Logo, company name

Description automatically generated

**FINAL PROJECT REPORT**

**SEARCH RETRIEVAL SYSTEM**

**NAWAAZ SHARIF**

**SYED NOOR RAZI ALI**

**MOHAMMED RASHIDUDDIN**

**CSC 575: INTELLIGENT INFORMATION RETRIEVAL**

**PROFESSOR BAMSHAD MOBASHER**

**Table of Contents:**

|  |  |  |
| --- | --- | --- |
| **Index** | **Content** | **Page No:** |
| 1. | Introduction | 3 |
| 2. | Dataset | 3 |
| 3. | Function and Implementation | 4 |
| 4. | GUI Application | 10 |
| 5. | Evaluation | 13 |
| 6. | Appendices | 14 |

**Introduction:**

Many clinical trials require patients for the studies that are to be conducted for the trials in the medical research, but due to the high rate of insufficient patients recruitment, there's a possibility that the study or trial may need to be terminated permanently.

The TREC clinical trials track has concluded to perform the trials in a way such that a patient who has similar disease or who has the similar symptoms related to a particular study, all their medical history and records are needed to be updated on the clinicaltrials.gov website so that the different doctor/researcher can refer to the previous trial that has already been performed. The data set is broken down into eligible and non-relevant queries to distinguish between patients who have sufficient information from people who have insufficient information about the disease or trial being conducted.

The structured data is helpful for clinical trials as the description of the patient that is mentioned in the data set is only limited to his or her disease topics. The TREC clinical trials provides a platform for doctors or researchers a platform, for evaluating patients matching symptoms for clinical trial requirements, for example if a researcher comes all the way to the middle of the research and then something bad happens to the patient and then after some time a person researcher gets a similar patient from the point where the previous research was held he can continue from there instead of starting the whole research from the beginning which can be more effective and save time in the field of medical research.

**About Dataset:**

In the dataset of 2021 Clinical Trials Track, we have data of several different patient’s cases that are created by people who have received medical training.

The information of patients is recorded in an .xml file, where we have:

1. Task: which is “2021 TREC Clinical Trials”

2. Number: patient case number recorded.

3. Description: in the form of corpus, where we have detailed description of patient, like how old he/she is, their gender, which department dealt with their case, and what disease or symptoms they had, and the detailed medical report of what all things have been identified by the practitioner.

**Functions and Implementation:**

Since we have five zip files we are combining into a single zip:

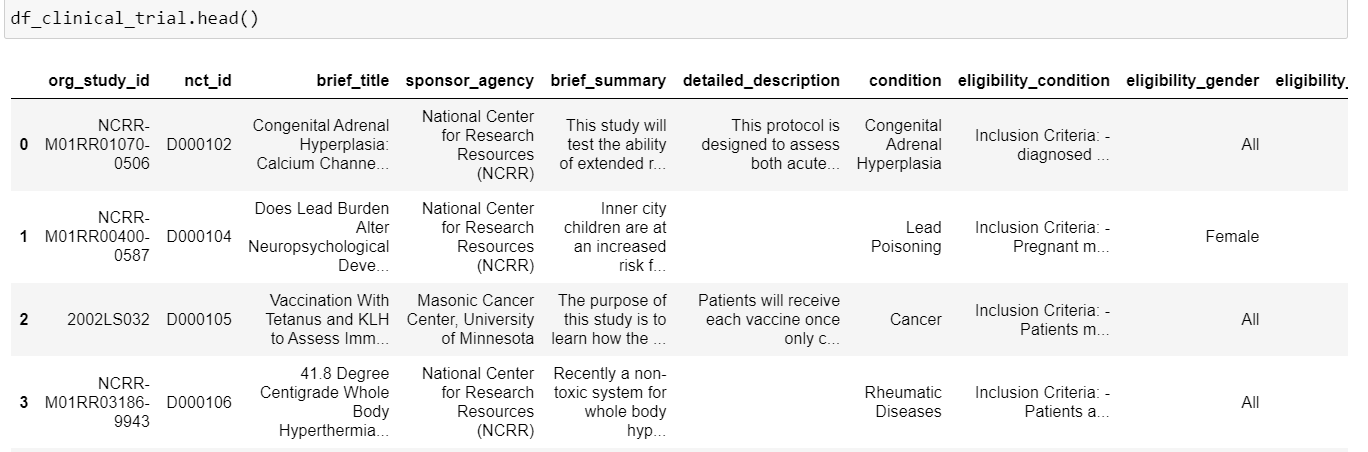
1. ClinicalTrials.2021-04-27.part1.zip
2. ClinicalTrials.2021-04-27.part2.zip
3. ClinicalTrials.2021-04-27.part3.zip
4. ClinicalTrials.2021-04-27.part4.zip
5. ClinicalTrials.2021-04-27.part5.zip

This will give us Final\_combined\_ClinicalTrials.zip.

