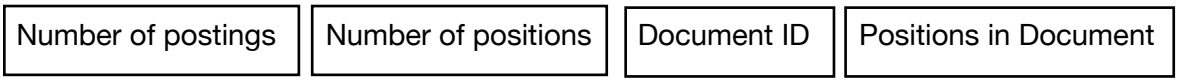
Design

The postings consists of a document ID which is simply sceneID+1 and a list of document position the term appears at. The inverted index for each term is then written into a binary file that follows the format as the following:



For the uncompressed binary file, every number is written in as a 4 bytes integer. For the compressed file, only the document IDs and positions are delta are delta encoded and then variable byte compressed.

A lookup table is constructed as well. For each term, it stores the following information: file offset for uncompressed binary file, file offset for compressed binary file, term frequency and document frequency. The lookup table is serialized into a Json file named “lookup.json”.

There are two runnable classes, one is Indexer and the other one is QueryEngine. Indexer can be run with no argument, it does the following tasks, builds the inverted indexes and write them into binary files, builds the lookup table and serialize it to a Json file.

QueryEngine as for now takes three positional arguments. The first one is a boolean value that if given as true, uses compressed binary file to perform query and if given as false, it uses uncompressed binary file. The second argument is the name of the query file that contains the terms to query. The third argument is a number that shows the top K query result.

Why might counts be misleading?

Simply using counts can be misleading because a case like one single term in the query that shows up too many times in a document can skew the result. For example, a query term “tropical fish”, if using counts alone, it could return documents that have a lot of occurrences for tropical but zero occurrence for fish and vice versa. We could improve this by counting co-occurrence instead.

Scene Statistic

Average length of a scene: 1199.5562 words
shortest scene scene ID: 549
longest play ID: hamlet
shortest play ID: comedy_of_errors

Compression Hypothesis

Table 1 is computed by randomly select 7 terms and calculate their term frequencies and document frequencies.

Compressed binary	Uncompressed binary
199ms	766ms

Table 1

Table 2 is computed by running the 100 sets of queries 100 times.

	7 terms queries	14 terms queries
Compressed binary	11866ms	585197ms
Uncompressed binary	33339ms	2020611ms

Table 2

All experiments were performed twice to eliminate the effect of any caching might had. The experiment result shows that compression hypothesis holds here. The result might change if the gaps between document positions are large. In that case, using variable byte encoding might actually result in larger file size, couple with the extra time it takes to decode the data, compressed binary in this case will take longer than uncompressed binary.