IR ASSIGNMENT 2 REPORT

Ques.1

a) TFIDF:
Import all the necessary libraries for the preprocessing process.
We have the raw files with us. We will be doing the preprocessing process on these files and get the filtered data.
Then we will make a list of all the words present in the file and then add this list to another empty list.
Then we take the input from the user and then do the preprocessing process on the input.
Then many functions are made to calculate the Term Frequency and Inverse Document Frequency.
We find the TF for weighting schemes such as Binary, raw count, Term Frequency, Log normalization, Double normalization. Then we made separate functions for each weighting scheme to calculate the TF-IDF values.
After getting all the TF-IDF values for all the weighting schemes we then calculate the TF-IDF score between the contents of the file and the input given by the user and store it in a dictionary.
We then sort the dictionary in decreasing order of the TF-IDF score and display the Top 5 relevant files for all the weighting schemes.

b) Jaccard Coefficient:
We have the preprocessed data which we created from the previous parts.
Then we took an input from the user and did the tokenization process on it and we also did the tokenization process on the content of the file in order to calculate the Jaccard coefficient.
Jaccard coefficient is Intersection/Union. So we wrote the code for Intersection and Union in different functions.
Then we calculated the Jaccard coefficient between the query and all the contents of the file and stored the values and file name in the dictionary.
Then we sorted the dictionary in decreasing value of the Jaccard coefficient and then at last print the Top 10 relevant files.

Ques 3.

The code imports the Pandas, Matplotlib, Numpy, and Math libraries using the import statement. These libraries provide functions for data manipulation, graphing, numerical computing, and mathematical operations, respectively.

1. The code reads in a text file named 'IR-assignment-2-data.txt' using the pd.read_csv function from the Pandas library. It sets the delimiter to a space and sets the header to None, indicating that the data does not contain a header row. The resulting data is stored in a Pandas DataFrame called data.
2. The code creates an empty dictionary called database_dict and populates it with the values from the 'qid:4' column of data. The keys of the dictionary are the row indices where the 'qid:4' value appears in the '0' column of data.
3. The code creates a new DataFrame called temp that only contains the rows in data where the key is in database_dict. It then saves this new DataFrame to a text file called 'query4max.txt' using the np.savetxt function from the Numpy library.
4. The code creates a list called unsortedDb containing the items from database_dict in the form of (key, value) tuples. It then sorts database_dict in descending order based on the value of each item using the sorted function with a lambda function as the key argument. The sorted database_dict is then reassigned to the original variable name.
5. The code defines two helper functions, check and calc, that are used in the findTotalFiles function. check takes a dictionary and counts the number of occurrences of the values 0, 1, 2, and 3. calc takes a list of counts and returns the factorial of each count multiplied together. findTotalFiles calls check and calc to calculate the total number of possible files that could be retrieved from the database_dict.
6. The code defines a function called run_findDCG that calculates the Discounted Cumulative Gain (DCG) for a given set of data and length. It uses a formula that calculates the sum of the relevance scores for each item, with a logarithmic discount factor applied to the position of each item in the list. The findDCG function calls run_findDCG to calculate the DCG for database_dict and unsortedDb, and then calculates the normalized DCG (nDCG) for each of these sets of data.
7. The code initializes some variables and defines two functions, plot_me and ff, that are used in the getPrecisionValueAndRecallValue function. plot_me takes two lists and plots them on an x-y graph using the plt.plot function from the Matplotlib library. ff takes a sorted list of (key, value) pairs and calculates the precision and recall values for each item. It then appends these values to the precision val and recall val lists, respectively.
8. The getPrecisionValueAndRecallValue function takes a dictionary of (key, value) pairs as input and sorts the pairs by value in descending order. It then calls ff to calculate the precision and recall values for each item and plots these values using plot_me.