Each of the zip file contains 500+ .xml files. Looping through each xml and extracting only useful tags and converting it into a Data Frame.

Important Columns in the clinical trials dataset:

['org\_study\_id', 'nct\_id', 'brief\_title', 'sponsor\_agency', 'brief\_summary' ,'detailed\_description' , 'condition' , 'eligibility\_condition' , 'eligibility\_gender' , 'eligibility\_minage' , 'eligibility\_maxage']



1. Corpus

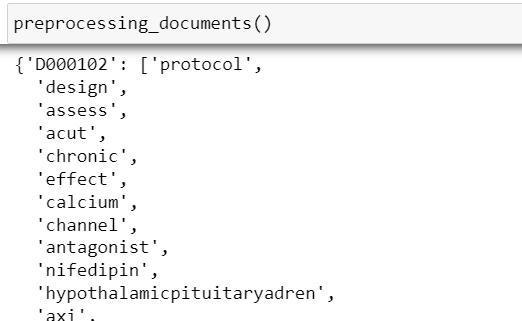
Converting of pd.DataFrame to a dictionary where ‘nct\_id’ is our index, ‘brief\_summary', 'detailed\_description' , 'condition' , 'eligibility\_condition' and ‘brief\_title’ and these columns are combined to form our document data for each document ID.

2. Queries

Fetching the data from the URL 'https://www.trec-cds.org/topics2021.xml' and converting it into pd.DataFrame for queries.

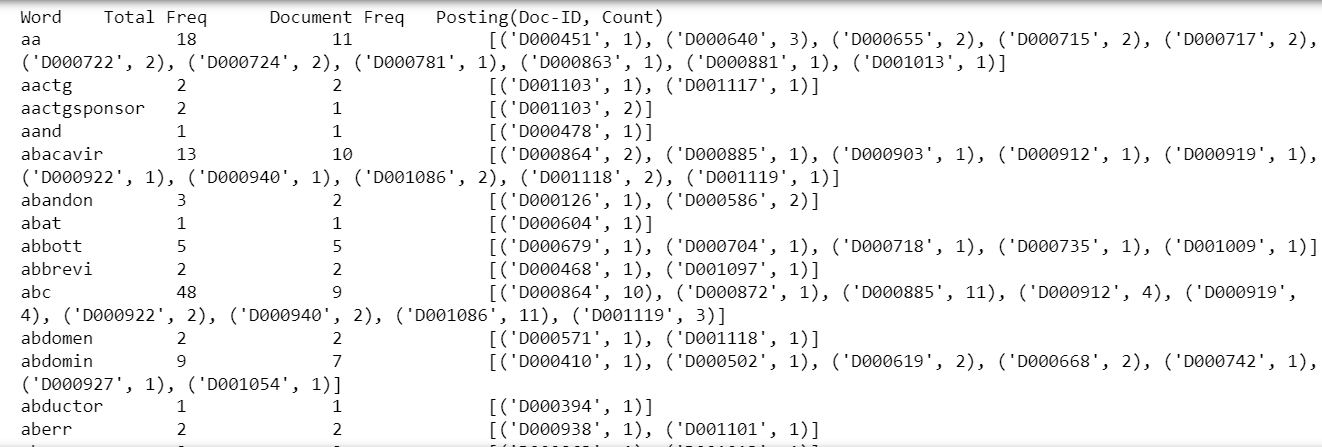
3. Pre-Processing of Corpus:

First, we have removed punctuations, then we converted our string to lower case and then split it into a list, and used a nltk package to fetch the stop words and performing stemming. This removed the stop words and do the stemming for the whole corpus dataset.



