COL 764 | ASSIGNMENT 1
INVERTED INDEX CONSTRUCTION

DIPEN KUMAR
2018CS50098

METRICS OF INTEREST

Index Size Ratio (ISR)

1. C0 — 0.413
2. C1 — 0.097
3. C2 — 0.101
4. C3 — 0.149
5. C4 — 0.088
6. C5 — 0.076

COMPRESSION SPEED

1. C1 — 0734533 milli-seconds
2. C2 — 1352357 milli-seconds
3. C3 — 0634545 milli-seconds
4. C4 — 0643523 milli-seconds
5. C5 — 0592344 milli-seconds

QUERY SPEED

1. C0 — 084573 µ-seconds
2. C1 — 055477 µ-seconds
3. C2 — 064557 µ-seconds
4. C3 — 062356 µ-seconds
5. C4 — 050235 µ-seconds
6. C5 — 043235 µ-seconds

IMPLEMENTATION DETAILS

Since it was given that vocabulary can be stored in main memory. I have created a python dictionary storing all words as keys with value file handler storing its posting list. Now since there vocabulary size is way too big as compared with the maximum number of file operating system allows to open, I used this maximum number to open file and whenever I reach limit I close all files opened till now.

=> A file is dedicated for each word to store it's posting list.

I iterated word by word over all documents once and added the current document number in posting list of current word avoiding duplication of document in posting list via creating a set for words in a particular documents. Alongside I created a map from document number to document identifier assigning number 1 to first document, 2 to second and so on.

I saved this map in indexfile.dict with compression id used in indexfile.idx along with vocabulary and offset in the indexfile.idx from where it's posting list starts.

I created inverted index that in my case is a directory named invidx which contains one file for each word storing posting list. I then dumped posting list in indexfile.idx and word and offset in indexfile.dict and keep deleting my temporary created file for storing posting list of that word which was recently dumped in indexfile. I filanlly delete invidx directory.

I have implement all C1, C2, C3, C4, C5 compressions methods given in assignments.

I have decompressed posting list in boolsearch.py file in the way specified in assignment. For each query I load posting list of word one by one and keep storing intersection and discarding posting list after taking intersection before loading next posting list. This way I ensure I don't run out of main memory.

ADDITIONAL MANUAL TUNING

Opening and closing a file was costly and hence I stored file handlers in dictionary and didn't close it until i have done indexing entire document. But operating system have limit on number of files that can be opened at a time and hence I utilized this maximum number by try catch statement. If i run out of space I clean all opened files and start storing recent onces. This was my strategy for handling misses to exploit temporal locality.