Information Retrieval
Sessional - 4

A. Mustafi

September 7, 2023

In the exercises for today we shall cover two cases of practical interest in Information retrieval i.e. answering phrase queries and suggesting spell corrections for misspelled queries. We shall use the BBC Sports dataset for both these exercise, however you are free to choose the corpus.

0.1 Answering phrase queries

1. Write a function called *tokenize_text* to tokenize a body of text. The function should accept three parameters i.e. the text to tokenize, the minimum length of the tokens and whether or not the tokens should be returned in lower case.

2. Write a function called *intersect* which should accept to posting lists as parameters and return the common document ids.

3. Write a function called *extended_intersect* which should accept a list of postings and return the doc_ids that feature in all of the posting lists. You should be able to sort the lst of postings based on thier posting lengths and then intersect two lists at a time using the *intersect* function.

4. Write a function called *parse_phrase_query* which should return a list of bigrams of the valid tokens in a phrase query. Remember in case the query has a single token return the token as a list e.g. "very serious condition" should be returned as ["very serious", "serious condition"].

5. Write code to loop through the corpus and create an *inverted index* containing the unigram and bigram tokens in each document.

6. Use the functions you have written to answer a phrase query. Test your code by providing a multi word phrase query and fetch the relevant documents. How long does it take to answer the queries on an average?

0.2 Suggesting spelling corrections

1. Same as in the previous question

2. Write a function *count_k_grams* to count the number of *k_grams* in a token.

3. Write a function *get_k_grams* to return the *k_grams* in a token as a list.

4. Write a function *jaccard_coefficient* to compute the Jaccard coefficient between two sets. Remember the Jaccard coefficient between two sets is defines as

$$dist_{jaccard}(A,B) = \frac{A \cap B}{A \cup B}$$

   Specific to IR, the numerator is the number of k_grams that overlap between two tokens and the denominator is the cardinality of the union of all the k_grams in the two tokens

5. Write a function called *intersect* which should accept to posting lists as parameters and return the common terms. This is the exact same function as in the previous exercise, however here we shall use it to intersect common k_grams rather than doc_ids.

6. Loop through the corpus creating two data structures (you can use dictionaries). The first called *vocabulary* should contain the list of unique tokens and the number of k_grams associated with the token. The second called *k_gram_index* containing all identified k_grams in the corpus with each k_gram associated with the tokens that contain it.

7. Create a dictionary called *count_overlap*. Accept a unigram query from the user and parse it into its k_grams. Traverse the k_gram postings in the k_gram_index and each time you find a matching token increment the count for the corresponding token in the *count_overlap* dictionary.

8. Which are the nearest matches for the query going strictly by the count overlap value?

9. Are the results any different when you compute the Jaccard Coefficient between the query term and each overlapping term and present the best matches?