Information Retrieval (IT458)

Assignment 1: Inverted Index for News Corpus

Submitted by: Jaidev Chittoria
Roll No.: 181IT119 Date: 12th September

Creating the Corpus:

Google News and many other online news websites were taken into account for creating the corpus. The articles chosen cover all the different fields of news. Further these news articles were combined and stored in a text file which will be used as corpus.

Preprocessing Techniques:

The Preprocessing Techniques which were applied to the corpus are as follows

- Removal of Punctuation signs like (!()-[]{};:'"\ etc.
- Converting the text to lowercase.
- Removing Stop words (The list of stopwords is extracted using the nltk package of Python).
- Stemming : All the morphological variants of the word are reduced to the root word in the document. Stemming helps to reduce multiple morphological variants into a common word which in turn helps to reduce the size of inverted index.

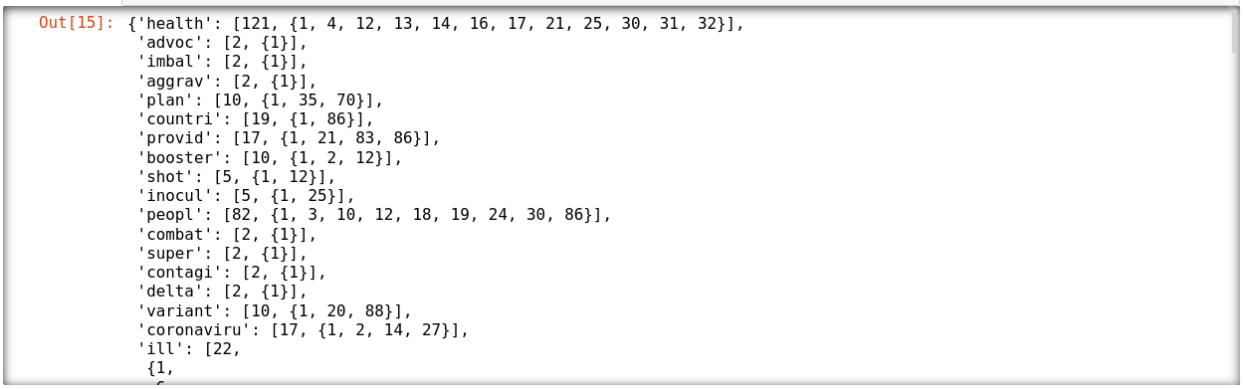
```
In [12]: print('Vocabulary Sizes after various preprocessing steps')
         print('Initial vocabulary size: ',initial_vocab_size)
         print('After tokenization: ',tokenize_vocab_size)
         print('After removing stop words: ',stop_word_vocab_size)
         print('After Stemming',len(stemmed_tokens))

         Vocabulary Sizes after various preprocessing steps
         Initial vocabulary size:  3114
         After tokenization:  3259
         After removing stop words:  1969
         After Stemming 1969
```

Fig.1.: Change in vocabulary after each preprocessing step

Inverted Indexing:

- The data structure which is most suitable for storing the inverted index is a dictionary or in more general sense a hash map. It supports the (key,value) pair format which is helpful for adding/updating and deleting the data.
- The keys for the dictionary will be the index terms which we get after preprocessing. The value corresponding to a particular index key will contain the count of that index term in the vocabulary as well as the document ids of documents in which they appear.
- If we use arrays or linked lists the values of document ids will be repeated in this case which will increase the size of the list and thereby increase the time complexity.
- Therefore, we will be using Set or unordered map in a more general sense to store the values of document ids. This will ensure that if the same term is occurring multiple times in a document, the document id will be added to set only once.
- So, in addition if the particular document id is present, we will not update the set and will just update the frequency. The time complexities for set retrieval and insertion are: O(1) in average case and O(n) in worst case. Set is also helpful as we can perform operations like union, intersection and disjoint which will be helpful in evaluating Boolean queries on the inverted index.



```
Out[15]: {'health': [121, {1, 4, 12, 13, 14, 16, 17, 21, 25, 30, 31, 32}],
          'advoc': [2, {1}],
          'imbal': [2, {1}],
          'aggrav': [2, {1}],
          'plan': [10, {1, 35, 70}],
          'countri': [19, {1, 86}],
          'provid': [17, {1, 21, 83, 86}],
          'booster': [10, {1, 2, 12}],
          'shot': [5, {1, 12}],
          'inocul': [5, {1, 25}],
          'peopl': [82, {1, 3, 10, 12, 18, 19, 24, 30, 86}],
          'combat': [2, {1}],
          'super': [2, {1}],
          'contagi': [2, {1}],
          'delta': [2, {1}],
          'variant': [10, {1, 20, 88}],
          'coronaviru': [17, {1, 2, 14, 27}],
          'ill': [22,
           {1,
```

