Assignment 1 - Information Retrieval (CSE508)

- *Shreyash Arya (2015097)*

1. Use the 20newsgroups dataset to build a unigram inverted index.

- Unigram inverted index is built on all the files present in '20newgroups' dataset by iterating over all the files in each folder.
- Preprocessing has been done on all the documents present:
    - Removing header-footer from the files as they contain non-relevant/junk data.
    - Removing HTML tags using BeautifulSoup inbuilt module.
    - Fixing the contractions (Eg. don't → do not) using contractions python library.
    - Tokenization using NLTK's word tokenizer.
    - Removing non-ascii characters using Unicode.
    - Lowercasing the document data.
    - Removing punctuations using regex (r'([^\w\s])|_+').
    - Removing stopwords using the NLTK's English stop-words dictionary.
    - Stemming using NLTK's Porter Stemmer.
- Dictionary has been generated considering each unique term as key and value as a list containing the overall term frequency in the documents and a list which contains the document ids in which the term has appeared in the sorted order.
    - *{ 'key' : [total_frequency, [doc_id1, doc_id2, ...], ...] }*
- The document names are mapped to indexes as a counter from 1 to 19997.
- The mapping from file index to file name has been stored as pickle files.
    - *(filename_mappings.pkl)*
- Generated inverted indexes are stored as pickle files so that the program can directly load the indexes and do quick query searching.
    - *(inverted_index.pkl)*
- Vocabulary size: 127089 (after preprocessing); Total documents: 19997; Stopwords list size: 179

2. Further, provide support for the following commands:

- **x OR y**
- **x AND y**
- **x AND NOT y**
- **x OR NOT y**

Where x and y would be taken as input from the user.

- For performing the above commands, AND, OR and NOT functions are implemented using the set theory functions implemented in python.
- x and y are taken as user input as mentioned.

- x and y are stemmed using Porter Stemmer to be matched to the stemmed inverted index terms.
- Merge posting algorithm is used for query processing.

Results:

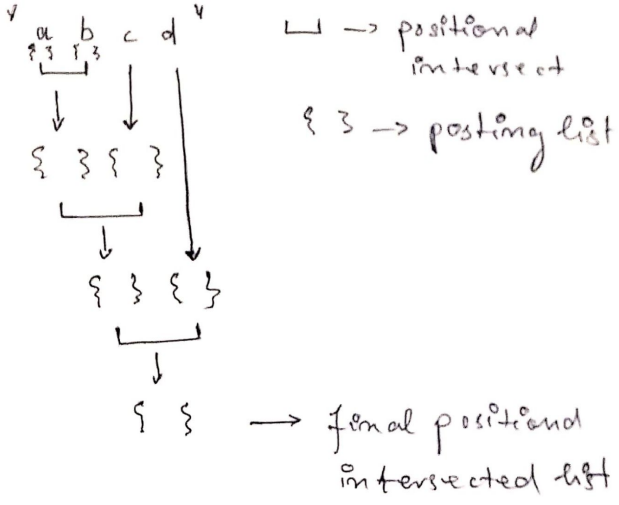
Below is the result for **x AND y** query where x = computer and y = graph:

The query returns all the documents which contain both computer and graph (specific to above AND query). The total number of documents containing query is also returned.

Other results can be tested directly by running the program on the command line as many results are returned to be displayed here.

3. Provide support for searching for phrase queries using Positional Indexes. (For this question, build index only on comp.graphics and rec.motorcycles)

- For providing the support for phrase queries, positional indexes have been implemented which has a similar structure as the inverted index but also contains the position of the term appearing in each document. So, the document ids are made as the keys and then a corresponding list of term indexes is stored.
    - *{ 'key' : [total_frequency, ['doc_id1' : [term_index1, term_index2, ..], 'doc_id2': [term_index1, term_index2, ..], ...], ...] }*
- Preprocessing is done similar to what was done in inverted index creation but stopwords are not removed as the queries like: "to be are not to be" will not be searched if we remove the stopwords.
- The query terms are stemmed using Porter Stemmer to be matched to the stemmed positional index terms.
- The phrasal indexes are created on comp.graphics and rec.motorcyles as mentioned and stored as pickle files for fast query processing.
    - *(phrasal_inverted_index_with_stopwords_q3.pkl)*
- The document names are mapped to indexes as a counter from 1 to 19997.
- The mapping from file name to file index and vice-versa has been stored as pickle files.
    - *(file_to_index_mappings_with_stopwords_q3.pkl, index_to_file_mappings_with_stopwords_q3.pkl)*
- Vocabulary size: 20150 (after preprocessing)
- Positional intersect algorithm is implemented for matching two terms in a query in order which returns the combined/intersected posting lists for the two terms. This is extended for the overall phrase by combining the results from two previous terms as explained below: "a b c d" is the phrase query.



Results:

Below are the results for a query ‘will be sent’ which returns the documents containing the phrase and the position where the last term of the phrase occurs. Also, count of such documents is returned.

```
In [21]: run my_positional_indexing.py

Enter the phrase (-1 to exit): will be sent
('comp.graphics/38609', [227])
('comp.graphics/37261', [226])
('comp.graphics/38971', [133])
('comp.graphics/38851', [4695])
('comp.graphics/38920', [219])
('comp.graphics/38377', [4584])
6
```

Another query: “have to be”

```
Enter the phrase (-1 to exit): have to be
('rec.motorcycles/105160', [144])
('comp.graphics/38937', [22])
('comp.graphics/38428', [122])
('comp.graphics/38431', [42])
('rec.motorcycles/103154', [173])
('rec.motorcycles/104581', [50])
('rec.motorcycles/104453', [58])
('rec.motorcycles/104328', [20])
('comp.graphics/38452', [92])
('rec.motorcycles/104722', [13])
('comp.graphics/38588', [33])
('comp.graphics/38985', [142])
('comp.graphics/39639', [163])
('comp.graphics/38363', [225])
('rec.motorcycles/105110', [59])
('comp.graphics/38922', [89])
('comp.graphics/38382', [14])
('rec.motorcycles/104850', [329])
('rec.motorcycles/102616', [185, 3696])
('rec.motorcycles/103220', [330])
('rec.motorcycles/104412', [134])
('rec.motorcycles/105149', [164])
('rec.motorcycles/104553', [52])
23
```

To run the program:

* Keep all the dictionaries in the same folder as the code files.
* Command: $ python 'code_file_name.py'

For part 2, run the code file 'inverted_index.py' and options are provided to run specific queries.
For part 3, run 'positional_indexing.py' and enter the phrase query (-1 to exit).

References:

https://nlp.stanford.edu/IR-book/html/htmledition/processing-boolean-queries-1.html
https://www.geeksforgeeks.org/python-intersection-two-lists/
https://www.geeksforgeeks.org/python-union-two-lists/