Report

Name: Rong Zhen
zID: z5225226

Step1: TF-IDF index construction for Entities and Tokens

In this step, all the implementation of TF-IDF index construction are in the **index_documents** function.
I initialized the **self.tf_tokens/self.tf_entities** as a dictionary whose value is also a dictionary. The outer keys are tokens/entities, the inner keys are documents id. But the inner values for **self.tf_tokens** are normalized frequencies of the tokens and the inner values for **self.tf_entities** are numbers of entities.
The first thing is to parse the documents and get my tokens and entities. Set a list to store tokens. The single entities are also in tokens, which means I need to delete them from tokens and I couldn't use tokens directly. But there is no need to store entities individually. So I stored entities using **self.tf_entities** and updated the count of entities. Once the entity is a single word, I store it in a list called **single_entity** and remove them from the tokens list. Then I used the **Counter** function to calculate the count of tokens. After doing these things, I stored the tokens in **self.tf_tokens** and updated the values with normalized frequencies of the tokens.

[*Note:* The normalized frequencies of the entities can be calculated in the **max_score_query** function. Because we need to use the count of entities to calculate normalized frequencies. So it is not way to calculate it directly and it need another for-loop and dictionary if I separately stored the count and the frequencies. It increases the time complexity. So I put the calculation in step3 which already has a for-loop, and I can calculate it easily.]

Step2: Split the query into lists of Entities and Tokens

In this step, all the implementation for splitting the query are in the **split_query** function.
I firstly used **space split** to get the token list of query. Then I used **Regular expression** to delete the entities in DoE which have tokens do not in Q. And then I used **itertools.combinations** to get the possible subset of entities. I split each possible subset and remove the tokens of subset from token list. Once the tokens of subset cannot be found in token list, this subset is not needed subset. I returned a list called **splits** in **split_query** function with different possible splits of the query to the lists of the filtered entities and the tokens.

Step3: Query Score Computation

In this step, all the implementation for calculating the query score are in the

max_score_query function.
The first argument recalled the **splits list** in **split_query** function and the second argument is the **document id**. I traversed the splits list, got the possible splits and calculated their TF-IDF scores using **self.tf_tokens** and **self.tf_entities**. [*Note:* the normalized frequencies of entities need to calculate here]. Calculate the final scores using **scores of entities + 0.4 * scores of tokens**, and store all the scores in a list. Find the maximum one and the corresponding tokens and entities.
Finally, format what we find and return it.