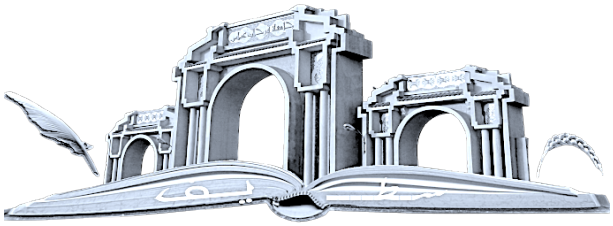

Farhat Abbas University

Course: Information Reteival
Speciality : IDTW
Report

Course Work

Students :
mohamed islem Belhaddad

Supervisor :
Dr.Harrag fouzi

September 7, 2023

Contents

1 The top methods used for tokenisation and stemming

1.1 Tokenisation

for tokenization (spliting the collection) I used “compile” method from “re” module, with specifying the pattern to "(\w{0,})".

1.2 Stemming

we use function porter2().stem(sentence) from Python stemming module, it helps to work the process of producing morphological variants of a root/base word.

2 How we implemented the inverted index?

1. we implemented tokenisation and stemming on all headlines and all texts and we combined them both for every single document.
2. We stored the inverted index in a dictionary with the key representing the “term” and value representing a list that containt a dictionary represent document with positions which the keys represent document numbers and values represent an array containing positions.
and a value "count" represent the document frequency which be incremented if the same word appears in a different document.

3 Efficient Implementation of Four Search Functions

3.1 Phrase search

1. we used a function called phrasesearch(i, phrase) with ’i’ is the distance of between terms and phrase contains the words of the searched query.
2. the function is used for both phrase search and proximity search, if phrase search ’i=1’, if proximity search ’i’ is passed from proximity search method.
3. we create a loops through all positions that both terms occur in, and adds to list if distance between terms <= i.
4. then return a list of documents retrieved.

3.2 Proximity search

1. format query(tokenisation, stemming...).
2. The maximum number of intermediate words between the search words can be taken from query "i".
3. send preprocced query to phrase search with "i" being the distance given.

3.3 Boolean search

1. we get the type of boolean query (AND, OR, AND NOT, OR NOT), splits into the two terms mentioned.
2. If either term is a "phrase search" then get results from "phrase method" .
3. A document will be retrieved for boolean query if it is logically true as exact match in document.
4. then return a list of documents retrieved.

3.4 RankedIR Search using TFIDF

1. we retrieve a vector from index that contains number of occurence of each term in the query in each document of the collection(Tf).
2. we calculate Idf of that term which is (inverse document frequency) of t by the formula log10(N/df) where N is the number of documents in the collection and Df(t) is the number of documents that contain t).
3. Then we compute the weight of term = Tf*Idf (The tf.idf weight of a term is the product of its tf weight and its idf weight by the formula w = log10(1+tf) log10(N/df)).
4. we put the weights in separate list.
5. the Score of each document based on the query will be : the sum of weights.

4 Analyzing System Implementation: Insights and Discoveries

4.1 Commentary

Our IR system consists of four processes: (i) indexing,(ii) query formulation, (iii) matching, and (iv) re-ranking.
Figure 1 shows the flowchart diagram of our IR process. In orde rto respond to a query of the user, IR systems may employ many different retrieval models in the matching process.

4.2 we learnt from it

- gain an understanding of the basic concepts and techniques in information retrieval.
- understand the issues involved in providing an IR service on a web scale, including distributed index construction and user modelling for recommendation engines.

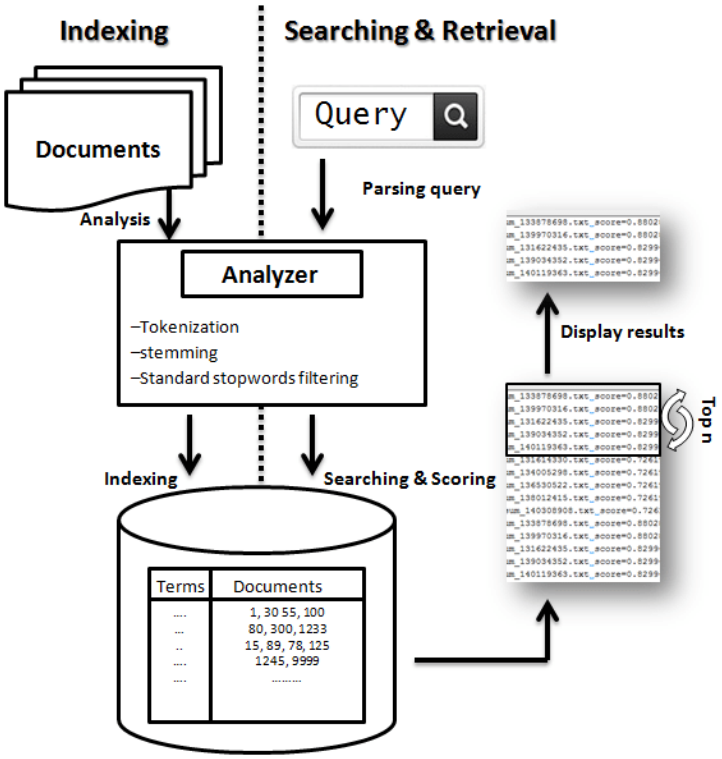

Figure 1: our IR system

5 Challenges

- it was so difficult and hard to find documentations that helps in our work.
- How implementing the inverted index and boolean search.
- challenge of the time , it took us a long time for implement it and organization the code.

6 Suggestions for Enhancing and Scaling Our Implementation

- we will make better use of cocine similarity measure.
- we will add feature to search for image.
- perhaps we will improve this code and apply it to websites with textual searches.