task1_31187366

September 16, 2020

1 FIT5196 Assessment 1

Student Name: PRASHANT JAJORIA

Student ID: 31187366 Date: 13/09/2020

Environment: Python 3.7.4 and Anaconda 4.8.4 (64-bit)

Libraries used: * langid - for finding English tweets, included in Anaconda Python 3.7.4 * re - for regular expression, included in Anaconda Python 3.7.4 * os - for reading and writing file, included in Anaconda Python 3.7.4

1.1 1. Introduction

The main goal of this assignment is to extract data from semi-structured text files, which forms the basis of Text analysis. A total of 2421 `.txt` files have been provided, containing data about COVID-19 related tweets. The main objective in this task is to get the information about the tweets into a structured `XML` format.

Following are the requirement of the task: 1. Extract unique tweet ID with their corresponding text and the date on which the tweet was created. 2. Only English tweets should be included in the `XML` file. 3. The data should be stored in provided `XML` structure.

A step by step explanation for of completing the requirements will be explained in the following code cells.

1.2 2. Import libraries

[689]:
```
import re
import os
import langid as lg
from os import listdir
from os.path import isfile, join, dirname
```

1.3 3. Get the name of all input .txt files

To start analyzing the tweets we need to get the names of all the `.txt` files. For this this I define a function named `get_name_input_files()`. This function returns a list of name of all the `.txt` files containing the data about the tweets. It gets all `.txt` filename‘ stored in the directory named **part1** relative to the current directory.

The `os.path.dirname(__file__)` function gets the directory name where the python script is running. Then we add the directory **part1**, as all the inuput `.txt` files are stored in that directory. Using `listdir()` passing the directory name where the `.txt` files are stored, we get the list of files. Using list comprehension here `[f for f in listdir(dir_name) if isfile(join(dir_name, f)) and f.endswith('.txt')]` to make a list of names of all the `.txt` files in **part1** directory.

[690]:
```
def get_name_input_files():
    '''
        Return type : List of names of txt files
    '''

    # variable to store the directory name
    dir_name = ""

    # variable to store the path of txt files
    list_txt_files = []

    # as all the input .txt files are in 'part1' directory, append 'part1' at␣
↪the end
    dir_name = os.path.dirname(os.path.realpath('__file__')) + "/part1/"

    # List of txt file names in the 'part1' directory
    list_txt_files = [f for f in listdir(dir_name) if isfile(join(dir_name, f))␣
↪and f.endswith('.txt')]

    return list_txt_files
```

1.4 4. Read all the .txt files

Defining a function named `read_txt_files()` with parameter being a list of names of txt files. Open each file which is in the list using the `open()` function and read the lines of `.txt` file using `readlines()` functions. Store all the read lines in a dictionary, with key being the filename and value is the lines read from that file. Finally return the dictionary which has the file content.

[691]:
```
def read_txt_files(list_txt_files):
    '''
        Parameter : List of txt file names
        Return type : Dictionary containing the content of txt files
    '''

    # defining variable to store the content of txt files.
    dict_file_content = {}

    # variable to store the directory path of txt files
    # append 'path1' as all the input .txt files are in 'part1' directory
    dir_name= os.path.dirname(os.path.realpath('__file__')) + "/part1/"
```

```
    # loop through all the filenames in the list
    for file_name in list_txt_files:

        # open each file in 'read' mode using 'UTF-8' encoding
        with open(dir_name + file_name, 'r', encoding = "utf-8") as f:

            #read the content of each file and store as string in a dictionary
            dict_file_content[file_name] = ""

            # read all the lines
            for line in f.readlines():

                # key of dictionary is the filename
                dict_file_content[file_name] = dict_file_content[file_name] +␣
↪line

    return dict_file_content
```

1.5 5. Driver function to get the results

Defining function named `get_results()` which acts as the starting point of all the process of data extraction. It firstly calls the `get_name_input_files()` which returns a list of `.txt` filenames. Once we have the name of `.txt` files, it calls the `read_txt_files()` function which returns a dictionary with key being the filename and value being the text of the file. Then the `extract_data()` function is called passing the **filename** and **content** of each file.

