Edureka Mastering Python Certification Project

TWITTER SENTIMENT ANALYSIS

By,

Maneesh D

Ph: 7795826802

Problem Statement

IMDB provides a list of celebrities born on the current date. Below is the link: http://m.imdb.com/feature/bornondate

Get the list of these celebrities from this webpage using web scraping (the ones that are displayed i.e. top 10). You have to extract the below information:

1. Name of the celebrity
2. Celebrity Image
3. Profession
4. Best Work

Once you have this list, run a sentiment analysis on twitter for each celebrity and finally the output should be in the below format

1. Name of the celebrity:
2. Celebrity Image:
3. Profession:
4. Best Work:
5. Overall Sentiment on Twitter: Positive, Negative or Neutral

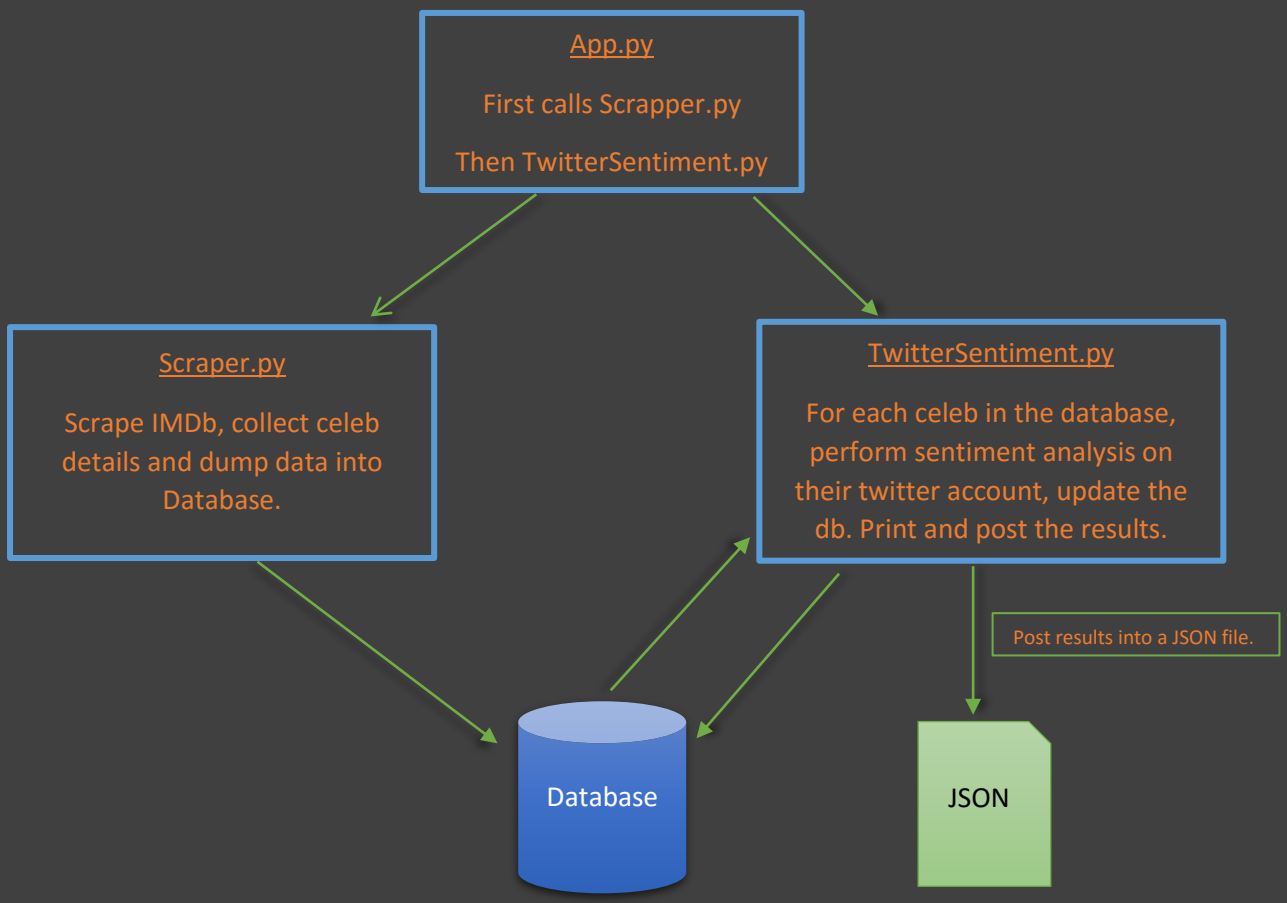
Tools and Libraries Used

1. Python 3.6.1 (64-bit) - Well, you know what it is.
2. Beautifulsoup4 - Python library for pulling data out of HTML and XML files.
3. Tweepy - Open Source python library for Twitter API.
4. Selenium - The web driver kit emulates a web-browser and executes JavaScript to load the dynamic content.
5. Textblob - Python library to perform sentiment analysis
6. lxml - A fast html and xml parser for beautifulsoup4
7. Mozilla Firefox - Web Browser to perform web scraping.
8. Gecko Driver - Driver for Selenium to invoke Firefox.