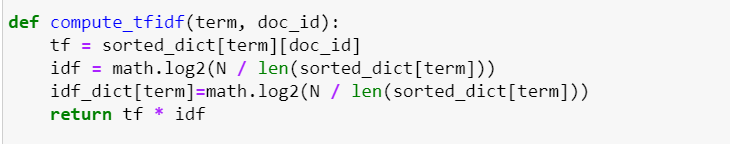
Counting the occurrences of each word in the document with document id and frequency, we get the output as below.

**Output:**

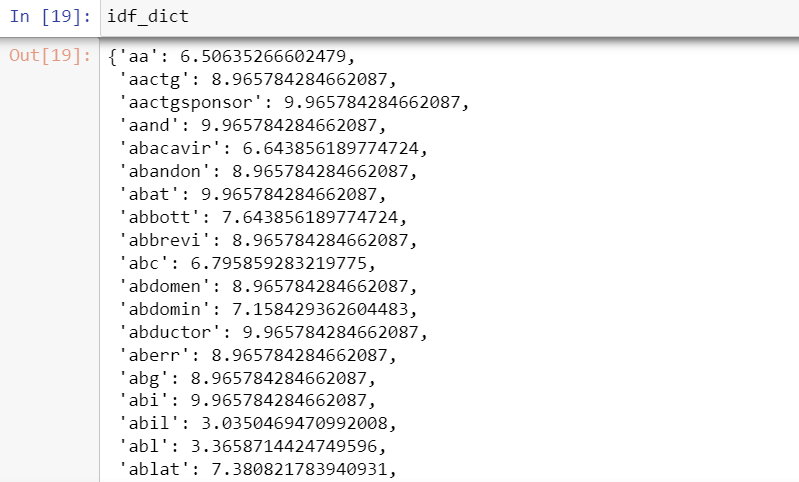


**4. Calculating IDF**

Writing a function to compute the IDF values of the terms present in the documents, we are creating a idf\_dict which holds the IDF values for all the words.

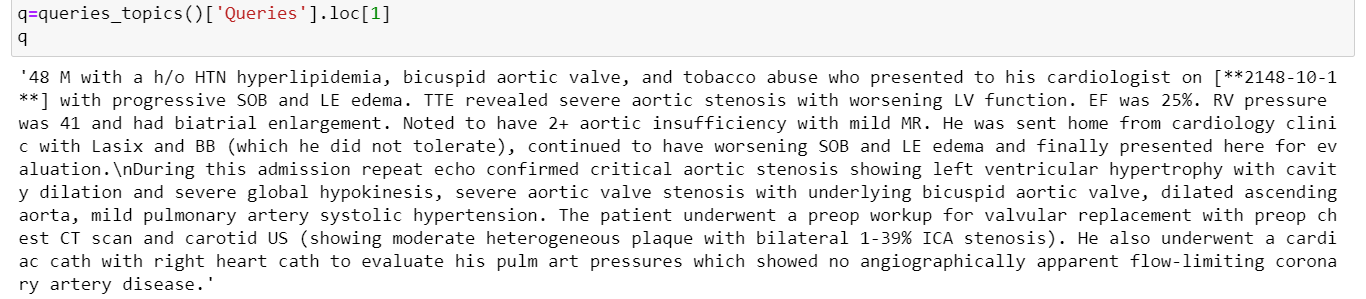


**Output:**



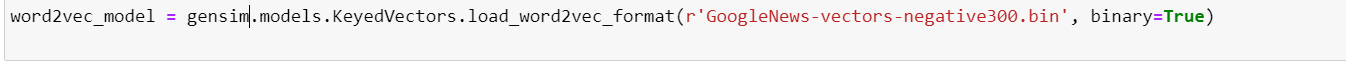
5. Synonyms for Query Expansion and Vector Representation of Query.