[692]:
```
def get_results():
    '''
    Driver function to carry out all the oprations of getting the final results
    '''

    # get the names of txt files
    list_txt_files = get_name_input_files()

    #read the content of each file and store as string in a dictionary
    dict_file_content = read_txt_files(list_txt_files)

    # to store of count of files processed
    count = 1

    # for each file extract the data
    for each_file in list_txt_files:

        # get the file content corresponding to the file name
        file_content = dict_file_content[each_file]

        # MAGIC FUNCTION
```

```
        # pass this file data to start analysing the tweets
        extract_data(each_file,file_content)

        # print number of files processed
        print( str( round(count/len( list_txt_files ),4)*100 ) + " % Files␣
↪processed" )

        count = count + 1

    tweets_to_xml(day_tweets)
```

1.6 6. Defining helper functions

1.6.1 get_tweet_id()

Defining function named `get_tweet_id()` to extract the **ID** from the given tweet.

The regular expression used is **(1ṇ)**. - () indicates a capturing group. - ṇindicates any digit from [0-9] - 1ṇ matches 19 digits which occur together.

[693]:
```
def get_tweet_id(tweet_data):
    '''
    Function to get the tweet Id
    Parameter : String
    Return type : List
    '''

    return re.findall(r'(\d{19})',tweet_data)
```

1.6.2 get_tweet_date()

Defining function named `get_tweet_date()` to extract the **DATE** from the given tweet.

The regular expression used is **(4̣-2̣-2̣T2̣:2̣:2̣.3̣Z)**. - () indicates a capturing group. - ṇindicates any digit from [0-9] - (4) indicates 4 digits occuring together. Similarly 2̣ indicates two digits together.

As the date is in the format YYYY-MM-DDTHH:MM:SS.SSSZ, the regex capture the date, month and year seperated by hyphens(-). Similarly the time is seperated by colon(:). And the **T** is used to denote the start of time and **Z** denote the end of date time.

[694]:
```
def get_tweet_date(tweet_data):
    '''
    Function to get the Tweet date
    Parameter : String
    Return type : List
    '''

    return re.findall(r'(\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}.
↪\d{3}Z)',tweet_data)
```

1.6.3 get_tweet_text()

Defining function named get_tweet_text() to extract the TEXT from the given tweet.

The text appears in different format in the given `.txt` files. So a single Regular expression cannot capture the text for all the files. We need to conditionally use different Regular expressions to capture the Text for all the files.

All the below Regular expresions uses lazy binding i.e. `.*?` to capture the text - `"text":"(.*?)"` - In this re the Tweet text starts with "text" and ends with ". - `"(.*?)","created_at` - In this re the Tweet text starts" and ends with ","created_at - `"text":"(.*?)","id"` - In this re the Tweet text starts "text":" and ends with ","id" - `"text":([^\\].*)` - In this re the Tweet text starts "text":. Also here text can have because it is escaped in the character set `[^\\]`

[695]:
```
def get_tweet_text(tweet_data):
    '''
    Function to get the tweet text
    Parameter : String
    '''

    # using this re first
    re_result_text =  re.findall(r'"text":"(.*?)"',tweet_data,re.DOTALL)


    # as the previous re used "text" the lenght of list is zero
    # so need a new re to get the text
    if len(re_result_text) == 0 :
        return re.findall(r'"(.*?)","created_at',tweet_data,re.DOTALL)

    # if the previous re could only captured a \ then go here
    elif re_result_text[0] == "\\" :

        # in this re the text ends with "created_at"
        re_result_text = re.findall(r'"(.*?)","created_at',tweet_data,re.DOTALL)

        # if still no data is captured then go here
        if not re_result_text:

            # in this re the text starts with "text" and ends with "id"
            re_result_text =  re.findall(r'"text":"(.*?)","id"',tweet_data,re.
↪DOTALL)

            # if no result is captured
            if not re_result_text:

                # in this re the text cannot contain \
                return re.findall(r'"text":([^\\].*)',tweet_data,re.DOTALL)
            else:
                return re_result_text
```

```
        else:
            return re_result_text
    else:
        return re_result_text
```

1.6.4 is_english_tweet()

Defining function named `is_english_tweet()` to classify if tweet is in English. It uses the `classify()` function from langid pacakge.

