CS 172 Twitter Post Collector (Spring 2020)

Group Members: Jeffrey Juan, Eric Chaing, Jose Guzman, Ji Hoon Choi

Collaboration Details:

- Jeffrey Juan
    - Used Tweepy to collect Tweets
    - Wrote function that stored an arbitrary amount of Tweets into a JSON file
- Eric Chaing
    - Wrote function to read through output_tweets.JSON and check each if a Tweet contained a URL
    - Wrote code that followed a URL and retrieved the title of that page
- Jose Guzman
    - Collaborated in writing code that retrieved the title of a page
- Ji Hoon Choi
    - Collaborated in writing code that reads tweets.JSON

Overview of System

We used python and Tweepy to collect tweets. We collected real-time tweets geotagged in Los Angeles and parts of Los Angeles County. Tweets are collected in "collect_tweets.py" and are then stored in "output_tweets.json".

Every time we run the program we erase all data in "output_tweets.json" in order to collect new tweets. At first, we collected all the information given by a tweet, which is not necessary. So we used "parsed_json" from the JSON package to gather the relevant information.

```
parsed_json = json.loads(data)
user = parsed_json["user"]["screen_name"]
urls  = parsed_json["entities"]["urls"]
#parsed_url = parsed_json["expanded_url"]
location = parsed_json["place"]["full_name"]
timestamp = parsed_json["created_at"]
truncated = parsed_json["truncated"]
```

We parsed the username, URLs, location, and timestamp of the tweet. These 4 entities were passed to the JSON file.

The parsed truncated part is to check if the text of the tweet is cut off because it is too big. If truncated is false then we get the original text. If truncated is true that means the tweet was too big to fit in the regular "text" field. So we have to get the text from

“extended_tweet” and then output that to the JSON file.

```
if( not truncated):
    text = parsed_json["text"]

    dictionary ={
        "user": str(user),
        "text": str(text),
        "url" : str(urls),
        "location" : str(location),
        "timestamp" : str(timestamp)
    }
    print("Not extended")

else:
    extendedtext = parsed_json["extended_tweet"]["full_text"]

    dictionary = {
        "user": str(user),
        "text": str(extendedtext),
        "url" : str(urls),
        "location" : str(location),
        "timestamp" : str(timestamp)
    }
```

We then created a dictionary with the information we collect and then used json.dump to write the dictionary to the JSON file.

```
    myStream.filter(locations = [-118.69, 33.73, -117.85, 34.22]) #coordinates are for Los Angeles
```

This line starts the stream of tweets to our program. We filtered our tweets by location but you could also filter by keywords.

After the program has finished grabbing the Tweets and dumping them into a JSON file, we run a function immediately that afterward checks the Tweets for URLs. We do this step after the stream is complete in order to not negatively impact speed and avoid data loss.

```
def URLTitleFinder(tweetFile):
    expanded_url = 'expanded_url'
    with open(tweetFile, "r+") as f:
        for line in f:
            data = json.loads(line)
            pageURL = re.findall('http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\(\), ]|(?:%[0-9a-fA-F][0-9a-fA-F]))+', data["text"])
            if data["url"] != [] and (expanded_url in data["url"][0]) and pageURL != []:
                #print(data["url"][0])
                pageURL = re.findall('http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\(\), ]|(?:%[0-9a-fA-F][0-9a-fA-F]))+', data['url'][0]['expanded_url'])
                if pageURL!= []:
                    for i in range(len(pageURL)):
                        #print(pageURL[i])
                        soup = BeautifulSoup(urllib.request.urlopen(pageURL[i]).read(), "html.parser")
                        title = soup.title.string.encode("utf-8")
                        data.update({"title" : str(title)})
                        json.dump(data, f)
#end of URLTitleFinder
```

The regex shown in the above code looks for any URLs in the designated fields and then stores that URL into a variable. If we wanted to exclude any kind of domain we could insert *\b(?!domainname\b)\w+* into the regex after the *//* and we will look for links that do not include a certain domain. Additionally, this function is written as a single read and write operation with the “r+” option when opening the file. We then want to append the JSON file by adding the respected title. By doing so, we use the JSON update to allow us to modify the dictionary function, to which we used the JSON dump to write in the title.

Running the Program

Make sure you have all the dependencies as stated in the requirements.txt file installed on your machine. Then just run the program with the following command:

python collect_tweets.py

Limitations

The program uploaded to iLearn only collects 1000 tweets for the simplicity of the demo.

Working Program

The program will output some messages as it is running and when it is finished running to notify the user of the program's state as shown below. The results of the program are written into a file that is named *output_tweets.json*. This will contain the end result with the revised objects having a title field when applicable. We have uploaded a small sample of 26 Tweets in the output file inside this zip file.