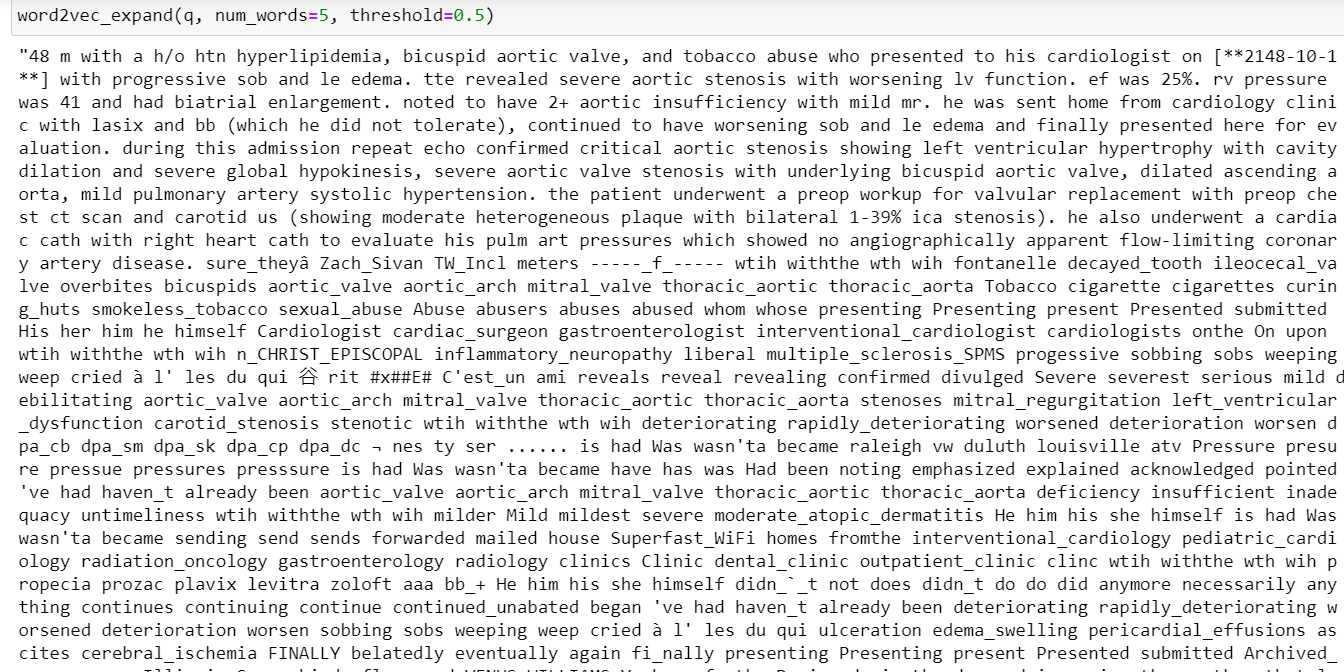
**Our Original Query**



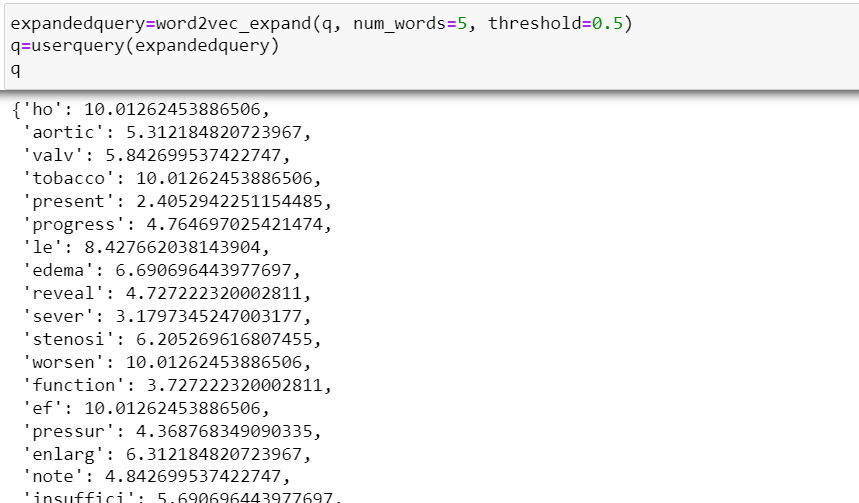
We have used this word2vec model and used to find different words/synonyms for our query expansion.

**After synonym expansion**





After expanding the query, I have used the same pre-processing steps for queries as we have used for corpus main dataset by removing punctuations, stop words and stemming of each query.



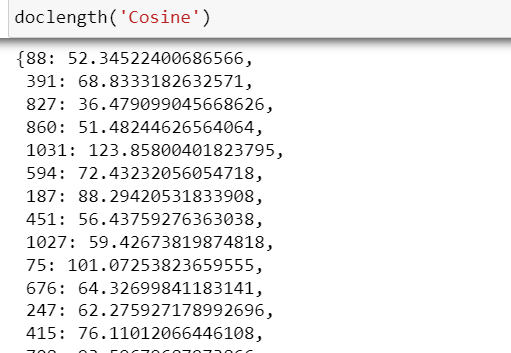
We get the doclength, queries length, which helps us in getting the inverted\_index values.

