CS4.406: Information Retrieval & Extraction
Mini-Project - Search Engine for Wikipedia
Phase 2 Requirements

1 Task for Phase 2

In phase 1, you have worked with a small XML dump ($\sim$ 1.6 GB) of Wikipedia by parsing, processing the text and building an inverted index over it. In phase 2, you'll be extending this to a complete English Wikipedia dump of 90 GB to build an inverted indexing mechanism over this large corpus. The index could be divided this time into multiple small files instead of one single index file. You also wrote a query parsing code to retrieve the entire posting lists corresponding to the keywords. Now, you have to complete the information retrieval task by adding a querying mechanism to work with your index. This system is expected to, in response to the given plain or field queries, output relevant Wikipedia page titles for the queries. Relevancy of the returned search output will need a ranking mechanism added to work with the query processing stage.

2 Expected Features

- **Support for plain & field queries**
  As in phase 1, you have to ensure support for both - *plain and field queries*. The expected fields from a Wikipedia page to be supported along with their respective identifiers used in query are as shown below in Table 1.

Field	Title	Body	Infobox	Category	External Links	Reference
Symbol	t	b	i	c	l	r

Table 1: Fields considered with their corresponding symbols

- **Index size**
  As specified in phase 1, the index size without any compression should at most be *one-fourth* of the total uncompressed size of the dump.

- **Search time & relevancy**
  The search results are expected to be output in up to *6 seconds*. Also, you'll be judged on the *relevancy* of the results w.r.t. query.

3 Query Search

For each query string, you have to return the **top 10** results, each result consisting of the *document id* and *title* of the page. Unlike phase 1, however, the inverted index will not be created or stored on our system. So you have to run the querying code using the inverted index on your own individual systems. The queries will be provided to you at the time of evaluation (viva) in a `queries.txt` file. It can be run as, for instance:

```
python search.py queries.txt
```

Your code should write the search output for the given queries in a `queries_op.txt` file which we will use for evaluation.

1. `queries.txt`
   This file given to you will contain each query string printed on a single line.
   A dummy `queries.txt`:

   ```
   t:World Cup i:2019 c:cricket
   the two Towers
   ```

2. `queries_op.txt`
   In this file, for each query, you should have the results printed on `10` lines, each line containing the *document id* (which is a number specific to your implementation) and *title* of the corresponding Wikipedia page. The title should be the original page title as contained in the XML dump. Apart from lowercasing, *no other processing* should be done on it.

   At the end of the `10` lines of search results, for each query, you need to print the total time taken in seconds for the `10` results of that query.

   The results for each query in the file should be separated by a blank line in between.

   Dummy results for the above queries:

   ```
   7239, cricket world cup
   4971952, 2019 cricket world cup
   57253320, 2019 cricket world cup final
   .
   .
   .
   (10 lines of retrieved document id - title pairs)
   5

   63750, the two towers
   173944, lord of the rings: the two towers
   .
   .
   .
   (10 lines of retrieved document id - title pairs)
   2
   ```

4 Evaluation

You will be graded based on the following criteria for phase 2:

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

5 Instructions

- Allowed Programming Languages: **Python3, C++ and Java**.
- Any sort of plagiarism or academic dishonesty will lead to **0** in the mini project.
- Please ensure that the restrictions on using external libraries in phase 1 are adhered to in this phase as well. *No other libraries other than NLTK and PyStemmer are permitted for text processing, as specified in the phase 1 requirements document.*

- Since the processing times on different systems would vary, difficulties in having the query result output in stipulated time would be dealt with on a case-to-case basis at the time of viva.
- **Submission Format:**
  You have to submit a zip file named `roll_number.zip`, e.g. `2020701007.zip`. It should contain the following files:

  1. Code for index creation
  2. Code for search
  3. `stats.txt`
     This text file should contain the following three stats on separate lines:
     - index size in GB (for e.g. 17.36)
     - number of files in which the inverted index is split (for e.g. 26)
     - number of tokens in the inverted index (for e.g. 872985644)
  4. `readme.txt`
     It can contain a short description of how your code is divided into files, and their primary functionalities.

  This submission will help us check that your code *doesn't change substantially* by the date of viva evaluation.

6 Resources

You can refer to following excerpts from the Introduction to Information Retrieval book by Stanford NLP for your conceptual understanding.

1. **Index construction:**
   (a) Hardware basics
   (b) Blocked sort-based indexing
   (c) Single-pass in-memory indexing
2. **Index compression:**
   (a) Motivation
   (b) Statistical properties of terms in IR
3. **Scoring & term weighting:**
   (a) Parametric & zone indexes
   (b) Term frequency & weighting
   (c) Variants of tf-idf