Information Retrieval Assignment 1

ON

Boolean Information Retrieval

BY

Name of Students	ID No.	Discipline
Atishay Jain	2019A7PS0106H	CS
Neil Mehta	2019AAPS0177H	CS
Fenil Bardoliya	2019A7PS0152H	CS

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

(Hyderabad Campus)

(23rd March 2022)

Problem Statement

- ➢ This assignment is aimed at designing and developing Boolean Information Retrieval System, i.e., to return those documents (specifically their names from corpus/dataset given) which satisfy Boolean (AND, OR and NOT with their combinations).
- ➢ The Boolean Information Retrieval System should include the following features / pre-processing steps:
  A. Stopword Removal: Remove the common stop words from the corpus.
  B. Stemming or Lemmatization: Employ either one of the techniques for normalization.
  C. Wildcard Query Handling: Any one of the techniques among Permuterm or K-Gram index should be used for wildcard query management.
  D. Spelling Correction: Edit Distance Method should be employed to correct misspelled words.

Introduction

This report aims to analyze and explain the design choices which have been taken while implementing the Boolean Information Retrieval System which has been designed using the corpus of text documents as dataset by following the format given below :

1. System Architecture
2. TestCases
3. Observations and Design Choices
4. Conclusions

System Architecture

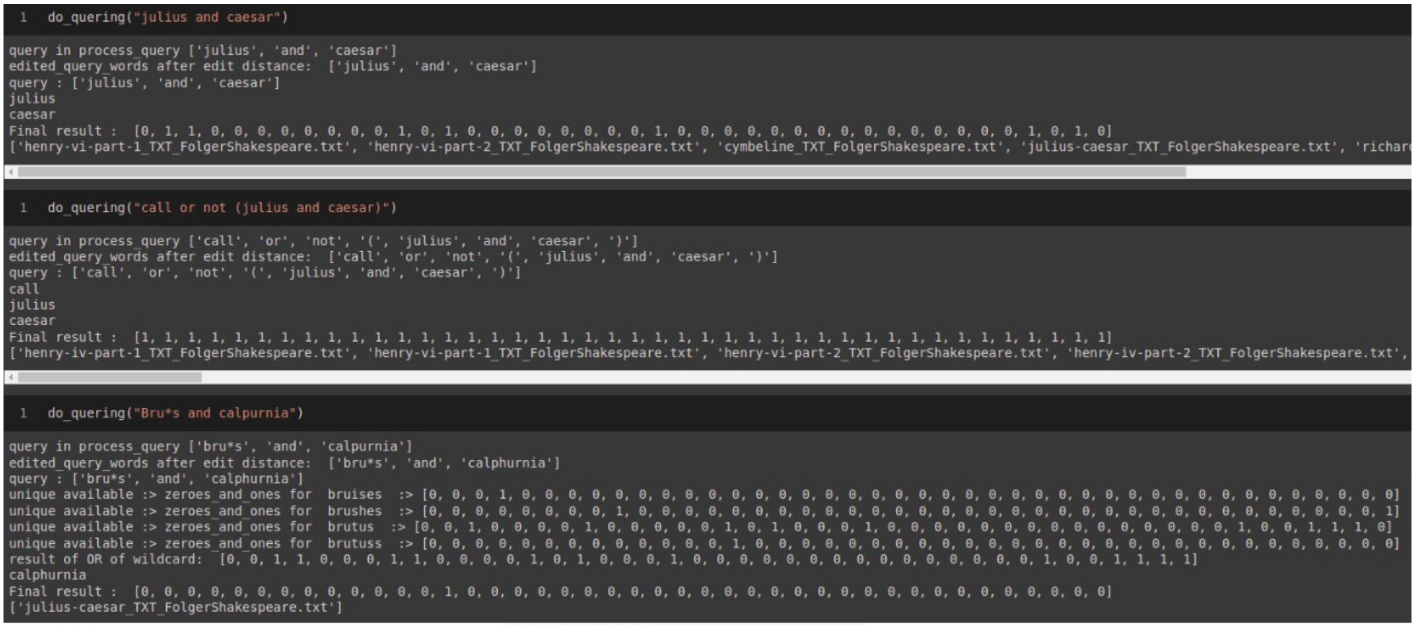
Preprocessing Steps :

- ➢ Tokenization : Tokenized all the text documents from the corpus.
- ➢ Stopword Removal : Removed common predefined stop words as well as single character words from the corpus.

