IR Assignment-2

Loading and Preprocessing the dataset

- For this, first of all, I made a git repository of the dataset and cloned it in the jupyter notebook that I am using.
- Using pandas, I imported the data frame from the CSV file given to us as the dataset(A2_data.csv), with a total of 1000 rows.
- I edited the dataset and placed the header value, Review Id for one column; it was not there and dropped some rows with empty values(dropna()) in place. Now, in total, 999 non null rows
- Then, the list of images column, into another dataframe, with only images individual and repeated dataset. To understand better, I have saved it as a csv file; Please view it. Therefore we get 1647 rows with repeated information and unique image links column.

Image Extraction

- Then, the image extraction using the InceptionV3 engine, preprocessing the images, applying all the preprocessing steps and using preprocess input from the function to preprocess input according to the requirement for the dataset.
- For each row, after extracting the image features, we normalized the whole dataset with the mean and standard deviation of each respective image feature.
- Please note that the normalization is with respect to one image.
- Saving the extracted normalized image features using a pickle module
- I had used the ls thing to save those indexes that have erroneous image links,(not able to open them) in a bit vector “ls” , so I decided to remove those erroneous review id from the normal data frame - 999 rows and expanded data frame for images - 1647 rows using the same bit vector ls- as through “ls” I found the Ids that had an error- then using those ids, I removed them from the 999 rows dataframe.
- Those erroneous review ids are- They are repeated as they are processed from 1647 rows datafarme, in total they are 6 review Ids whose image link is not working

```
Errorness Review Ids
2235.0
2235.0
3317.0
3317.0
2912.0
2265.0
2088.0
3474.0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1639 entries, 0 to 1638
Data columns (total 3 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Review Id    1639 non-null   float64
 1   Image        1639 non-null   object
 2   Review Text  1639 non-null   object
dtypes: float64(1), object(2)
memory usage: 38.5+ KB
```

- Why I am removing now? - Because I doin’t want their text reviews to be processed into the tf-idf matrix

- Then, loading the pickle file and putting the normalized image features into the dataframe in the list wise order as it is in the image column and then saving the file using pickle module as the final file

Text Extraction -

Calculate Term Frequency (TF):

- The calculate_tf function computes the term frequency for each word in a given text. It iterates through each word in the text, counts its frequency, and divides it by the total number of words in the text. This ensures that the TF value is normalized. The result is stored in a dictionary.
- The TF values are then applied to each document in a DataFrame df under the column 'Cleaned_Review', and the resulting TF dictionaries are stored in a new column 'TF'.

Calculate Inverse Document Frequency (IDF):

- The calculate_idf function computes the inverse document frequency for each word in a collection of documents. It iterates through each document, counts the number of documents containing each word, and calculates the IDF value using the formula: IDF = log(N / (df + 1)), where N is the total number of documents and df is the document frequency of the word. The IDF values are stored in a dictionary.

Combine TF and IDF to calculate TF-IDF:

- The calculate_tfidf function combines the TF and IDF scores to calculate the TF-IDF score for each word. It multiplies the TF score of each word by its corresponding IDF score and stores the result in a dictionary.
- This function is applied to each row of the DataFrame df to calculate TF-IDF scores for each document. The TF-IDF scores are stored in a new column 'TF-IDF'.

Save IDF Dictionary and TF-IDF Scores:

- The IDF dictionary is saved using pickle into a file named 'idf_dict.pkl' and the TF-IDF scores for each document are saved using pickle into a file named 'tfidf_scores.pkl'.

- Everything is placed in a column of the data frame and saved and then for actual programming it is called and used the current datafarme will have:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 993 entries, 0 to 992
Data columns (total 7 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Review Id                 993 non-null    float64
 1   Image                     993 non-null    object
 2   Review Text               993 non-null    object
 3   Normalized Image Features  993 non-null    object
 4   Cleaned Review            993 non-null    object
 5   TF                        993 non-null    object
 6   TF-IDF                    993 non-null    object
dtypes: float64(1), object(6)
memory usage: 54.4+ KB
```

- Using the pickle and read_csv module, we import tfidf_scores for the dataset, idf_dict, normalized_image_features, df and new_df

Taking the Input and Flattening the array

- Now, we take the input (text andinput image url) and preprocess it accordingly and extract the normalised input image features.
- Now we are going to flatten the normalized image features column, where each row is a 3d array of shapes (5,5,2048), into a 1d array, similarly we will flatten the input_normalized_image_features as well but we have issues with the noramlized image features in df as it is a list fo 3d arrays - it is not flattened hence - we start to flatten it as well, using the image[index].count - we create a list and append the features accordingly since it is already in the order
- Now, using the idf dictionary saved - we start to produce the tf-idf vector for the input review text after preprocessing it.

Cosine Similarity

- We ask the user, about how many ranks are required in the form of input “n”
- The code loads precomputed TF-IDF vectors from a pickle file, which represents the textual features of the reviews.
- For each review, the cosine similarity between its TF-IDF vector and the input TF-IDF vector (representing the input text query) is calculated using the cosine_similarity_text function. These similarities are stored in the DataFrame under the column 'Cosine Similarity Text
- Similarly, for each review, the cosine similarity between its image features and the input normalized image features is calculated using the cosine_similarity_image function. The average cosine similarity across all images associated with a review is calculated and stored in the DataFrame under the column 'Cosine Similarity Image'.
- The composite similarity score for each review is computed as the average of the text-based cosine similarity and the image-based cosine similarity. This composite score is stored in the DataFrame under the column 'Composite Similarity'.
- The top 'n' reviews are selected based on their text cosine similarity scores, image cosine similarity scores, and composite similarity scores using the “nlargest” function. The indices of these top-ranked reviews are stored.

Since the upload limit for each file should be less than 100MB, I saved the file in the google drive: The link is here

Link for Pickle File for Image Features