**README**

**1.Inverted Index Construction**

Stages involved in constructing index for the given data set follows as,

1. Parsing file to create intermediate postings.
2. Merge and sort into one file used to construct actual index.
3. Construct actual postings to generate inverted list for the word term.
4. Generate page table to retrieve URL of the given docID.
5. Build Lexicon that stores metadata for each word term found in the data set.

**2.Design**

**2.1. Creating Intermediate posting**

* Used map container in C++ to count the occurrences of word term present in dataset msmarco-docs.trec.
* And, used map container to map between docid and URLs to output into page\_table.txt.
* Utilized Regex to filter non-English characters and extract URL for the document.

A screenshot of a computer

Description automatically generated with medium confidence.

**2.2. Creating Intermediate posting**

Used linux sort to sort each misc\_\* files generated by generate\_postings.cpp.

First, sorted in batch size of 100 and then piped into single file (s\_misc1.txt ….smisc9.txt)with below operation,

Sort –parallel=15 misc1.txt>s\_misc1.txt.

This executed 2 mins for each file. Hence, cumulatively it took 20 mins for initial first phase sort.

Then merged all s\_misc files into one using,

Sort -m smisc\_\* > intermediate\_postings.txt.

**2.3. Construct Inverted List**

* For each word term in the intermediate\_posting, all DocID and frequency is pushed as pair into the vector. This is method was helpful, to encode all the DOCID.
* If buffer size exceeds, for all the items in the map containing posting in the form (Word, <DocID, Freq>), metdata entry is created for all terms and the container is reset.

**Text

Description automatically generated**

* For each word term in the container, respective docID stream is encoded using varbyte compression as it is suitable for this implementation that makes use of vector of docID.

Text

Description automatically generated

**3. Execution**

For 22GB dataset(3.5 million docs) given, program able to parse about (3 million docs)20GB in 6.5 hours.

Linux sort completed in about 45 mins to sort and merge into single (intermediate\_postings.txt) file.

Finally, meta.cpp constructed inverted list and lexicon in about 5 hours.

|  |  |  |
| --- | --- | --- |
| **FILE** | **SIZE** | **DESCRIPTION** |
| inverted\_list.dat | ~2.3GB | Encoded docid, freq stream for each word term |
| lexican.txt | ~450MB | Contains metadata of each word term for the inverted list |
| Page\_table.txt | ~225MB | Contains mapping of docid and URL. |

**4. Limitations.**

* Inverted list is constructed only for English characters.
* Any numeric term is ignored to create intermediate postings.
* Performance of program to create intermediate postings is not satisfactory due to delay in file read/write, and insertion of element into container to store the occurrences.
* Inbuilt sorting is used to sort the docIDs for preparing input to Varbyte encoder which takes hit in the performance of meta.cpp to construct inverted\_list.