[696]:
```
def is_english_tweet(tweet_text):
    '''
    Function to calssify tweet
    Return type: Boolean
    '''

    # tuple has value 'en' for English
    if lg.classify(tweet_text)[0] == 'en':
        return True
    else:
        return False
```

1.7 7. Defining variables to store the Tweet data

Following are the main variables storing all the tweets data.

- `tweets` : This is a dictionary mapping each Tweet ID with its corresponding text.
- `day_tweets` : This is a dictionary mapping the date and the List of tweet ID for that date.

[697]:
```
# mapping of id and tweet text
tweets = {}
```

[698]:
```
# dict of tweets of a day
day_tweets = {}
```

1.8 8. Add mapping for the tweet

Defining function named `add_tweet()` to add the Tweet to dictionary which has all the Tweet Ids mapped to the Tweet text. Using a dictionary ensures that we don not have repeated Tweets

[699]:
```
def add_tweet(tweet_id,tweet_text):
    '''
    Function : Add to dictionary named 'tweets' which maps all the Tweet ID to␣
 ↪their text.
    '''

    if tweet_id not in tweets:
        tweets[tweet_id] = tweet_text
```

1.9 9. Group the Tweets of same date together

Defining function named `collect_tweets()` to keep the Tweet of same date together in the dictionary `day_tweets`. This function first checks if we have a corresponding mapping for that date in our `day_tweets` dictionary. If there is already a list for that date, then the coming Tweed ID is added to that list. Else a new mapping is made in `day_tweets` for the date and a new list is created.

[700]:
```
def collect_tweets(tweet_id,tweet_date):
    '''
    Function : Store thetweet of same date together
    Parameter : tweet_id - Id of tweet
                tweet_date - Date on which the tweet was created
    '''

    # Tweet is of a new date
    if tweet_date not in day_tweets:
        day_tweets[tweet_date] = [tweet_id]

    # tweet is of an existing date
    else:
        list_tweet_ids = day_tweets[tweet_date]
        list_tweet_ids.append(tweet_id)
```

1.10 10. Extarct the data

Defining function named `extract_data()` to extract every Tweet information i.e. Text, ID and date of each tweet.

Parameter of the `extract_data()`:

`file_name` : This is the name of a single `.txt` file out of all the 2421 files.

`file_content` : It is the concatinated String containing all the tweets of a file.

The function gets the concatinated String of a `.txt` file and captures the Tweet text, date and ID using the follwoing Regular expression

`{"id":(".*?")},|{"text":(".*?")},|{"created_at":(".*?")},`

Here I use alternation i.e. `|` to capture the whole tweet information. As the format in which data about Tweet is stored is different for some files, we need 3 regular expressions to capture the data.

- `{"id":(".*?")},` : This regular expression captures the data from file in which the tweet data starts with `{"id":` and ends with `},`. `(".*?")`. Basicaly it gets captures the Text, ID and date created for each tweet. In this Regular expression I use lazy binding i.e. `.*?` to match as little as possible. Otherwise it will match the whole file.
- `{"text":(".*?")},` : This regular expression captures the data from file in which the tweet data starts with `{"text":` and ends with `},`. Basicaly it gets captures the Text, ID and date created for each tweet. Again lazy binding is used to capture data of single tweet.

- `{"created_at":(".*?")},` : This regular expression captures the data from file in which the tweet data starts with `{"created_at":` and ends with `},`. This will capture the Text, ID and date created for each tweet. Lazy binding used to match little as possible.

The ID, text and date of each tweet captured by the regular expression is then passed to the helper functions i.e. `get_tweet_text()`, `get_tweet_id()` and `get_tweet_date()` to capture the Text, ID and date of each tweet from the whole string.

THe text of the tweet is then passed to `is_english_tweet()` function to see if the tweet was in English.

If the Tweet is in English, then `add_tweet()` is used to add the tweet to the common dictionary `tweets` with its corresponding tweet ID.

