Sessional - 6
Information Retrieval

Abhijit Mustafi

November 3, 2022

In today's sessional we shall try to build a full fledeged IR system which includes almost all the primary components i.e. index creation, scoring and evaluation. We shall use the universally acknowledged Cranfield Corpus for the exercise (Please see the distribution policy of the corpus available at University of Glassgow's site in case you want to use it publicly). The corpus is reasonably small and comes with the following files -

1. The main corpus file
2. A queries file (Contains a set of queries against which we need to fetch documents)
3. A relevance indicator file (A file containing the list of relevant documents for each query)
4. A readme file.

Since the exercise is significantly longer than our other sessionals, we shall break it down into smaller components and solve each component as a separate exercise. Unfortunately beacuse of the paucity of time we shall make one compromise and avoid creating an inverted index instead using a numpy array to store the term frequencies. This is not desirable in a real life scenario and in fact shall not work for any large corpus, so beware of this !! I strongly suggest you redo this exercise using inverted indexes in your spare time (and obviously if IR interests you.)

The components that need to be solved are as follows:

1. Parse the corpus file and extract the 1400 documents. You will observe that each document has some rudimentary metadata associated with it e.g. I. for item no., .T for title, .A for authorname, .B for affiliation of the author and finally .W for the actual document text. Our current interest is in using regular expressions to extract the .W component.
2. Parse all the documents to a list of tokens containing only alphabets, digits, a single dash or a single hyphen. Additionally you can put some length capping. A suggestion is to maintain only those tokens which have a minimum length of 4 characters.

3. Create a vocabulary out of the set of tokens. (We are expecting a vocabulary size of approximately 8300 tokens)

4. Create a term_frequency matrix of size $M * N$, where M is the size of the vocabulary and N is the nos. of documents. The matrix should contain how many time a token occurs in a particular document. Ensure you have this matrix as a numpy array. You can convert a list to a numpy array using *np.array(list_ name).*

5. Create a vector called idf, which contains the inverse document frequency for each token in the vocabulary. Obviously this matrix should be of size $M * 1$. Use numpy's reshape command to get the desired shape if the original shape is obtained as $(M, 1)$.

6. Create a matrix called tf_idf by multiplying term_freq with idf. Usually the idf of a term is given as

$$idf(t) = log\frac{N}{||t||} \tag{1}$$

7. Pick a query from the queries file and parse it using the same parsing tool you used on the corpus.

8. Convert the query into a vector of size $(M, 1)$, where terms in the query are represented by a 1 and all other terms are represented by zeros.

9. Compute the cosine similarity between this query vector all document vectors in the tf_idf matrix (remember each document is a column.). The cosine similarity betweentwo vectors shall be calculated as

$$score(q, d) = \frac{\sum_{i=1}^{M} q_i * d_i}{||d||} \tag{2}$$

10. Sort the scores array and fetch the top "K" scoring documents. Can you now calculate the precision and recall values for the query using the relevant document informationavailable along with the corpus?

We can do a lot of other experiments at this stage including measuring the MAP value and calculating the AUC. Please feel free to experiment.