INFORMATION RETRIEVAL

ASSIGNMENT 1

Group No 24

Waquar Shamsi (MT20073)

Reshan Faraz (Phd19006)

Divya Pandey (MT20128)

Shubham Bhansali (MT20105)

Kunal Anand (2018293)

ASSUMPTIONS:

1. The user will input space and/or comma separated values.
2. The dataset folder must be present besides the code file.

PRE-PROCESSING STEPS:

We used *Pandas ,mumpy and nltk* libraries for preprocessing.

We read all the files (467 = 452 + 17) and perform the following preprocessing:

1. **Tokenization:**

   Given a character sequence and a defined document unit, tokenization is the task of chopping it up into pieces, called *tokens* , perhaps at the same time throwing away certain characters, such as punctuation. Here is an example of tokenization.

2. **Remove Stop Words:**

   After tokenization we removed the stop word.S top word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query.

3. **Filter Out Punctuation:**

   After that we filter out the punctuation.

4. **Lemmatization:**

   Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma.

5. **LowerCase:**

   After Lemmatization we lower case the words.

Implementing Unigram inverted index data structure

After that we create a dictionary of term vs posting list , Where we maintain the posting list of each term in a list in non decreasing order. All the operations are performed on this data structure we also attached the .pkl for the same.

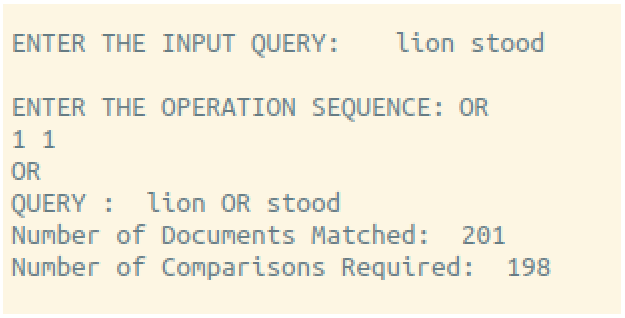
Following are the steps used to generated unigram inverted data structure:

We create a dictionary which has word/term as key and posting list as the value. We sorted the dictionary based on the term in non decreasing order. The posting list itself is in non decreasing order which contain the document id of each document

We also create another dictionary which has word/term as key and frequency of each word as values. Here frequency means the number of documents contains the word/term.

C : Support for the following Queries

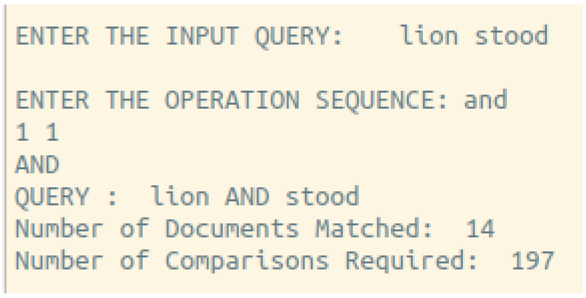
x OR y:



```
ENTER THE INPUT QUERY:    lion stood

ENTER THE OPERATION SEQUENCE: OR
1 1
OR
QUERY :  lion OR stood
Number of Documents Matched:  201
Number of Comparisons Required:  198
```

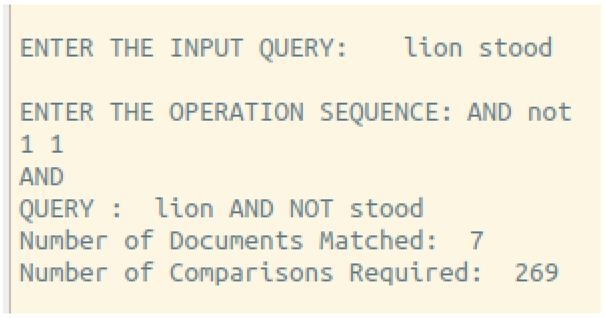
x AND y:



```
ENTER THE INPUT QUERY:    lion stood

ENTER THE OPERATION SEQUENCE: and
1 1
AND
QUERY :  lion AND stood
Number of Documents Matched:  14
Number of Comparisons Required:  197
```

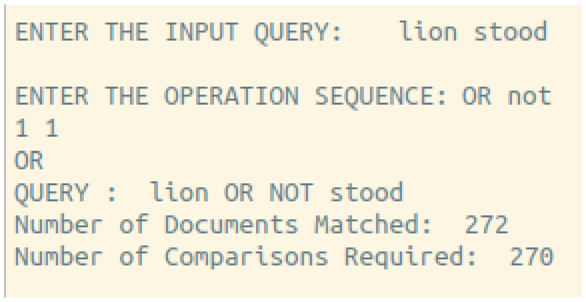
x AND NOT y:



```
ENTER THE INPUT QUERY:    lion stood

ENTER THE OPERATION SEQUENCE: AND not
1 1
AND
QUERY :  lion AND NOT stood
Number of Documents Matched:  7
Number of Comparisons Required:  269
```

x OR NOT y:



```
ENTER THE INPUT QUERY:    lion stood

ENTER THE OPERATION SEQUENCE: OR not
1 1
OR
QUERY :  lion OR NOT stood
Number of Documents Matched:  272
Number of Comparisons Required:  270
```

References : https://nlp.stanford.edu/IR-book/