Also, for English tweets, the `collect_tweets()` is called to add it to common dictionary of each day `day_tweets`.

[701]:
```
def extract_data(file_name, file_content):
    '''
    Function to get the Tweet text, date and ID
    Parameter : file_name - String
                file_content - String
    '''

    # use re.findal all the match the regular expression pattern
    # re.DOTALL flag to include newline from the string.
    re_result_data = re.findall(r'{"id":(".*?")},|{"text":(".*?
 ↪")},|{"created_at":(".*?")},', file_content,re.DOTALL)

    # loop throuht the list of result returned by re.findall()
    for each_tweet_data in re_result_data:

        # first expression catures the data
        if each_tweet_data[0]:
            # get the tweet text
            tweet_text = get_tweet_text(each_tweet_data[0])[0]
            # get the tweet ID
            tweet_id = get_tweet_id(each_tweet_data[0])[0]
            # get the Date
            tweet_date = get_tweet_date(each_tweet_data[0])[0]

            # process only english tweets
            if is_english_tweet(tweet_text):

                # add mapping of tweet and its ID
                add_tweet(tweet_id,tweet_text)
                # store tweets of same date together
                collect_tweets(tweet_id,tweet_date[:10])
```

```
        # second expression catures the data
        elif each_tweet_data[1]:

            # get the tweet text
            tweet_text = get_tweet_text(each_tweet_data[1])[0]
            # get the tweet ID
            tweet_id = get_tweet_id(each_tweet_data[1])[0]
            # get the Date
            tweet_date = get_tweet_date(each_tweet_data[1])[0]

            # process only english tweets
            if is_english_tweet(tweet_text):
                # add mapping of tweet and its ID
                add_tweet(tweet_id,tweet_text)
                # store tweets of same date together
                collect_tweets(tweet_id,tweet_date[:10])

        # third expression catures the data
        elif each_tweet_data[2]:

            # get the tweet text
            tweet_text = get_tweet_text(each_tweet_data[2])[0]
            # get the tweet ID
            tweet_id = get_tweet_id(each_tweet_data[2])[0]
            # get the Date
            tweet_date = get_tweet_date(each_tweet_data[2])[0]

            # process only english tweets
            if is_english_tweet(tweet_text):
                # add mapping of tweet and its ID
                add_tweet(tweet_id,tweet_text)
                # store tweets of same date together
                collect_tweets(tweet_id,tweet_date[:10])
```

1.11 11.Function to write to XML file

1.11.1 tweets_to_xml()

Defining function named `tweets_to_xml()` to write the final Tweet text, ID and date of each tweet to the XML file.

First we open a new file object for file in mode write binary `wb` named file_object.

Writing the XML encoding and declaration to the file.

```
<?xml version="1.0" encoding="UTF-8"?>
```

Then writing the data in the following XML structure.

```
<data>     <tweets>          <tweet> </tweet>     </tweets> </data>
```

We iteratively loop through each date and the Tweets using the `day_tweets` dictionary, for each date to write to the file. For each date, since the tweet ID is stored in a list we loop throught the list and get the text mapped for each ID from `tweets`

We need to replace the newline character i.e. replace `\\n` with `\n`, since Python escapes the `\` for us

