IRE – MINI PROJECT REPORT

Directory Structure

```
2021202027
├── index.py
├── index.sh
├── readme.txt
├── search.py
├── stat.txt
├── stopwords.txt
├── dict_to_text.py
├── merge.py
├── preprocessing.py
├── split.py
├── requirements.txt
└── index
    ├── final
    │   ├── word_dict_fin_0.txt
    │   ├── word_dict_fin_1.txt
    │   ├── ...
    │   └── ...
    ├── body_index_0.txt
    ├── body_index_1.txt
    ├── ...
    └── ...
```

The optimizations used

- Used Primary and Secondary Indexes to faster and more efficient searching. Also the size of the index is also significantly reduced due to using this index structure.
- In Primary Index, the word dictionary files are stored in sorted order in small chunks resulting in faster search time and more efficient searching
- The titles are also divided into primary and secondary indexes where primary indexes contain titles stored in sorted order on title_id

- Primary index contains seek offset values for secondary indexes which results in faster search

Improvements in terms of time and space from those optimizations

- Searching in primary index takes O(logn) time due to files being stored in sorted order where n = number of files in primary index.
- Searching in secondary index takes O(1) time since primary index contains seek offset values through which the results are directly extracted from the secondary index
- There are no duplicates word present in the primary index resulting in a lot more compressed index size and the secondary indices also do not contain duplicate entries in terms of title id and frequency

Index creation time and size

Index creation time of whole 90GB dump ~ 12 hours

Index Size = 16.6GB

The format of the final index created -> what are the keys and values

Index is made up to of 2 types - Primary and Secondary Index

- Primary Index
  - Primary Index is in final folder inside the index directory
  - Primary Index consists of all unique words sorted in ascending order and split into multiple files
  - After Each words there are batch entries of the format (batch_number: list of 6 values)
  - These list of 6 values are file seek pointer for each category like title,infobox,references,etc
  - The batch number denotes the file number to look into for each word in main index folder

  - There is title primary index which consists of (batch_number: seek_offset token_count) where token count is the total token counts in the page
  - Key Value Example for word_dict
    sachin-0:76 12 342 32 n n,2:23 12 45 n n 2
    word-batch_number_1:[list of 6 integers],batch_number_2:[lisst of 6 integers]
  - List notation index values for each of the 6 integers
    {'body': 0, 'title': 1, 'ext links': 2, 'category': 3, 'infobox': 4, 'ref links': 5}
  - Key Value Example for title_dict
    101-0:43 23,4:46 78
    title_id:batch_number_1:[seek_offset, token_count],
    batch_number_2:[seek_offset, token_count]

- Secondary Index
  - There 6 types of files for each category which have multiple batches
  - In each file, each line is an unique word the contents of the line are of the format (title_id:frequency)
  - These occurrences are seperated by commas and can be used to extract all documents containing the particular word in respective category
  - There are title indexes which store the title name for each page/document which can be accessed via primary title index
  - Key Value Example for body_index
    101:2,322:1,3999:4
    title_id_1:frequency,title_id_2:frequency, title_id_3:frequency,
  - Similary other types of indexes also have same structure in secondary index