Boolean Information Retrieval System

Pranay Tarigopula (2018A7PS0237H)
Adithya Jayan Warrier (2018B3A70873H)
Rishi Saimshu Reddy (2018A7PS0181H)

Information Retrieval Assignment I

22 March 2022

Data Preprocessing

When the program is first launched, an `IR_system` object is fetched, and in that process, the `__init__` function of the `IR_system` class is run.

In the `__init__()` function, the function first creates variables to store information regarding the list of documents, stopwords, tokens, inverted-index(a dictionary), which are then sent to preprocessing the documents for our querying system.

`preprocess_text_files()` is the function which the called by the `__init_()` function to process the document list. `preprocess_text_files()` function sends path of each text file to `preprocess()` function. The `preprocess()` function converts the text to lower case, tokenizes the text, removes the stops words, lemmatizes the tokens, and returns it all together as a string to be written into a new file in the folder created to store the files with preprocessed text. `preprocess_text_files()` iterates like this for all the text files and return to the `__init__()`.

After documents are preprocessed until getting a list of lemmatized tokens, an inverted index is built next done by the `construct_inverted_index()` function. This function reads the text from each preprocessed file and the after getting the tokens from them, a new entry is created for unique tokens, and then for each token, the document number is appended to the posting list.

One last step before being able to carry out queries is to build a permuterm index. We have implemented a trie that would store all permutations of a token so that it will be helpful in searching for wildcard words. All of this is taken by the `build_permuterm_index()` function and the `Trie`, `TrieNode` classes. `__init__()` function returns to the main function after completing all the preprocessing steps to now let the user actually query.

Query Processing and Information retrieval

After the user inputs their query (duly following the rules of writing a query), it is sent to `process_user_function()` to get processed. Here, the function takes the query and converts the boolean query into the postfix form to correctly evaluate the query according to the boolean logic order.

In evaluating the postfix boolean query:

1. when you find a token:
   - The logic is first to convert it to lower case and then lemmatize.
   - If the word is a key in the inverted index, its posting list is taken for further evaluation.
   - If it contains (*), the program goes ahead with a wildcard search, where all matching words are found through the preprocessed permuterm index and then the union of the posting lists of all the matching words is taken as the posting list for the wildcard.
   - If the word doesn't exist as a key in the inverted index, it is assumed that the spelling is wrong and the edit distance is calculated for all indexed tokens, and the one with the least edit distance is considered the correct spelling of the word. The posting list of this word is taken as the posting list of the misspelled word.
2. When you find a logical operator
   - NOT is subtracts the posting list from the total set of documents
   - AND the intersection of the right and left posting lists is done
   - OR the union of the right and left posting list is done

After evaluating each unit of the boolean query, it is popped out of the postfix stack. The resultant list of documents is the final answer for the boolean query of the user.

Performance metrics

→ The preprocessing until the user is allowed to write a query takes about 25secs
→ The Result for a query is almost instantaneous (result for a query like W* is returned within a second)

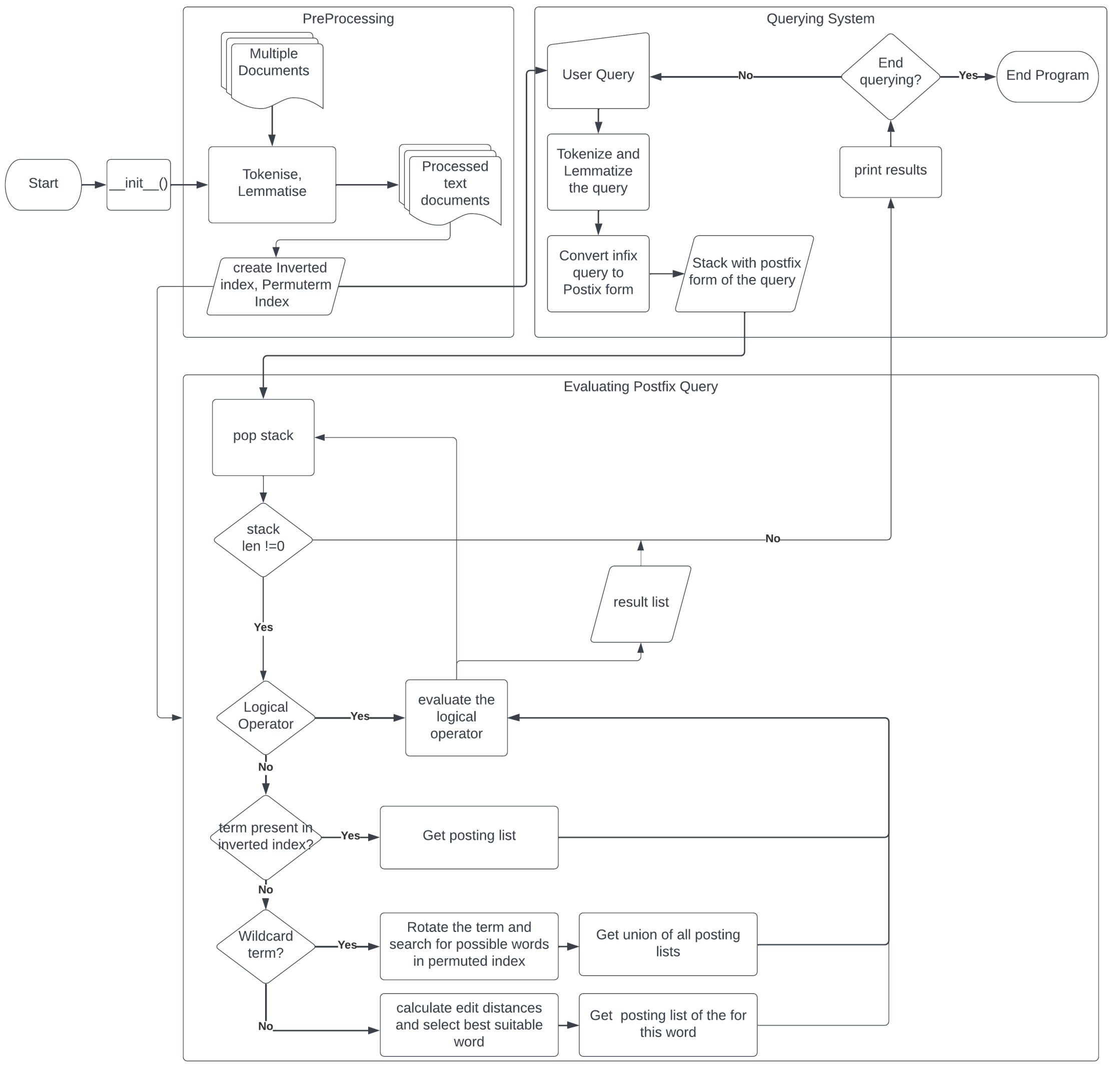
PreProcessing
Querying System
Evaluating Postfix Query
Start
__init__()
Multiple Documents
Tokenise, Lemmatise
Processed text documents
create Inverted index, Permuterm Index
User Query
No
End querying?
Yes
End Program
Tokenize and Lemmatize the query
print results
Convert infix query to Postix form
Stack with postfix form of the query
pop stack
stack len !=0
No
result list
Yes
Logical Operator
Yes
evaluate the logical operator
No
term present in inverted index?
Yes
Get posting list
No
Wildcard term?
Yes
Rotate the term and search for possible words in permuted index
Get union of all posting lists
No
calculate edit distances and select best suitable word
Get posting list of the for this word

Figure 1: Process Flowchart