precision
- MAP
- bpref
- NDCG

What you should submit

Submission is through Brightspace. You should submit a single .zip archive containing:

- The .py file with the source code for your program. If you have attempted to deal with both the small and large corpora, you can submit a separate program to deal with each corpus.
- A README.txt or README.md file that describes what your program can do and how to run it.

Contact

If this specification is unclear, or you have any questions, please post in the Brightspace forum or contact me by email (david.lillis@ucd.ie).

Sample Run (Manual)

```
$ python search.py -m manual
Loading BM25 index from file, please wait.
Enter query: library information conference

Results for query [library information conference]
1 928 0.991997
2 1109 0.984280
3 1184 0.979530
4 309 0.969075
5 533 0.918940
6 710 0.912594
7 388 0.894091
8 1311 0.847748
9 960 0.845044
10 717 0.833753
11 77 0.829261
12 1129 0.821643
13 783 0.817639
14 1312 0.804034
15 423 0.795264

Enter query: QUIT
```

Note: In all of these examples, the results, and similarity scores were generated at random for illustration purposes, so they are **not** correct scores.

Sample Run (Evaluation)

```
$ python search.py -m evaluation
Loading BM25 index from file, please wait.

Evaluation results:
Precision:    0.1382
Recall:       0.4124
P@10:         0.6210
R-precision:  0.3429
MAP:          0.2532
bpref:        0.5632
NDCG:         0.4119
```

Grading Rubric

Below are the main criteria that will be applied for the major grades (A, B, C, etc.). Other aspects will also be taken into account to decide minor grades (i.e. the difference between B+, B, B-, etc.).

- Readability and organisation of code (including use of appropriate functions, variable names, helpful comments, etc.).

- Quality of solution (including code efficiency, presence of minor bugs, avoiding absolute paths,etc.).

Questions should be sent to david.lillis@ucd.ie or posted in the Brightspace forum.

Passing Grades

``D'' Grade
Good implementation of the primary aspects of Information Retrieval, using the small corpus. This includes extracting the documents from the document collection, preprocessing (stemming and stopword removal), indexing and retrieval. The solution may contain some implementation errors. It is clear that the student has correctly understood the Information Retrieval process.

``C'' Grade
Good implementation of the primary aspects of Information Retrieval, using the small corpus. The program can also save and load the index to/from an external file, as appropriate.

``B'' Grade
Correct implementation of all sections of the assignment using the small corpus (some minor implementation errors will be tolerated). It is clear that the student has understood both the information retrieval process and the evaluation process.

``A'' Grade
Excellent implementation of all sections of the assignment, including choice of appropriate efficient data structures and efficient programming. The efficiency of the programs will be measured using the large corpus. In particular, a response to a query must be returned in a reasonable amount of time, although efficiency is important in indexing also.

Failing Grades

``ABS'' Grade
No submission received.

``NM'' Grade
No relevant work attempted.

``G'' Grade
Wholly unacceptable, little or no evidence of meaningful work attempted.

``F'' Grade
Some evidence of work attempted, but little (if any) functionality operates in the correct manner.

``E'' Grade
Clear evidence that work has been attempted on implementing retrieval using BM25, but there are serious errors in implementation, or in understanding of the process.

Other notes

1. This is an *individual* assignment. All code submitted must be your own work. Submitting the work of somebody else is **plagiarism**, which is a serious academic offence. Be familiar with the UCD Plagiarism Policy and the UCD School of Computer Science Plagiarism Policy.

2. If you have questions about what is or is not plagiarism, ask!

Document Version History

v1.0: 2022-04-30, Initial Version