Query Processing :

- ➢ Spelling Correction : Edit distance method has been employed to correct misspelled spellings.
- ➢ WildCard Query Handling : n-gram index has been used for wildcard query management.
- ➢ Stemming : Employed stemming as a normalization technique.

TestCases



```
do_quering("julius and caesar")
```

```
query in process_query ['julius', 'and', 'caesar']
edited_query_words after edit distance:  ['julius', 'and', 'caesar']
query : ['julius', 'and', 'caesar']
julius
caesar
Final result :  [0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0]
['henry-vi-part-1_TXT_FolgerShakespeare.txt', 'henry-vi-part-2_TXT_FolgerShakespeare.txt', 'cymbeline_TXT_FolgerShakespeare.txt', 'julius-caesar_TXT_FolgerShakespeare.txt', 'richar
```

```
do_quering("call or not (julius and caesar)")
```

```
query in process_query ['call', 'or', 'not', '(', 'julius', 'and', 'caesar', ')']
edited_query_words after edit distance:  ['call', 'or', 'not', '(', 'julius', 'and', 'caesar', ')']
query : ['call', 'or', 'not', '(', 'julius', 'and', 'caesar', ')']
call
julius
caesar
Final result :  [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
['henry-iv-part-1_TXT_FolgerShakespeare.txt', 'henry-vi-part-1_TXT_FolgerShakespeare.txt', 'henry-vi-part-2_TXT_FolgerShakespeare.txt', 'henry-iv-part-2_TXT_FolgerShakespeare.txt',
```

```
do_quering("Bru*s and calpurnia")
```

```
query in process_query ['bru*s', 'and', 'calpurnia']
edited_query_words after edit distance:  ['bru*s', 'and', 'calphurnia']
query : ['bru*s', 'and', 'calphurnia']
unique available :> zeroes_and_ones for  bruises  :> [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
unique available :> zeroes_and_ones for  brushes  :> [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
unique available :> zeroes_and_ones for  brutus  :> [0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0]
unique available :> zeroes_and_ones for  brutuss  :> [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
result of OR of wildcard:  [0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1]
calphurnia
Final result :  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
['julius-caesar_TXT_FolgerShakespeare.txt']
```

Observation and Design Choices

- ➢ Tokenization has been done using built-in nltk library functions to get individual words from the given corpus of text documents.

- ➢ Stopword removal along with removing single character words is then performed to remove the low-level information from our text in order to give more focus to the important information.

- ➢ The data structure used for storing the keywords is an inverted index / posting list which is a database index storing a mapping from content, such as words or numbers, to its locations in a document or a set of documents.

- ➢ Normalization techniques help in reducing the number of unique tokens present in the text, removing the variations in a text. and also cleaning the text by removing redundant information.

- ➢ Stemming is an elementary rule-based process for removing inflationary forms from a given token.We have used it since it is easier to implement and faster to run.

- ➢ Edit distance has been used for spelling correction which is a way of quantifying how dissimilar two strings are to one another by counting the minimum number of operations required to transform one string into the other.

➢ N-gram index which refers to the sequence of n terms (or generally tokens) from a document by moving a floating window from the begin to the end of the document, has been used for performing wildcard searches.

➢ While the permuterm index is simple, it can lead to a considerable blow up from the number of rotations per term; for a dictionary of English terms, this can represent an almost ten-fold space increase. Hence we have used n-gram index.

Conclusions

We have implemented a Boolean Information Retrieval System with the following features :

➢ Tokenization
➢ Stopword Removal
➢ Normalization using stemming as a technique
➢ Wildcard query handling using n-gram indexing
➢ Spelling correction using edit distance method