Homework 2 Report

INDEX SIZE

Compressed | Stemmed	84.1 MB
Decompressed | Stemmed	186.4 MB
Compressed | Unstemmed	85.4 MB
Decompressed | Unstemmed	193.2 MB

Brief Explanation on the process used for Indexing

1. Processed the documents in the batch of 1000 documents in multiple threads
2. For each batch of 1000 documents, I created a partial inverted index and a partial catalog file which stored the metadata of partial inverted indexes and stored them on disk. This resulted in 85 inverted index files and 85 catalog files per index.

   Format of Partial Inverted List : docId : [positions]

   Format of Partial Catalog File : term : {[path_to_inverted_index, file_name_of_inverted_index, offset, size]}
   If you look at the partial catalog file, it basically tells me the exact location of partial inverted lists for the term!
3. Now since the term might be in different partial indexes, I first merged all the catalog files one by one in memory to form a single merged catalog file in memory which will be used to form the complete inverted list
4. Next, I traversed term by term in the catalog and used the array of metadata mentioned above to retrieve all the partial inverted indexes for the term and merged them to store them in a new file which will basically be the inverted index for the corpus for each term. Also created a new catalog for the term containing the metadata of the complete inverted index for the term
5. That's it. Now we have a catalog file which contains the metadata for inverted indexes for each of the terms.

MODEL PERFORMANCE

Index	Model	Old Score	New Score	Percent (New/Old)
Decompressed | Stemmed	TF-IDF	0.2959	0.2886	97.53%
Decompressed | Stemmed	Okapi TF	0.2510	0.2308	91.95%
Decompressed | Stemmed	Okapi BM-25	0.3020	0.2925	96.85
Decompressed | Stemmed	Unigram LM with Jelinek Mercer	0.3007	0.2878	95.71%
Decompressed | Stemmed	Unigram LM with Laplace	0.2420	0.2376	98.18%
Decompressed | Unstemmed	TF-IDF	N/A	0.2356	N/A
Decompressed | Unstemmed	Okapi TF	N/A	0.1921	N/A
Decompressed | Unstemmed	Okapi BM-25	N/A	0.2432	N/A
Decompressed | Unstemmed	Unigram LM with Jelinek Mercer	N/A	0.2303	N/A
Decompressed | Unstemmed	Unigram LM with Laplace	N/A	0.2132	N/A
Compressed | Stemmed	Okapi BM-25	N/A	0.2925	N/A

Inference on above results *(Make sure to address below points in your inference)*

- *1. Explain how index was created*
- *2. Pseudo algorithm for how merging was done*
- *3. Explain how merging was done without processing everything into the emory (Important)*
- *4. How did you do Index Compression (For CS6200)*
- *5. Brief explanation on the Results obtained*
- *6. How did you obtain terms from inverted index*

1. **Indexing :** Processed documents in a batch of 1000 documents and created a partial inverted list for each term appearing in those documents and stored it in file. Maintained a catalog to store the information about the partial inverted index. This resulted in 85 inverted index files containing partial inverted indexes for terms and 85 catalog files which contained information about the location of the partial inverted list.
   Format of Partial Inverted Index File : {docId:[pos1,pos2], docId2:[pos3,pos4,...]}{...}

Format of Partial Catalog File : {term :{path : '/output', filename : 'inverted_index36', start : 34, size : 1000} {...}

2. Pseudo Algorithm for Merging :

Merging Catalog : Pseudo Code

```
main_catalog = {}
for file in all_catalog_files :
        main_catalog = merge(main_catalog, file)
```

main_catalog will look like {'term' : [{path, filename, start, size},{path, filename, start, size}]}

Merging Inverted Index : Pseudo Code

```
for term in main_catalog :
        inverted_list_by_term = {}
        for info in main_catalog:
                partial_index = seek(path, filename, start, offset)
                inverted_list_by_term = merge(inverted_list_by_term, partial_index)
        start = getCurrentOffsetToEnd(path, 'main_inverted_index')
        writeToFile(path, 'main_inverted_index', inverted_list_by_term)
        end = getCurrentOffsetToEnd(path, 'main_inverted_index')
        catalog_by_term = {term : {path: path, filename: 'main_inverted_index', start : start, size :
        end-size}}
        writeToFile(path, 'main_catalog_file', catalog_by_term}
```