To calculate the document length, we are taking the sum of the squares of document weights and multiplying them with their respective IDF values of their term. To calculate the cosine similarity, we are taking the square root of the tf\_idf, and similarly to calculate dice and jaccard similarity, we take the sum of the tf\_idf values.

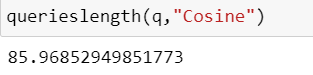
To calculate the queries length, we take the sum of square of their respective weights.

To store the documents with their respective similarity score we create a hashmap.

To get the similar documents for the query we create a new hash map with zero scores and using the respective similarity we find the score for each document which is then run iteratively to find the term weights for the whole query and document ID.

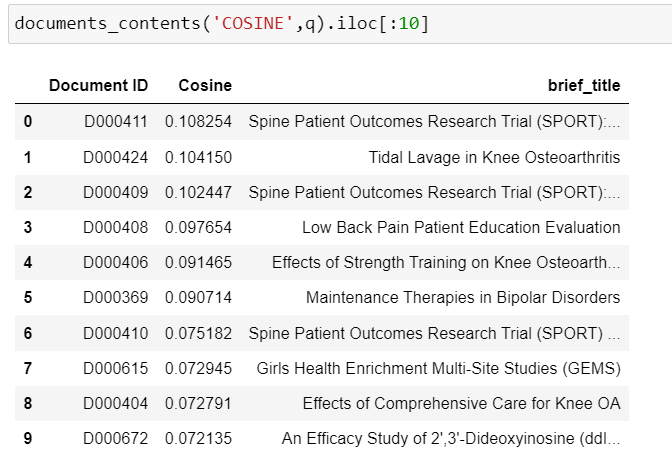


**Queries Length**

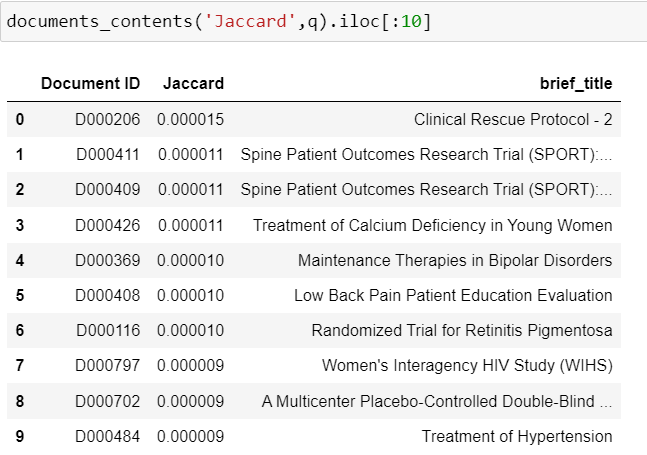


This gives us the output”

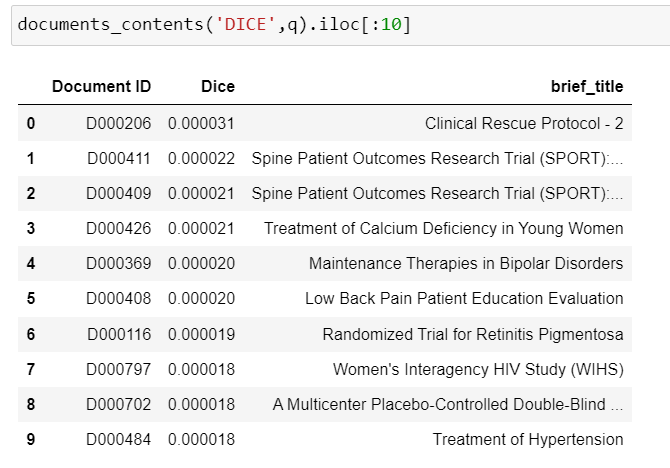
**Cosine similarity values for our documents:**



**Jaccard similarity values for our documents:**



**Dice similarity values for our documents:**



From our original corpus data, we have selected the title information and the similarity to show with respective to the document id.

