2024-05-04 Information Retrieval - Boolean Retrieval

Submitter	Yusuf Syaifudin
Student ID Number	23/525770/PPA/06617
Date of submit	20 May 2024

Code

Source code: https://github.com/yusufsyaifudin/20240504-boolean-retrieval

System Architecture

- For Indexing task, I separate into two sub-task: data cleaning (case-folding, stopwords using `nltk.corpus` Indonesian corpus, stemming using `Sastrawi==1.0.1`) and then create the inverted index.
- For Retrieval task, I create function to the index model and try two query for each boolean logical operator: `AND`, `OR`, and `NOT`.

Analysis

Computation time

To build the index from 14343 documents, it took 1 hour 35 minutes 58 seconds on my Apple Macbook M1 Pro CPU 8-Core, GPU 14-Core, RAM 16 GB. You can see this process in indexed.ipynb.

When retrive data with query `["pemerintah", "korupsi"]`, the index `word_map_stemmed_all_word.json` took longest time compared to other index. This is because it need to do stemming before searching. It still take longer than `word_map_stemmed_not_stopword.json` because the index file is larger. Also, why `word_map_not_stemmed_all_word.json` take longer time even if I am not doing stemming, I guess that is because the index file is the largest than the others.

For query `["jalan", "rusak"]`, the index `word_map_not_stemmed_all_word.json` is took the longest time. Same as previous query, I guess that it is because the index file is large enough to do search. My assumption is supported by the fact that in the query `["pemerintah", "korupsi"]` is also takes longer time.

Also, in general we can see that `word_map_stemmed_all_word.json` is always longer than `word_map_stemmed_not_stopword.json` because it has combo: large index and need stemming.

The time to query and it's detail is described in table below:

Logical Operator	Index file	query: `["pemerintah", "korupsi"]` (in seconds)	query: `["jalan", "rusak"]` (in seconds)
AND	word_map_not_stemmed_all_word.json	1.037159	**1.039380**
AND	word_map_stemmed_not_stopword.json	0.776101	0.702490
AND	word_map_stemmed_all_word.json	**1.101779**	0.978623
OR	word_map_not_stemmed_all_word.json	1.086989	**1.036240**
OR	word_map_stemmed_not_stopword.json	0.759185	0.711646
OR	word_map_stemmed_all_word.json	**1.113027**	0.984228
NOT	word_map_not_stemmed_all_word.json	2.037013	**2.032055**
NOT	word_map_stemmed_not_stopword.json	1.340236	1.380375
NOT	word_map_stemmed_all_word.json	**2.057885**	1.923374

Are the retrieved documents relevant to the queries?

Since the query used in this trial is pretty simple, I can conclude:

- Using logical operator `AND`, both queries (`["pemerintah", "korupsi"]` and `["jalan", "rusak"]`) return good relevance of documents (using all index file). This is because the stemming index is consistently applied during indexing and retrieval process. Also, since `AND` query means that all `token` or `word` must exist in the same document, it is easily for inverted index to match the desired documents.

- `OR` queries return somewhat less relevant document. My guess is because it simply push any document with minimum one matches of `token` or `word` to the results array. Then it only sort the documents based on the `word` occurences.

- `NOT` query is clear and predictable to have most less relevant result. Actually we cannot say it is irrelevant, because the nature of the query is "we want a document that NOT contain all of the words in the array". So, since the returned document already return the documents that not contain any single word or token from the query list, then we can say that it is work well and can return results as expected.