Homework Assignment № 2

Prof. Joao Palotti (jpalotti@andrew.cmu.edu), CMU-Q Due to **6/Apr/2017**

1 Introduction

Search engine technology heavily relies upon evaluation. This means that there is no theory which states that a *method A* is always better than a *method B*. For a given task or a given dataset, a method might perform better than another. In order to measure which is the best method for each case, we often have to implement both methods A and B, and test both of them. Usually, we have much more than only two methods to implement. For example, during the search engine lectures, we saw a large number of retrieval methods: Boolean Search Model, Vector Space Model, BM25, Language Models...

2 First Part: implementing retrieval models (20 points)

In this homework, you will have to implement as many methods as possible. As I do not want to impose which method you will implement, you are free to choose any method from the list below. The maximum score that you will get is 20 points. For example, if you implement BM25 and VSM (and both of them work very well), your final score for the first part of this assignment is 8 out of 20 points. If you implement all method below, you will get 20 points, but you will make your professor very happy.

1. **OR Boolean Search (3 points)**: this is a variant of Boolean search that applies a OR operator for each token in the query. There is no need to write query operators. For example: a query like "doha qatar" is automatically interpreted as being "doha OR qatar".
2. **AND Boolean Search (2 points)**: this is another variant of Boolean search that applies a AND operator for each token in the query. Very much like the previous one, it is not necessary to write query operators. If you decided to implement the OR Boolean Search, implementing this one is trivial, therefore it is worth only 3 points.
3. **FULL Boolean Search (5 points)**: this is a full operational Boolean search. Valid queries now need to be expressed with all Boolean operators. Then queries look like "qatar AND location AND NOT(arabia OR saudi)". Although this method is not the most useful one, it is the hardest one to implement.
4. **Vector Space Model (4 points)**: the VSM with cosine similarity. Remember that you will need to save document length that is not only the number of tokens a document has!
5. **BM25 (4 points)**: the BM25 formula. You will need to save document length (now, it is the number of tokens in the document) and the average document length in your collection.

6. **Language Model with Laplace smoothing (4 points)**: the simplest LM model. You just need to save document length as for BM25.
7. **Language Model with Jelinek-Mercer (5 points)**: additionally to LM-Laplace, you will need to calculate P(t| wholeCollection).

3 Second Part: creating runs (5 points)

Right now, we have a number of search models implemented, but we still do not know which one is the best one. We still need to evaluate them. A useful terminology to know is that every single variant of a search engine is a **system**. In other words, we want to know which system is the best one.

In order to do so, we need a list of representative queries (often referred as **topics**) made by users of our system. The queries that we are going to use were provided by ourselves and are saved in *queries.txt* file on GitHub. When we execute each one of the queries in our system we create a specific file called **run**, i.e., a **run** is the output of our system for a set of queries.

We will create **three runs** in this homework, i.e., you will run three different variations of your search engine implementations and output the **top 100 documents** for each query/topic for each system variant.

For example, your System 1 could be an implementation of BM25 removing stopwords and punctuation, processed with Porter stemming. You system 2 could be a VSM instead of BM25, but with the same preprocessing steps. Your system 3 could be another BM25, but this time with Snowball stemming instead of Porter. Note that for your Systems 1 and 3 are both BM25, but they have different stemming steps. You would have to created two different indices of your collection as the stemming is done in the indexing step. With your systems 1 and 3, you would be able to decide which stemming is the best one to use with BM25.

You are allowed and encourage to be creative. Three extra points will be awarded to very creative systems (think for example of merging the output of different models, would an ensemble approach be a good one?). The analysis of these runs will be made in your next homework.

3.1 Run Format

Unfortunately, we have to follow a specific format for your runs. We will use the TREC format. Each line of a file in the TREC format follows this pattern:

$$< topicId, Q0, docName, docRank, score, runName >$$

where, *topicId* is a unique number for each query, *Q0* is just the string “Q0”, *docName* is the name of the retrieved document (simple000000x.txt), *docRank* and *score* are the rank of that document and the score attributed by your system. Finally, *runName* is the friendly name that you gave to your system. We could use as system names your first name followed by the number of your system: joao1, joao2, joao3.

An example of output is following the TREC format is:

```
1 Q0 simple000327.txt 1 1.00 joao1
1 Q0 simple000837.txt 2 1.00 joao1
2 Q0 simple000459.txt 1 8.00 joao1
2 Q0 simple000391.txt 2 3.00 joao1
2 Q0 simple000669.txt 3 3.00 joao1
2 Q0 simple000675.txt 4 2.00 joao1
2 Q0 simple000313.txt 5 2.00 joao1
```

Here, there are two queries/topics. For topic 1, two documents were returned and for topic 2, 5.

4 Provided Code

As in the previous homework, I am providing working code examples (check the course GitHub page). In special, I am providing a fully working retrieval model (*DohaQatarTFModel*) that outputs document rankings already in the TREC format. Run the *main_searcher.py* script to see it.

5 Deliverable

It is expected as the deliverable for this homework a zip file containing:

1. The whole code that you created to implement the search models;
2. A short documentation with your name, AndrewID and a brief explanation of which methods you implemented and what did you modify in your code for HW1.
3. The three runs as described in Section 3.

Send your zip file to my email address (jpalotti@andrew.cmu.edu).