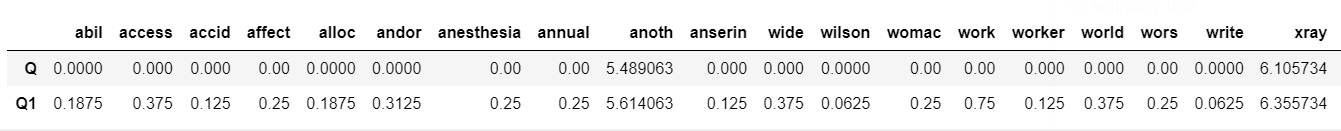
**Query Expansion and Relevance Feedback:**

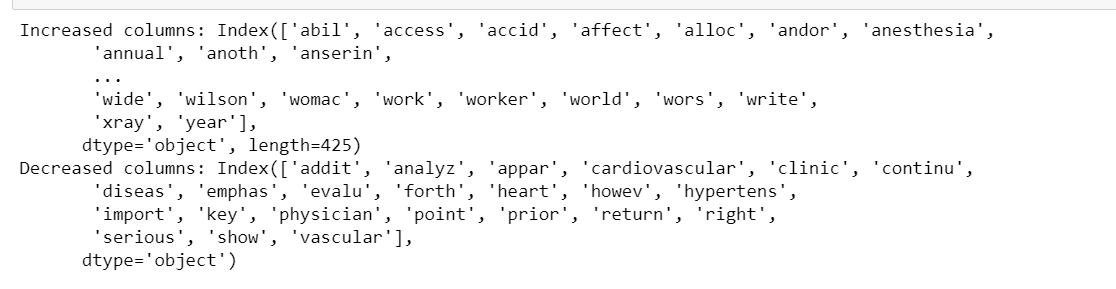
Sometimes when a user enters a query, the terms may be not in his dictionary, so this is where rocchhio relevance feedback comes which ask the user about relevant and non-relevant documents and gives the new query.

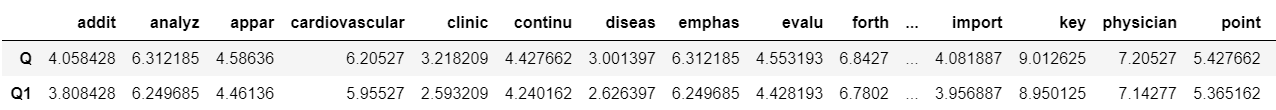
This new query may have few added terms and may have few terms disregarded when compared to the original query.

By default, we have used alpha =0.5 and beta=0.25 to calculate the modified query.

**Output**

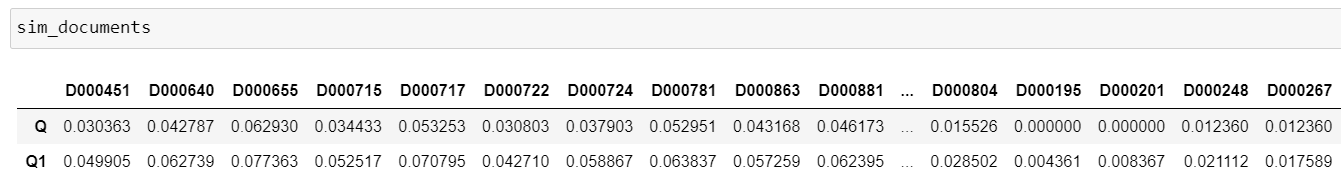






Similarly, we have also found the cosine similarity of the old and new query with the documents.

