IT458 – Information Retrieval

Assignment – 1

Name: Niraj Nandish

Roll No: 191IT234

Question:

Construct the inverted index representation for a corpus of any 100 articles from the Incredible India website (https://www.incredibleindia.org/) across different categories like Heritage, Adventure, Art, Food & Cuisine, Nature & Wildlife, Culture etc (use the full text from each article as one document).

Solution:

Collab Link:
https://colab.research.google.com/drive/1485YcOM3JD_MQYQYkqEAnBI2336IaT8D?usp=sharing

I scrapped 100 articles from the Incredible India website across different categories and used Beautiful Soup 4 to extract the data from the HTML files.

a. **Observe and report the effect of different preprocessing techniques applied to the corpus, and the changes in the vocabulary size w.r.t. final index terms, when compared to the number of initial tokens (show each step in the process clearly)**

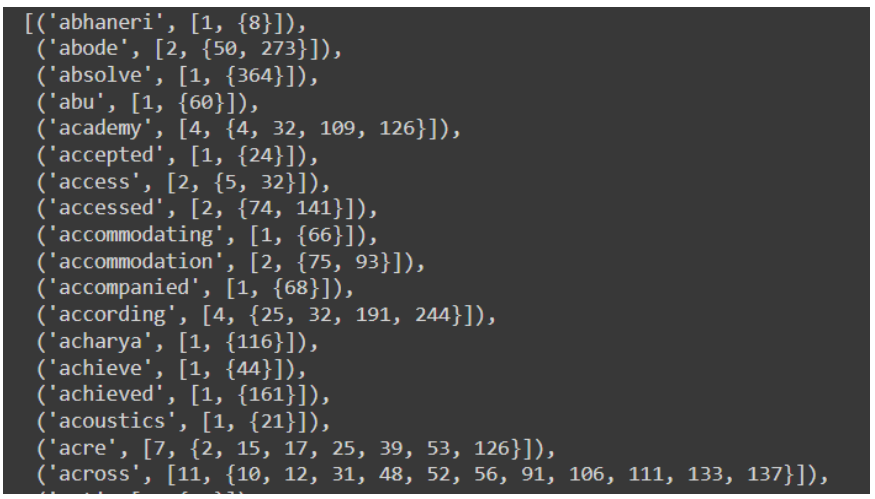
Preprocessing	**Size of corpus (unique index terms)**
Before pre-processing	5085
Lowercase and removing special characters	4800
Removing non-alphanumeric words	4016

Removing numbers	3842
Removing punctuation marks	3842
Removing extra white spaces	3842
Removing stop words	3735

Following are the preprocessing techniques used :

- Lowercase conversion: Converts all text into lower case letters.
- Remove non-alphanumeric characters: We keep only alphanumeric characters and remove all other special characters.
- Remove numeric characters: We remove all the numeric characters from the documents.
- Remove punctuation marks: We remove all the punctuations from the documents.
- Remove extra white space: We remove extra white spaces from the document.
- Removing stop words: We first tokenize the text and then remove the stop words. In tokenization we break up the text into pieces such as words, keywords, phrases, symbols and other elements called tokens. Tokens can be individual words, phrases or even whole sentences. Also in the process of tokenization, some characters like punctuation marks are discarded. Removing stop words is done to reduce recurring words such as “a”, “the” etc. Hence, vocabulary size is reduced by removing them and we also remove the low-level information from our text in order to give more focus to the important information.

Following is the inverted index :



b. **What type of data structure may be most optimal in storing this index? Compare and provide a detailed analysis w.r.t. the different data structure choices and give the cost analysis from storage and retrieval/insertion/deletion perspectives.**

Data structure	Storage	Retrieval	Insertion	Updation	Deletion
Sorted array	O(n)	O(log n)	O(n * log n)	O(log n)	O(n * log n)
Hash map	O(n)	O(1)	O(n)	O(1)	O(n)
BTree	O(n)	O(log n)	O(log n)	O(log n)	O(log n)
Linked list	O(n)	O(n)	O(n)	O(n)	O(n)
Tries	O(a * m * n)	O(m)	O(m)	O(m)	O(m)

