Code documentation

Three files: naive_indexer (project I), query processing (project II), dictionary_compression (project II)

Run query_processing and dictionary_compression to get outputs in the console.

Project III: To give dictionary compression statistics as given in table 5.1, I worked directly with term-docId pairs lists.

To do so I had to make some equivalences:

- Distinct terms = to making a set of all terms
- Non-positional postings = making a set of all terms, but only for each document. We keep duplicates across documents
- Tokens = keeping duplicates term entries within a document and across all documents

Compressed dictionary is generated at the end from the compressed list.

All projects are run on the first 5 files for decent performance.

Query analysis

One word query on the index is trivial as it only results in calling a dict key. Associated posting list is returned from preset pointer. Wildcard queries, OR, AND or performance optimizations would have added complexity.

Queries on the compressed index have to be compressed with the same algorithm as the dictionary compression algorithm in order for the key to get a hit. Doing so on the index gives similar results. “Similar” because the compressed index has an entry in its posting list for each term instance in the document (equivalent to positional postings) -> more results but same if put into set.

Compression analysis

1. Distinct terms:
    a. Removing numbers: Good effect. Many numbers in the collection, it is why variation is better than in table 5.1.
    b. Case folding: Good effect. Many uppercase tokens in the collection, it is why variation is slightly better than in table 5.1.
    c. Removing Stop words: Minimal effect as expected. Stop words to regular word ratio is low. Similar to table 5.1.
    d. Stemming: Did not reduce term count. This is a different result from the table. Maybe a bigger input would have increased the variation.
2. Non positional posting:
    a. Removing numbers: Good effect. Many numbers in the collection, it is why compression is better than in table 5.1.
    b. Case folding: Good effect. Better result than in table 5.1. Worse result than for distinct terms, we can assume that counting the appearance of a word in a

document did not have an even repartition through the dictionary. Words with longer posting lists were less affected.
    c. Removing Stop words: Same analysis as b)
    d. Stemming: Same analysis as #1. Bit smaller corpus so a slightly better compression.
3. Positional posting:
    a. Removing numbers: Similar compression and explanation as #1 #2
    b. Case folding: No effect as a mapping from m -> n where n is smaller because of lowercasing will only result in merging some posting lists.
    c. Removing Stop words: Major effect as in table 5.1 because stop words have large posting lists.
    d. Stemming: No effect. Similar to b). Expected.

Design

- Separation of concerns into three modules and individual functions for each operation and compression.
- Pipeline style compression in order to be able to output compression results Dictionary index, as requirement is single word queries
- Functional filtering and mapping in memory of term-docId lists as performance is not a requirement

What I learned

All compression operations that result in merging posting lists under the same key like lowercasing or stemming do not reduce the size of the corpus for positional posting. Removing stop words does not do much for non-positional postings, but the opposite is true for positional postings.

Performance is quickly an issue for even small corpus as keeping this amount of data in memory is costly. Stemming operations are also very costly. I would suggest a streamed like pipeline with parallelized compression steps and asynchronous merging to the main index.

Compression results on the next page.

```
 -- Positional postings --
Unfiltered: 1072751
No numbers: 908016
Variation -16%

Case folding: 908016
Variation -0%

Without stop words: 661638
Variation -28%

Stemming: 661638
Variation -0%
```

```
 -- Non positional postings --
Unfiltered: 86369
No numbers: 70354
Variation -19%

Case folding: 54448
Variation -23%

Without stop words: 53776
Variation -2%

Stemming: 40150
Variation -26%
```

```
 -- Distinct terms --
Unfiltered: 45321
No numbers: 34322
Variation -25%

Case folding: 25777
Variation -25%

Without stop words: 25635
Variation -1%

Stemming: 25635
Variation -0%
```