Information Retrival

Assignment-1

Q1. Inverted Index:

Preprocessing: Using os library I load all folders and traverse them one by one with directory address and load all the documents and make a dictionary in the form of the inverted index . Using NLTK library I use word_tokenization to make tokens for the text of each document and save every token in the dictionary with the key as tokens.

All the tokens have their posting lists in which all documents id stored in a sorted manner and count of all documents augmented at first index.

Methodology: I took input and tokenize it and save it in a list. I processed all NOT operations and save all postings in a list . Now I categorized query in two parts as following:

1.Homogeneous queries: Only containing AND operations

2.Heterogeneous queries: Queries containing both AND and OR operations.

I take precedence of operators as NOT>AND>OR

After resolving all NOT queries, If a query contains heterogeneous property I selected all the postings which conjecting with AND perform optimized_AND on all the lists , which takes intersection on the bases of document frequency in postings and substitute the result in the previous list.

Now we take the union of all postings one by one for all OR operators and get the resultant as output documents and count all the comparison within all the operations.

Assumption: User providing all the valid queries.

Q2: Positional index:

Preprocessing: Using os library I load all folders and traverse them one by one with directory address and load all the documents and make a dictionary in the form of the Positional inverted index . Using NLTK library I use word_tokenization to make tokens for the text of each document and save every token in the dictionary with the key as tokens.

All the tokens have their posting lists in which all documents id stored in a sorted manner and count of all documents augmented at first index and each and every document id stored in a dictionary with the key as document id and their corresponding list with positions in the document for a particular word.

Methodology: we perform the word_tokenization on every query and now I perform searching on the dictionary and retrieve a document for the first word. Now remaining words are searching being done on the basis of the previous retrieved document and with positional differences 1 . All the documents which are not relevant to the query are eliminating in a recursive manner . and in the last or recursion all the documents containing the query stored in a list.

Assumption: Query should be valid means no commas and punctuations.