Ahmed Alhaj
Phase3: inverted index

An inverted index a data structure used for storing it plays an essential role in the search engines indexing system. The inverted index is sort of nice solution of an unstructured data. It is also used in mapping from words to posting lists. it is non-trivial to get the number of unique words in a single document. We have to traverse the whole index for this purpose, and this is clearly very expensive.

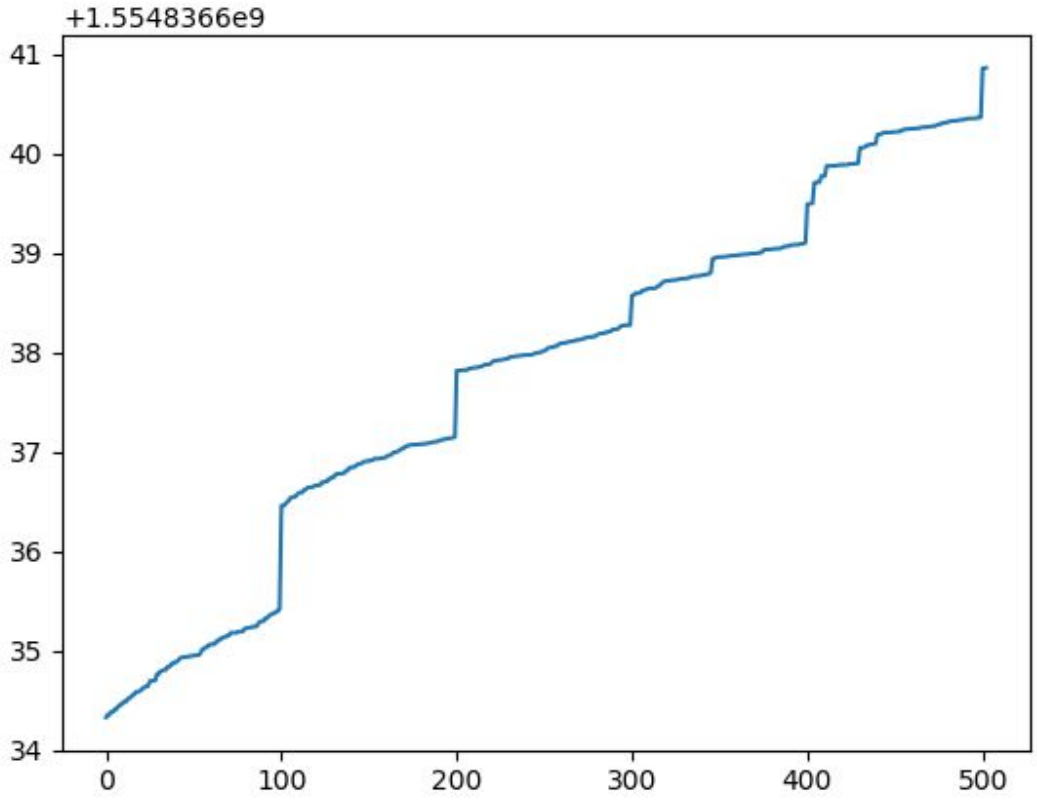
The processor already was discussed in phase 2 report. The preprocessor has three parts, a files pipeline, document processor, and tokenizer. The pipeline open the set of the file filter the html syntax and use re.compile to read only characters and discard whitespace and symbols and the pass the file content a document processor. The document processor has parser to tokenize the file from the common words. I already built the inverted index for homework 2. As of this assignment I just fix some minor details. during the studying for midterm I read about the Blocked sorted based indexing for the inverted index. One of thing that I stumped at is how would identify and add common phrase as separate term to index.

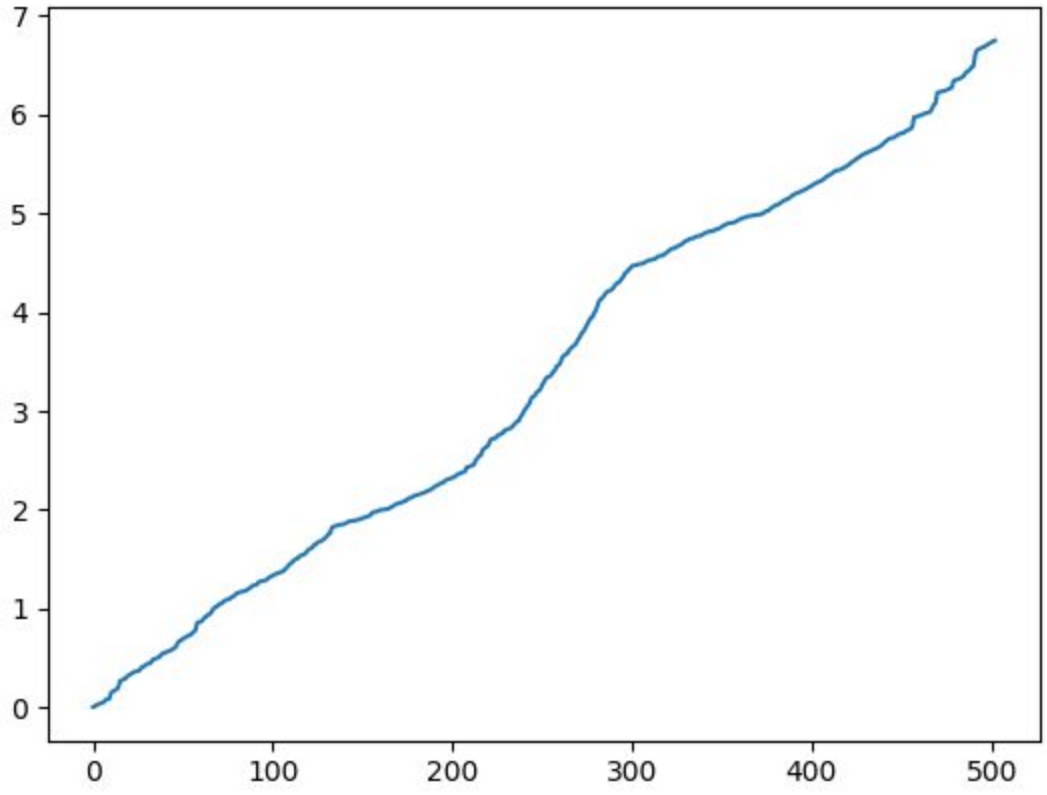
The way I am building the inverted index with the term weight and I am collection nested collection dictionary with the key as term ID and values would be a nested dictionary with key as Document ID and the value as frequency of term in that dictionary. For the the construction algorithm I used block sorted based indexing. The way I did it, instead of splitting the html files into blocks, I just had a counter and if statement that said if you reached a 100 html file start new inverted index and append it to list of inverted indexes. Because of our data set is relatively small so I did ignore that part in which I saved to the disk. The I am calculating the tf-idf, after finish collecting each block “100 html” I I am calculating the the term weight. After everything is finish I am merging all the blocks together into 1 final inverted index.

The algorithem goes as following

BSBI():
while (all documents not processed)
do block
BSBI-Invert(block)
WriteBlockToDisk(block, fn)
MergeBlocks(f1, f2..., fn, fmerged)

Since we have only one block of document, so I just used an if-statement to separate each 100 as one block.


+1.5548366e9
41
40
39
38
37
36
35
34
0
100
200
300
400
500


7
6
5
4
3
2
1
0
0
100
200
300
400
500