Fig.2.: An example showcase of inverted index

- We can observe in the above screenshot from terminal output that for the index term ‘plan’ the frequency is 10 and its set contains 3 document ids. This indicates that the term repeats many times in the documents.

Boolean Queries:

- The Boolean queries can be handled by traversing the query string and executing the methods specified for AND, OR and NOT sets in the program.
- We initialize an empty set that will store our final computed result. We will start traversing the query string. As soon as we encounter any word which is not from [AND, OR, NOT], we will know it is an index term and needs to be processed. We will find its posting in our inverted index.
- We will take its set of document ids and pass it to the relevant function of and_Sets(set1,set2), or_Sets(set1,set2) or not_Sets(set1,set2). Here the appropriate function will be decided by the term which is just previous to the current term in the query string and the second parameter passed to the string will be the set we initialized to store the result.

Case 1:

and_Sets(set1,set2) is called if the previous word is AND. It is the intersection of two sets s1 and s2. Time complexity will be O(min(len(s1),len(s2)))

Case 2:

or_Sets(set1,set2) is called if the previous word is OR. It is the union of two sets s1 and s2. Time complexity will be O(len(s1)+ len(s2))

Case 3:

not_Sets(set1, set2) is called if the previous word is NOT.It is the difference of two sets s1 and s2. In case of only one set specified, the
another one will be taken as a Universal set. Time complexity will be O(len(s1)

```
In [18]: #performing queries
         query1 = "bharat OR delta OR low NOT given"
         search_doc(query1)

         Query: bharat OR delta OR low NOT given
         bharat [5, {50, 52}]
         delta [2, {1}]
         low [6, {64, 33, 68, 71, 86}]
         given [26, {3, 101, 13, 48, 30}]
         Result Set: {64, 1, 33, 68, 71, 50, 52, 86}
```

```
In [19]: query2 = "third AND pandem AND low"
         search_doc(query2)

         Query: third AND pandem AND low
         third [10, {3, 28, 13}]
         pandem [2, {5}]
         low [6, {64, 33, 68, 71, 86}]
         Result Set: Empty Set
```

```
In [20]: query3 = "flu OR vaccin NOT bharat"
         search_doc(query3)

         Query: flu OR vaccin NOT bharat
         flu [5, {5, 6}]
         vaccin [589, {2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 21, 25, 27, 30, 52, 86, 107}]
         bharat [5, {50, 52}]
         Result Set: {2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 21, 86, 25, 27, 30, 107}
```

Fig.3: The three queries that were performed

Query.1: “bharat OR delta OR low NOT given”:

{50,52} or {1} or {64,33,68,71,86} not {3,101,13,48,30} = {50,52,1,64,33,68,71,86} not {3,101,13,48,30} = {64,1,33,68,71,50,52,86}= RESULT -> hence verified!

Query.2: "third AND pandem AND low":

{3, 28, 13} and {5} and {64, 33, 68, 71, 86} = {} and {64, 33, 68, 71, 86} = {} (empty set) = RESULT -> hence verified!

Query.3: "flu OR vaccin NOT bharat":

{5, 6} or {2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 21, 25, 27, 30, 52, 86, 107} not {50, 52}= {2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 21, 86, 25, 27, 30, 107}= RESULT -> hence verified!