Information Retrieval
Sessional - 4 (Additional)

A. Mustafi

September 8, 2023

0.1 Building a more full fledged invereted index

While so far we have created our inverted indexes in Python, they have been slightly naive. In this challenge exercise we want oto build a more comprehensive inverted index. The design goals for the project are as follows:

1. We shall have a tokenize function which can tokenize our texts as per our wish. The function should be able to contorl the minimum length of the tokens, case transformations, whether stopword removal is required, whether stemming needs to be done etc.

2. We shall store the text files in a separate indexed data structure for quick retrieval. The file to be stored is the raw file.

3. Our vocabulary should contain unigrams, bigrams and phrase templates.

4. The inverted index itself shall not containg tuples of the form *(token, doc_id)*, but rather of the form *(token_id, doc_id)*.

5. We shall maintain a separate vocabulary which shall contain additional metadata about each term e.g. length of the string, number of bigram subsets, number of documents containing the term etc.

6. Each posting shall not only contain information about the document where a term was found i.e. the doc_id, but aso about the positions in the document where the term was found. These postions are usually captured as a list of offsets.

7. We should also facilitate the looking up of a token given its token_id.

8. We should be able to write all the involved data structures to secondary storage to be loaded back when required. We can use the *pickle* or *cpickle* packages for this.

0.2 Taking it even further

If we are successful in building the inverted index in the previous exercise we can take it even further. The new project specifications would require us to tokenize an expanding corpus on the fly i.e. in real time. The project specifications shall be:

1. We shall design a parser to download "N" pages starting with a seed URL.

2. The downloaded pages shall be saved in their raw form in a folder on your machine.

3. Your parser shall pick a single file from the folder to add it to the inverted index.

4. The file shall subsequently be removed from the downloads folder. *(#Hint you can use the shutil package for this.)*

5. The index creation shall continue in parallel with the spider, untill all downloaded pages have been indexed.

6. Remember there may be a synchronization issue as the indexer could end up asking for a file when none exists in the downloads folder ready to be indexed e.g all files have been indexed and the next file is yet to be downloaded.