**Boolean Information Retrieval System**

By - Prithvi Hegde

Kavyanjali Agnihotri

Kishan Koundinya

Introduction and Usage:

We have implemented a Boolean Information retrieval System, whose functionalities include Stopword removal, Lemmatization, Spelling-check and correction using Levenshtein distance, and Wildcard query handling, through the use of K-grams (bigrams)

The user is shown a menu, and has to input 1 to pass a new query, and 0 to exit the program. The user can pass as many queries as they wish to, one at a time. Once the user inputs a query, the program displays what the preprocessed query is, and then searches the document space for relevant documents, and returns the names of the relevant documents in an unordered fashion.

Data Structures Used:

* We have used *python dictionaries* to store the inverted indexes. The dictionary keys are the unique words in the document space, and the values are their respective document IDs in which they occur, which are stored in a List. Looking up values in a python dictionary is very fast due to the use of HashMaps and a lookup operation can be done in O(1) time complexity.
* For the Boolean AND/OR and the Unary NOT operations, we have converted the List of document IDs into a Python Set. The Set data structure in python, which is akin to a List with only unique values, provides us an elegant and fast way to perform Boolean AND/OR operations. These operations are performed in O(n+m) time complexity, where n and m are sizes of the two sets. The NOT (set.difference()) operation is performed in O(n) time complexity, where n is the size of the inverted index.
* We have used a stack to store the Boolean preprocesssed queries, which are popped one-by-one.

Running time for Preprocessing:

The query preprocessing function performs Spell check, Stopword removal and Lemmatization. The time complexity is O(n), where n is the number of words in our document space.

Running time for Building Inverted Index:

The running time for building the Inverted Index is O(n), where n is the number of words in our document space.

Running time for Search/Retrieval:

The running time for search or retrieval is O(n), where n is the number of words in the document space.