**Assignment**

There are two primary objectives of this assignment:

1. To get started with the necessary tools and techniques required to work with unstructured text corpus.
2. To have a first-hand building experience of an information retrieval system.

You are free to work on the assignment using any programming language and any open-source text processing libraries or toolkits. A few popular libraries are mentioned in the end. However, everyone is encouraged to use Python programming language, as it is easy and widely used.

**Corpus Details:** ​ English Wikipedia in partially processed form and divided into approx 1500 files is available

at <https://drive.google.com/drive/folders/1ZsnuEm7_N6aUwhjFpv-TZXFt4DiYex4t?usp=sharing>. For this assignment, you can randomly choose any one file from the sub-folders available at the mentioned link. A single file in the corpus contains multiple documents. These documents have a start and the end tag as:

start-tag: <doc id="Document id" url="Wikipedia URL" title="Document title">

end-tag: </doc>

All the text present in between the start and the end tag is associated with the same document.

Note that the document contains some URLs in HTML format, as illustrated below:

<a href="electrical%20engineering">electrical engineering</a>

For text analysis and inverted index construction, URLs don't play any significant role, so these must be converted to text by extracting anchor text. As an example, the anchor text for the above URL will be electrical engineering. The other parts of the URL must be discarded.

**Part 1: Corpus Analysis**

Q1. Unigram analysis:

 (a) Mention the total unique unigrams present in the corpus.

 (b) Plot the distribution of the unigram frequencies.

 (c) How many (most frequent) uni-grams are required to cover the 90% of the complete corpus.

Q2. Bigram analysis:

 (a) Mention the total unique bigrams present in the corpus.

 (b) Plot the distribution of the bigram frequencies.

 (c) How many (most frequent) bi-grams are required to cover the 90% of the complete corpus.

Q3. Trigram analysis:

 (a) Mention the total unique trigrams present in the corpus.

 (b) Plot the distribution of the trigram frequencies.

 (c) How many (most frequent) tri-grams are required to cover the 90% of the complete corpus.

Q4. Repeat Q1, Q2, and Q3 after performing the stemming process on the tokens.

Q5 Briefly summarize and discuss the frequency distributions obtained in Q1 to Q4. Do these distributions approximately follow Zipf's law?

Q6. What library you used for tokenization and stemming? What were the underlying algorithms used by the library for these tasks?

Q7. Report three examples based on your observation, where the tool used for tokenization did not tokenize the character sequence properly.

**Part2: Vector-space based IR System**

In this part, the objective is to build a vector-space based information retrieval system. Implementing a working system will help you in better understanding of how actual IR systems work and what are some practical issues faced during the building of an IR system.

For this assignment, you can use an in-memory based index construction algorithm. A simple way would be to use a python dictionary data structure, with keys as a term and a list as doc id. However, you are free to use any other data structure. The primary focus of the assignment is on the ranking of documents wrt queries.

Follow the following guidelines in building the IR system:

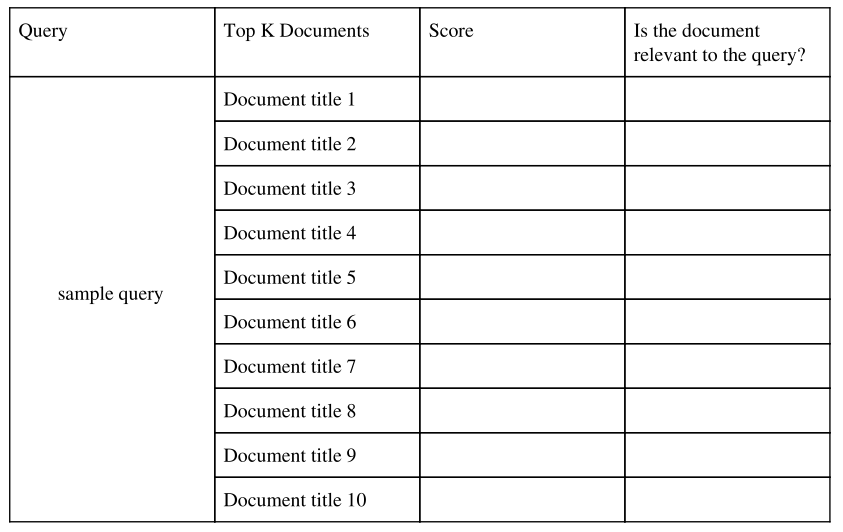
1. The vector space model should be used for computing the score between document and query.

2. Use the lnc.ltc scoring scheme (based on SMART notation).

3. The queries should be free-text queries.

4. Do not remove stop words. Do not perform stemming/lemmatization and normalization. Punctuations can be removed.

After the system is built, you need to evaluate the system on at least five multi-term queries. The queries should be selected such that it covers several cases, for example, common nouns, proper nouns, rare terms, ambiguous terms, etc. For each query, you have to evaluate the top 10 documents retrieved, and mark them manually, whether they are relevant to the query or not as per the following format:



Briefly summarize the results you have obtained.

**Implementation guidelines for part2:**

There should be two components in the code:

1. The index creation code: One or more code files that will be used to create the inverted index. This code should store the inverted index (and/or any data you want to extract from corpus) on one or more files on the disk (in any format).
2. A single code file that will take a query and paths for the stored inverted index (and/or other stored data) as an input and should output the top K documents. This file should never read the text corpus.

**Deliverables**:

1. Well commented code​. The purpose and intent of each method, class, function, and module should be mentioned appropriately.
2. Report: A report describing all important assumptions made for implementing, limitations, algorithms used, etc.
3. Readme file​ having all the steps for running your code. Also, include the name for the Wikipedia file used by you.

**List of tools/libraries**

• Python NLTK (http://www.nltk.org/)

• CoreNLP (http://nlp.stanford.edu/software/corenlp.shtml)

• Apache OpenNLP (http://opennlp.apache.org/)

• spaCy (https://spacy.io/)

You are can also use Google Colab (<https://colab.research.google.com/>). In Colab, you can write and execute Python code online on Google's servers.

The assignment can be easily managed on a decent machine (i3, 4GB RAM). If you are facing some issues, please post it on the eLearn portal; there might be something wrong with the approach.

Note on plotting n-gram frequencies: You can plot a Frequency vs Rank line plot on log-log scale, as the Zipf's law is plotted. If you try to plot a histogram, it may freeze your machine for some time.

Submit a single file (.zip) containing all deliverables. The report should be in PDF format. The size limit for the .zip file is 10 MB. Thus, do not submit any data, only the code, readme, and report. The report should include the images of the plots asked in part 1.

**Additional Info**

The skill required to complete the assignment, irrespective of a specific programming language :

1. Reading files

2. Extracting text from HTML tags.

3. Tokenize text and convert it to n-grams.

5. Stemming.

6. Plot graphs

7. Using a dictionary to build an index

8. Convert the document and queries to vector

9. Compute similarity between documents and queries.

It is advised not to use C or C++ programming language, as the code can become very complicated and there might not be libraries available for stemming.

A good and free programming resource to learn python is available at <https://learnpythonthehardway.org/python3/>