IR Assignment
CS F469

Name : Prathmesh Srivastava

ID No : 2019A7PS1322H

Boolean search engine

Boolean search is a type of search which makes use of words or symbols to reduce, widen or define the search. An example of a few boolean operators are AND, OR, NOT, +, -. A use case could be if the user wants to search for free games it can be processed as free AND games.

Performance

PreProcessing:

For preprocessing we are removing stop words and performing stemming. Both the processes take linear time.

Indexing:

We are indexing the dataset by visiting each token and then firstly inserting each word in our PermutermIndex and then in our InvertedIndex.

- **Insertion in PermutermIndex:**
  Insertion in PermutermIndex takes O(n*lengthOfEachToken)

- **Insertion in InvertedIndex:**
  We are using a hashmap where each value of the hashmap represents a posting list. Each insertion can at max take O(N) time where N is the size of the posting list for a token.

Searching/Retrieval:

In case of a hit, i.e not wildcard and correct spelling, the complexity to get a posting list is constant time. The complexity to merge two posting lists is linear. So, overall complexity will depend on the number of keywords in the query multiplied by time to retrieve each posting list plus time to merge two posting lists.
In the case of wildcard query, the complexity can increase and we need to get the posting list for each possible term (based on prefix).
In case of incorrect spelling, we need to go to each keyword in our index and check the edit distance. This operation will take O(N) time as the length of the tokens can be assumed to be constant.

Index Structure

PermutermIndex:

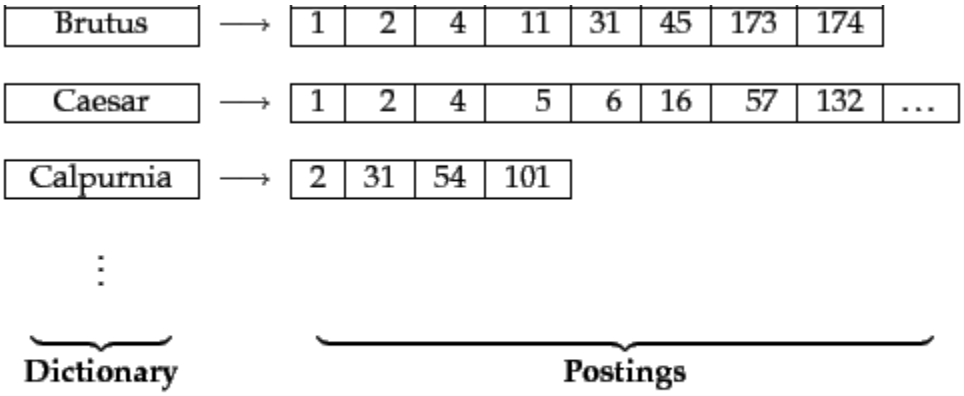
PermutermIndex is implemented using a Trie.

Structure of a TrieNode:

```
class TrieNode:
    """
    TrieNode Class
    TrieNode class is a class to represent the trie node data structure
    It has a children attribute which is a python dictionary and a is_eow to
signify if a word
    ends at that node.
    """
    def __init__(self):
        self.children = dict()
        self.is_eow = False
```

InvertedIndex:

The inverted index is implemented using python dicts and python lists. Each value in the dict represents a posting list which is in the form of a list.



► **Figure 1.2** The two parts of an inverted index. The dictionary is commonly kept in memory, with pointers to each postings list, which is stored on disk.

The posting list elements are in a sorted order.