CS4.406: Information Retrieval & Extraction
Mini-Project - Search Engine for Wikipedia
Phase 1 Requirements

Deadline : 16th August 15:00 (3PM)

1 Project Objective

In this mini-project, your task is to build a scalable and efficient search engine on Wikipedia pages. This constitutes two stages - inverted index creation and query search mechanism, where the scope of performance in the second stage relies heavily on the quality of index built in its preceding stage. Throughout the project, efforts should be made to build a system optimized for search time, search efficiency (i.e. the quality of results), indexing time and index size. We will provide Wikipedia dumps in XML format, you are required to parse it to get Wikipedia pages.

2 Basic Stages

- XML Parsing
  Prefer SAX parser over DOM parser. If you use a DOM parser, you can't scale it up for the full Wikipedia dump later on.
- Tokenization
- Case folding
- Stop words removal
- Stemming / Lemmatization
- Posting list / inverted index creation
- Optimization

Please note that external libraries that would require installation apart from the following for stemming/lemmatization or other purposes aren't allowed:

- *NLTK (PorterStemmer, SnowballStemmer, WordNetLemmatizer)*
- *PyStemmer*

3 Expected Features

- Support for field queries – Fields include **Title**, **Infobox**, **Body**, **Category**, **Links** and **References** of a Wikipedia page.
  - Plain query examples : *Lionel Messi*, *Barcelona*

  - Field query examples : *t:World Cup i:2018 c:Football* – search for "World Cup" in **Title**, "2018" in **Infobox** and "Football" in **Category**
  - Given a query, you have to return the **matched posting list(s)** from your index - this should be human readable.

- Index size should be at most **one-fourth** of the original dump size (try different index compression techniques to get the best results).
- Index should be created in around **180 seconds for C++/Java** and **400 seconds for Python**.
- For this phase, the data you are given is relatively small. You can experiment with different indexing methods easily. Prioritize optimizing your indexing strategy in this phase, because the data for phase 2 will be significantly larger ($> 50GB$) and creating a new index after every optimization will not be feasible.

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

4 Evaluation

- For this phase, you'll be evaluated based on the following:
  - Indexing time
  - Index size
  - Support for field queries
- *We **won't** be evaluating the quality of your search results for this phase.*

5 Instructions

- Allowed Programming Languages: **Python3, C++ and Java**.
- Any sort of plagiarism or academic dishonesty will lead to **0** in the mini project.
- There will be **no** extension in the deadline.
- The runtime will be different on different systems. So, to resolve this, we'll be running codes for this phase on our server.
- Submit a zip file named roll_number.zip, Eg, 2018101011.zip. The Directory Structure of the zip file is follwoing:
  - Directory name : roll_number. Eg, 2018101011
  - It should have files named **index.sh** and **search.sh**.
- We'll run your code like this:

```
$ bash index.sh <path_to_wiki_dump> <path_to_inverted_index> invertedindex_stat.txt
$ bash search.sh <path_to_inverted_index> <query_string>
```

- We'll be passing the path to the inverted index folder to **index.sh** and **search.sh**. So make your script according to that.
- **invertedindex_stat.txt** should contain two numbers on separate lines
  - Total number of tokens (after converting to lowercase) encountered in the dump.

  - Total number of tokens in the inverted index.

- A sample of what **index.sh** and **search.sh** might contain if you are using Python

```
python indexer.py $1 $2 $3
python search.py $1 $2
```

6 Resources

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

1. Text preprocessing
2. Boolean retrieval
3. Processing boolean queries
4. Basic information retrieval
5. Illustration of an inverted index