**Output:**



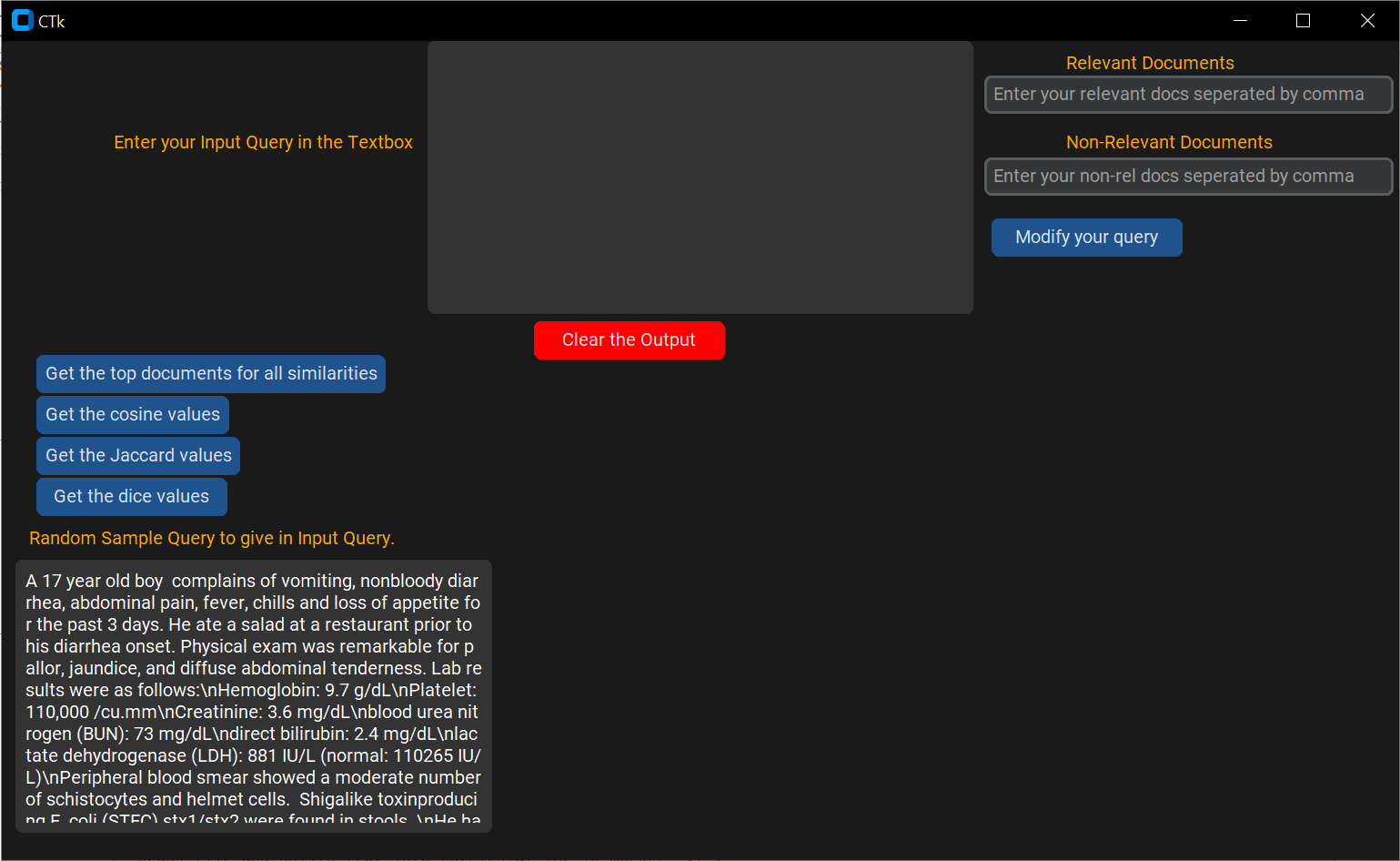
GUI APPLICATION:

We have created an interactive implementation of our search retrieval system using Tkinter module, where we enter the query and it gives us the combined results of the cosine similarity, jaccard similarity and dice similarity.

We also ask the user about the relevant and non-relevant documents and shows the suggested vocabulary terms which can be helpful in retrieving the desired sets of documents.

Since the queries are large, we have also created a random generator of query which is given to the user to select and see the result.

**Output:**



Entering the query:

Getting all the similarity scores, with respect to the top 10 similar documents.

Text

Description automatically generated

Displaying the cosine similarity scores:

Text

Description automatically generated

Displaying the jaccard similarity scores:

Text

Description automatically generated

Displaying the dice similarity scores:

Text

Description automatically generated

Giving the relevant and non-relevant documents it gives us the words that we can use in our vocabulary.

Text

Description automatically generated

**Evaluation:**

The dataset used to evaluate our search/retrieval system is Medline dataset from the University of Glasgow.

The dataset can be obtained on <http://ir.dcs.gla.ac.uk/resources/test_collections/medl/>

We have performed same pre-processing steps, term-weighting and inverted index as we have performed above and we got an precision of 0.6 and recall of 0.25.

From the above precision and recall values, we can say that our search/retrieval system works fairly well.

**Appendices:**

Here is the Jupyter Notebook and the HTML file for the search engine retrieval.

YouTube Link:

<https://youtu.be/Kz1P5lzgqic>