PROJECT REPORT- PHASE 2

Course: CSCE 5200 – Information Retrieval
Project: Text Parser and Indexer – Phase 2
Dataset: TREC FT911 Collection (5,368 documents)

Name: Sadvik Kondadi

Student Id:11785837

1.System Design:

- Implementation The Information Retrieval Engine was written in Java in two modules:
- Porter.java - provides the default implementation of Porter Stemming suffix stripping algorithm.
- Indexer.java - constructs the dictionary and forward index as well as inverted index of the entire FT911 collection.
- The program loads the stopwordlist.txt which contains 523 stopwords.
- Then it reads parser_output.txt that has 33,180 distinct stemmed terms of Phase 1.
- These terms are placed in an alphabetical order and allocated ID numbers in series (1 -33,180).
- Every file in ./ft911/ is read by the extraction of text between the <TEXT> and </TEXT> tags and the tokenizing of alphabetic words, the elimination of stopwords, and frequency counts.

Output files generated:

forward index.txt - gives term id and document frequency.

inverted_index.txt - contains the list of document id and frequency of the words.

dictionary.txt - dictionary alphabetically sorted.

The file paths are all relative which makes them compatible across systems.

2. Indexing Performance:

Metric	Value
Total Documents Indexed	5,368
Unique Terms	33,180
Stopwords	523
Indexing Time	~ 3.2 seconds
Index file size	~ 15 MB(total)
Search Type	Exact term lookup

The indexing was performed on MacBook Air (M3 chip, 8 GB RAM) with Java 17. Phase 1 was followed by Indexer.java (program based on the results of Phase 1) to process the entire TREC FT911 dataset (5,368 documents).

Execution Steps

1.Stopword Loading:

The program loads 523 stopwords in the file stopwordlist.txt. Their inclusion in tokenization is not allowed to reduce noise and index size.

2.Term Loading & ID Assignment:

The file of the parseroutput.txt 33,180 stemmed terms are read. The terms have been sorted into alphabets and assigned incremental integer numbers (1 - 33,180).

The use of alphabetical ordering guarantees deterministic indexing, i.e. executing the same code multiple times results in the same term IDs and entries in the dictionary.

3.Document Parsing:

All files in the ./ft911/ folder (files ft9111 - ft91115) are read in order. The engine recognizes the tags of <DOCNO> and <TEXT> and reads out the text content, stemming Porter.java, and eliminates all stopwords.

4.Index Construction:

Forward Index: This index is a mapping of termID:frequency pairs in each document which is stored in increasing order of term ID.

Inverted Index: A postings list of pairs of docID:frequency is kept in a record of inverted indexes, which are ordered by document ID.

File Output:

After processing all the documents, three files are written:

forwardindex.txt - 5,368 entries

invertedindex.txt - 33,180 entries

dictionary.txt - 33 180 alphabetically sorted term.

3.Observations and Conclusion:

- The indexer generates totally sorted, consistent indexes and is in synchronization with Phase 1 indexes.
- Pre-ID assignment term sorting by alphabetical terms is to prevent term mapping stability and to provide consistency in the order of the dictionary.
- The system constructs the index of the entire collection of FT911 effectively and enables the interactive search of terms.
- This design is considered to satisfy all the requirements of the project and create a strong base of Boolean retrieval and ranking in later stages.