CSE474: Information Retrieval and Extraction
Mini-Project (Phase 2): Search Engine for Wikipedia

List of allowed external libraries for Python

PyStemmer
NLTK (PorterStemmer, SnowballStemmer, WordNetLemmatizer)

Desirable Features:

- Support for Field Queries - Fields include Title, Infobox, Body, Category, Links, and References of a Wikipedia page.
- Index size should be less than one-fourth of the dump size (you can experiment with different index compression techniques if you want).
- Search results should be displayed within 0 - 5 seconds depending upon query type/length
- Check out the relevant sections from Chapters 4, 5.2, 5.3, 6, 7, 11.4.3 in the 'Intro to IR' book.

One important thing to note is the trade-off between index size and search time - a highly compressed index might increase the search time.

Evaluation Criteria:

Evaluation for phase 2 will be on following two criteria:

- Inverted index size
- Search time and relevance
- Viva on your implementation

Submission Format:

You have to submit a zipped folder named as your roll_number having 4 files at least

1. Code for index creation
2. Code for search
3. stats.txt - A text file containing these stats on separate lines:
    - index size in GB (for e.g. 10 GB)
    - number of files in which inverted_index is split (for e.g. 26)
    - number of tokens in inverted_index (for e.g. 1000000)
4. readme.txt - A text file mentioning any convention, for e.g. if your index creation code is split into 3 files, please mention those files here.

    This is to make sure that your code for index creation and search doesn't change substantially by the date of evaluation.

Query Search:

Given a query string you would have to return the page_id, page_title of top K results. Unlike phase 1, this time, you will run the code on your system as inverted_index is there. The only difference is, we will provide these search queries via a queries.txt file during your evaluation and you would have to write the output in a queries_op.txt file.

`$python search.py queries.txt`

Input:
`queries.txt` file, it contains each query on a separate line. Each line is of the format `K, QS` where K is the number of results to return and QS is the query string. `1<=K<=100`

Sample `queries.txt`

```
3, t:World Cup i:2019 c:Cricket
2, t:the two towers i:1954
```

Output:
`queries_op.txt` file. It should contain K `page_id, page_title` on each line for a given query. After K lines, another line will contain two times in seconds - Total time for K queries, Average time per query. Results for different queries will be separated by a new line.

`page_id` - This is the original page id of doc as per the .xml file. In case, you have used your own ID scheme, you can print that instead.

`page_title`: This is the original page title of the doc as per the .xml file. Apart from lowercasing, no other preprocessing should be done on this.

Sample `queries_op.txt`:

```
7239, cricket world cup
4971952, 2019 cricket world cup
57253320, 2019 cricket world cup final
10, 3.33

63750, the two towers
173944, lord of the rings: the two towers
8, 4
```