[702]:
```
def tweets_to_xml(day_tweets):

    '''
    Function to write Tweet Text, ID and date to the XML file
    Parameter : Dictionary having key as Date and value as List of Tweet IDs␣
↪for that date
    '''

    file_object = open("output_tweets_new123.xml","wb")

    xml_dec = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n"
    file_object.write(xml_dec.encode('utf-8'))
    print(xml_dec)

    xml_data_tag_open = "<data>\n"
    file_object.write(xml_data_tag_open.encode('utf-8'))
    print(xml_data_tag_open)

    for date , list_ids in day_tweets.items() :
        xml_tweets_tag_open = "<tweets date=\"" + date + "\">\n"
        file_object.write(xml_tweets_tag_open.encode('utf-8'))
        print(xml_tweets_tag_open)

        for tweet_id in list_ids:
                tweet_text = tweets[tweet_id]
                tweet_text = tweet_text.replace("\\n","\n")

                if tweet_text == "\\":
                    print(tweet_id)

                if not is_tweet_printed(tweet_id):

                    xml_tweet_tag_open = "<tweet id=\"" + tweet_id + "\">" +␣
↪tweet_text + "</tweet>\n"
                    file_object.write(xml_tweet_tag_open.encode('utf-8'))

                    print(xml_tweet_tag_open)

        xml_tweets_tag_close = "</tweets>\n"
        file_object.write(xml_tweets_tag_close.encode('utf-8'))
        print(xml_tweets_tag_close)
```

```
    xml_data_tag_close = "</data>"
    file_object.write(xml_data_tag_close.encode('utf-8'))
    print(xml_data_tag_close)

    file_object.close()
```

1.12 12. Function to check if the Tweet is already written to XML

Defining function named is_tweet_printed() to check if the Tweet data is already written to XML. If the it is written to XML then the function returns `True` else it adds the Tweet ID to a list of tweets that are written to XML file `list_tweets_written` and returns `False`

[703]:
```
# List of tweets written to XML
list_tweets_written = []
```

[704]:
```
def is_tweet_printed(tweet_id):
    '''
    Function to check if the Tweet has been writtent to the XML file
    Parameter : ID
    Returns : True - if Tweet already written to XML file
              False - if Tweet not written to XML
    '''

    if tweet_id in list_tweets_written:
        return True
    else:
        list_tweets_written.append(tweet_id)
        return False
```

1.13 13 Invoke the Driver function

As we have all the functions and files in place. Lets invoke the driver function `get_results()` to start the Tweet processing.

NOTE : Run all the previous code cells to get the final XML result.

In case of following Error

IOPub data rate exceeded.

The notebook server will temporarily stop sending output

to the client in order to avoid crashing it.

Relaunch Jupyter Notebook using following command

jupyter notebook –NotebookApp.iopub_data_rate_limit=10000000

[706]:
```
get_results()
```

1.14 14. Summary

This assessment helps to build up the knowledge of processing semi-structured data and extract data, which is he fundamental step for Text analysis. The main objective achieved are as follows:

- **Parsing Text files to extract from Raw data** : By using the `listdir`, `dirname` and `isfile` functions from `os` package, along with inbuilt python `open` function, it was possible to read all the `.txt` files from a particular directory.
- **Formulating Regular Expressions** : Developing Regular expression to capture the Tweet text, ID and Date helped to build upon the knowledge of forming Regular expressions. Using `findall` method from `re` package helped to capture a list of data using the defined regular expression.
- **Remove Non-English Tweets** : Using `langid` package it was possible to check if the text of Tweet was in English or not. This was achieved by using the `classify` function.
- **Writing data to XML file** : Using the inbuilt python `write` function it was possible to write the Tweet text, ID and date to `XML` format.
- **Manipulating Python Data structures**: For successful completion of this task knowledge of manipulating basic Data Structures like `list`, `tuple` and `dictionary` was important. Using dictionary functions like `items` and `keys` was helpfult for iteration. Also the `in` operator was needed for various condition check.

1.15 15. References

- Built-in Functions — Python 3.8.6rc1 documentation. (2020). Retrieved 15 September 2020, from https://docs.python.org/3/library/functions.html#open
- os.path — Common pathname manipulations — Python 3.8.6rc1 documentation. (2020). Retrieved 15 September 2020, from https://docs.python.org/3/library/os.path.html
- re — Regular expression operations — Python 3.8.6rc1 documentation. (2020). Retrieved 15 September 2020, from https://docs.python.org/3/library/re.html
- Regex Cheat Sheet. (2020). Retrieved 15 September 2020, from https://www.rexegg.com/regex-quickstart.html
- langid. (2020). Retrieved 15 September 2020, from https://pypi.org/project/langid/1.1dev/
- Regular Expression (Regex) Tutorial. (2020). Retrieved 15 September 2020, from https://www3.ntu.edu.sg/home/ehchua/programming/howto/Regexe.html