Linked List, Arrays, and HashMap will give out sparse data structures. So, HashMap is the suitable data structure. An inverted index is an index data structure storing a mapping from content, such as words or numbers, to its locations in a document or a set of documents. HashMap data structure directs the user from a word to a document or a web page. Also, the retrieval time and searching time for HashMap compared to List and Array is O(1) which is faster than the rest.

c. **Using the constructed inverted index, perform some sample Boolean queries of the pattern shown below. What is the time complexity of finding the results assuming that there are N postings lists in the inverted index? Analyze and explain in detail.**

    1. **term1 AND term2 AND term3**
    2. **term1 OR term2 AND NOT term3**

Complexity analysis of Boolean queries

Worst case time complexity of AND is **O(N**).

Worst case time complexity of OR is **O(N).**

Worst case time complexity of NOT is **O(N).**

If there are M boolean operators in the query, time complexity of finding result is **O(M * N).**

Query 1: built AND temple AND heritage
For the above query, following is the resultant set.
Result Set: {4, 14}

```
Query: built AND temple AND heritage
built -> [51, {0, 4, 8, 11, 12, 13, 14, 15, 16, 18, 531, 19, 21, 22, 23, 24, 25, 26, 27, 156, 30, 32, 35, 37, 39, 41, 170, 43, 45, 47, 180, 53, 56, 58, 59, 5
temple -> [71, {1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 14, 15, 145, 18, 19, 148, 21, 22, 20, 24, 25, 26, 155, 28, 284, 30, 23, 32, 33, 34, 291, 164, 37, 166, 40, 41
heritage -> [21, {1, 4, 6, 14, 277, 28, 159, 36, 38, 44, 47, 49, 50, 181, 60, 65, 71, 78, 85, 86, 92}]
Result Set: {4, 14}
```

Query 2: built OR food NOT shop
For the above query, following is the resultant set.
Result Set: {0, 4, 6, 8, 11, 12, 13, 14, 15, 16, 18, 531, 19, 21, 22, 23, 24, 25, 26, 27, 156, 30, 32, 35, 37, 39, 41, 170, 43, 45, 47, 180, 53, 56, 58, 59, 575, 64, 66, 69, 72, 330, 75, 216, 91, 221, 99, 356, 485, 103, 104, 108, 117, 123}

```
Query: built OR food NOT shop
built -> [51, {0, 4, 8, 11, 12, 13, 14, 15, 16, 18, 531, 19, 21, 22, 23, 24, 25, 26, 27, 156, 30, 32, 35, 37, 39, 41, 170, 43, 45, 47, 180, 53, 56, 58, 59, 5
food -> [5, {32, 6, 216, 25, 221}]
shop -> [3, {81, 52, 71}]
Result Set: {0, 4, 6, 8, 11, 12, 13, 14, 15, 16, 18, 531, 19, 21, 22, 23, 24, 25, 26, 27, 156, 30, 32, 35, 37, 39, 41, 170, 43, 45, 47, 180, 53, 56, 58, 59,
```

Query 3: dried AND dropped AND eight
For the above query, the resultant set was an empty set.

```
Query: nine AND building AND temple
nine -> [3, {48, 58, 167}]
building -> [15, {97, 2, 68, 7, 73, 44, 109, 14, 30, 84, 22, 118, 123, 61, 446}]
temple -> [71, {1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 14, 15, 145, 18, 19, 148, 21, 22, 20, 24, 25, 26, 155, 28, 284, 30, 23, 32, 33, 34, 291, 164, 37, 166, 40, 41
Result Set: Empty Set
```

Query 4: nine OR building AND NOT temple
For the above query, following is the resultant set.
Result Set: {97, 167, 73, 109, 84, 118, 58, 123, 446}

```
Query: nine OR building AND NOT temple
nine -> [3, {48, 58, 167}]
building -> [15, {97, 2, 68, 7, 73, 44, 109, 14, 30, 84, 22, 118, 123, 61, 446}]
temple -> [71, {1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 14, 15, 145, 18, 19, 148, 21, 22, 20, 24, 25, 26, 155, 28, 284, 30, 23, 32, 33, 34, 291, 164, 37, 166, 40, 41
Result Set: {97, 167, 73, 109, 84, 118, 58, 123, 446}
```