Steps Involved in Code Execution

Set Up:

Programming language Used: Python.

How to run the code: Any machine with python installation can be used to run this code.

Providing. ipynb extension file. Should be able to run using anaconda console. File is self-exploratory with all the descriptions mentioned. Required python libraries imported or installed while executing the note book itself.

Input Requirements:

Assumptions

- Ignore Image/Video search results.
- Choose any two categories for the same purpose and choose any random date for extracting news for this task.

Approach

- Categories as Sports and Science with search query **cricket** for **Sports. Vaccine** for **Science.**
- Chosen date as 2021-01-01 for sports.
- Chosen date as 2021-01-04 for sports.
- Search sites taken are Google news and Bing news.

Function Descriptions:

Task 1 News Extraction from a search engines

- **Functions Description and usage:**
    - **Getdatefilter()** to filter the date time as per new feed format, news feed date time is for example **"Fri, 01 Jan 2021"**
    - **ExtactNews(datefilter,category,query,searchsite,url)**

        - datefilter is to filter the extracted results based on date.
        - category ex: Sports/Science for which trying to extract the news.
        - query: Search query to perform search under given category.
        - searchsite : Target search engine name as we are storing specific search results into dedicated folder w.r,t search engine ex googlenews or bingnews.
        - This method will perform the search based on date, query against category for given search engine url and save the results under **Data**/{**category**}/{**searchsite** }news_{**category**}.csv file.
        - url: If we target search engine is google then pass google rss feed base url. Query and category are also parameter for this method, final target url will be constructed dynamically.
        - Same generic method is used for both google and bing search engines.

Task 2 Aggregated News Collection

- **Functions Description and usage:**
  - **Aggregatenews(category,searchsites,encodingtype):**Takes input parameter Category can be Sports or Science , searchsite will be google and bing (list) and encoding type. Based on the parameter given will check the unique documents based on title in both collections and removes the duplicates.
  - Remove the duplicates based on the unique titles in both the document collection.
  - Considers the documents from google news if it finds any duplicates between extracted news collection.
  - As an output merges both the collections and provide aggregated output collection as a pandas dataframe.
  - By calling above method results will be stored in uniquenews.cvs under Data/Sports/ uniquenews.csv

Task 3 Ranking

- **Functions Description and usage for Approach 1:**
  - **Rankdocument(df):** Take the dataframe add 2 columns to the dataframe and returns the same dataframe.
  - Adding Rank column based on latest news gets the highest ranking from starting from 1 max to total now of rows in the dataframe.
  - Adding DoucmentId column to name each document using naming convention as Doc 1
  - DoucmentId column is combination of "DOC" string and Ranking.
  - **StoreApproach1Results(df, category): Take the dataframe and category as inputs and store the results in /Data/{category}/Rank/ResultantRanks_A1.txt.**

- **Functions Description and usage for Approach 2:**
  - **ToLowerCase(df):** Convert the title and document text for give dataframe and returns the same dataframe.
  - **expand_contractions(text,contractions_dict=contractions_dict) :** Does the expanding of Contractions for given text .
  - **clean_text(text)** : Takes the text as input and moves characters like new line, http and others and returns the clean text.
  - **Lemmatization(df)** : Takes the dataframe as input and perform the lemmatization and stop word removal on Title and Description columns of document collection.
  - **tfidfweighting(news_list):** Takes the list of documents on which need to construct the vocabulary. Returns the vocabulary indexed dataframe.
  - **get_similar_documents(q, df_vocbIndex,category,df,news_list)**: Perform the cosine similarity calculation based on given query q , vocabulary index data frame, category and document collection data frame and processed news_list.
  - Store the ranked documents based on cosine similarity between the query and document collection. Results get stored in Data/{Category}/Rank/ResultantRanks_A2.txt file.

- Approach 2 document is ready now.

Task 4 Evaluation

- **Functions Description and usage:**
  - **Findprecision(number, targetfile, category, encodingtype)**: This method will take a number, target file, category and encoding type as as input parameters.
  - Category can be Science or Sports. Enocdingtype is to read the file as now code uses **'cp1252' and 'utf-8'.**
  - Calculate the precision (which Is float datatype) against ground truth file and targetfile which is passed. For this assignment it can be ResultantRanks_A1.txt/ ResultantRanks_A2.txt which got generated as part of Ranking task for approach 1 and approach 2 in task 3.
  - Can call precisionat5=Findprecision(5,'ResultantRanks_A1.txt',categories[1],'utf-8'), This means calculate the precision for top 5 documents from RankedDocuments.txt (ground truth file) and ResultantRanks_A1.txt'. This will return example output will look like 0.5 etc..
  - **Precisionplotting(data) :** This method is used to plot the obtained results
  - X-axis needs to have labels such as P@5,P@10, P@15,P@20, P@25,P@30. Y-axis needs to have values obtained for P@5,P@10, P@15,P@20, P@25,P@30. i.e. will vary between 0 & 1
  - It takes the **data** dictionary as input which contains the keys and values list. Keys are used as labels that is used as x-axis variables. Values contains the precision values against each label like precision@5, precision@10 etc.