3. **i) Merging Catalog :** Once the partial indexes and catalog were formed, I merged the catalog file one by one to form the merged catalog which contained information about the location of each partial inverted index for the term.
Example :

Catalog 1 → {'president' : {path : '/output', filename : 'inverted_index6', start : 20, size : 1000}} {...}
Catalog 2 → {'president' : {path : '/output', filename : 'inverted_index28', start : 0, size : 345}} {...}

The merging of the two catalog would result in
{'president' : [{'president' : {path : '/output', filename : 'inverted_index6', start : 20, size : 1000},{path : '/output', filename : 'inverted_index28', start : 0, size : 345}]}

This result would in turn would be merged with Catalog 3 and so and so forth to form the merged catalog file which contains the list of information details about each term's partial inverted index

ii) Merging Inverted Index : Now that we have a merged catalog file, we will proceed to merge the partial inverted indexes to form one. To do so, I traversed term by term in the merged catalog file and used the list of information metadata to fetch the partial inverted list for each term and merged them

into one. This would result in the complete inverted index for a specific term. I would then store this into a new file and create a new catalog on disk containing the information about the complete inverted index for the term. We would do this for every term and since we are doing this term by term, we never keep a large chunk of inverted indexes in memory.

Example :
Let's say we are merging partial indexes for president
{'president' : [{'president' : {path : '/output', filename : 'inverted_index6', start : 20, size : 1000},{path : '/output', filename : 'inverted_index28', start : 0, size : 345}]}

We would use seek() to get data from the particular filename along with the path which would provide us partial inverted index for president from both the files

inverted_index6 → [4:[34,5,6],29:[673,839]]

inverted_index28 → [88:[13,45]]

The merged inverted index for the term would like [4:[34,5,6],29:[673,839],88:[13,45]] which will be stored in the main inverted index file and the offset information along with size will be stored in the final catalog file for easy access.

4. **Index Compression :** Next to create a compressed index, I made use of the zlib library in Python to compress my data i.e. convert it into bytes and store it in a binary file. For each term in the catalog, I compressed the inverted index for each term i.e. converted the string into bytes using zlib compress functionality and stored it in the main inverted index file and created a catalog file which provided the offset and size for the bytes in the main inverted index file. This way while decompressing I did not decompress all the data but instead did a seek to read the bytes for a given term and decompressed only the data for a specific term.

5. **Results Obtained :** Ran the queries on each of the following models that were a part of Homework 1 : OKAPI TF, TFIDF, OKAPI BM25, LM WITH LAPLACE SMOOTHING, LM WITH JK SMOOTHING.
I was able to receive more than 90% precision for stemmed index using my own index as compared to Elasticsearch's index

6. I did not store the term in the inverted index, instead stored the inverted index for the term and stored the term along with the offset information in the catalog file. To read the inverted index for a specific term, I would look into the merged catalog which was maintained in memory to get the offset information and the size for the list. I would then use seek() operation to look for the inverted index in the file at the particular offset with the size.

   Format of Partial Inverted Index File : {docId:[pos1,pos2], docId2:[pos3,pos4,...]}

   Format of Partial Catalog File : {term :{path : '/output', filename : 'inverted_index36', start : 34, size : 1000}

   To get the inverted index for the term, I would open the file using the path and filename information stored in the catalog file for each term and then simply do seek using the offset and size info.

PROXIMITY SEARCH (*For CS6200*)

Index	Score
Unstemmed	0.2441
Stemmed	0.2940

Inference on the proximity search results *(Make sure to address below points in your inference)*

- *Which matching technique you have implemented*
- *Pseudo algorithm of your Implementation*

1. **Pseudo Algorithm :**

```
scores = CalculateScoreUsingBM25(query)
info = {}
for word in query :
        inverted_list_by_term = getInvertedIndex(word)
        for (doc, positionList) in inverted_list_by_term :
                info[doc][word] = positionList
for doc, words in info:
        for word, positionList in words:
                proximity_score = CalculateProximityScore()
                scores[doc] += proximity_score
return scores
```

2. **Technique used** : Used Positional Indexing to get the distance between the terms in a specific document and accordingly updated the score for a particular document obtained using Okapi BM 25 based on the proximity score.

3. **Analysis** :
   The score for Okapi BM25 on a Stemmed Index resulted in a precision score of **0.2925.** After using proximity search using Okapi BM25 for stemmed index, the precision increased to **0.2940** which shows an increase in precision after using proximity search.