Challenges Faced

The IMDb website contains dynamic content loaded by JavaScript. The web scraping tool *beautifulsoup4* doesn't invoke the JavaScript. Hence was facing problem with scraping the data. This was solved by first opening the webpage using *Selenium* which can invoke JavaScript and load dynamic content. The page source (which now contains the data needed) got from selenium driver is then passed to beautifulsoup4 scraper, thus solving the problem.

Solution Flow Diagram



SOLUTION

1. App.py

```
"""
@author: Maneesh D
@date: 05-Jun-17
@intepreter: Python 3.6.1

Application to scrape IMDb for celebrity details and then perform Sentiment
Analysis on the celebrities Twitter account.
"""
from datetime import datetime
from json import dump
from os import getcwd
from sqlite3 import connect

SCRAPER = __import__("Scraper")
SENTIMENT = __import__("TwitterSentiment")
URL = "http://m.imdb.com/feature/bornondate"


def main():
    """
    Scrape IMdb for celb data and perform sentiment analysis on their tweets.
    """
    print("TWITTER SENTIMENTAL ANALYSIS")
    print("-" * len("TWITTER SENTIMENTAL ANALYSIS"))

    # get the scraper object to scrape IMDb
    my_scraper = SCRAPER.ImdbScraper(URL)
    print("Scraping IMDb...Please wait...")
    my_scraper.scrape_imdb()
    print("Successfully scraped IMDb...\n")

    # Perforn Twitter Sentiment Analysis
    sentiment_analyzer = SENTIMENT.TwitterSentiment()
    result = sentiment_analyzer.get_twitter_sentiment()
    if result == -1:
        print("Twitter Sentiment Analysis Failed...")
        exit(1)
    print("\nThe Twitter Sentiment Result:")
    print("-" * len("The Twitter Sentiment Result:"))
    try:
        with connect("./data/celebData.db") as con:
            cur = con.cursor()
            # Get the result data from db and print.
            cur.execute("SELECT * FROM CELEB_DATA;")
            celeb_list = list()
            for row in cur.fetchall():
                celeb = dict()

                print("Name: %s" % row[0])
                celeb["name"] = row[0]

                print("Photo: %s" % row[1])
```

```
                celeb["Photo"] = row[1]

                print("Profession: %s" % row[2])
                celeb["Profession"] = row[2]

                print("Best Work: %s" % row[3])
                celeb["Best Work"] = row[3].replace('"', "")

                print("Overall Twitter Sentiment: %s\n" % row[4])
                celeb["Twitter Sentiment"] = row[4]
                celeb_list.append(celeb)
        # Dump the results into a JSON file.
        suffix = datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
        file_name = "Sentiment-Analysis-Result_%s.json" % suffix

        with open("./results/%s" % file_name, "w") as f_stream:
            dump(celeb_list, f_stream, ensure_ascii=True, indent=2)

        print("Result JSON created: %s" % (getcwd() + file_name))
    except Exception as exp:
        print("An Exception Occurred:\n%s" % exp)
        exit(1)
    print("THANK YOU")
    return 0

if __name__ == '__main__':
    main()
```

2. Scraper.py

```
"""
@author: Maneesh D
@date: 05-Jun-17
"""
from sqlite3 import connect, Error
from bs4 import BeautifulSoup
from selenium.webdriver import Firefox


class ImdbScraper:
    """
    Scrape IMDb for Celebrity Data.
    """
    def __init__(self, url="http://m.imdb.com/feature/bornondate"):
        """
        Constructor
        :param url: URL for the IMDb page.
        """
        self.__url = url
        self.__celeb_list = list()

    def __dump_into_db(self):
        """
        Dump the data into db.
        :return: None
        """
        try:
```

```
            with connect("./data/celebData.db") as con:
                cur = con.cursor()
                cur.execute("DROP TABLE IF EXISTS CELEB_DATA;")
                cur.execute("CREATE TABLE CELEB_DATA("
                            "NAME TEXT, "
                            "PHOTO TEXT, "
                            "PROFESSION TEXT, "
                            "BEST_WORK TEXT, "
                            "SENTIMENT TEXT DEFAULT '');")
                con.commit()

                for celeb in self.__celeb_list:
                    cur.execute("INSERT INTO CELEB_DATA(NAME,"
                                "PHOTO,"
                                "PROFESSION,"
                                "BEST_WORK) "
                                "VALUES(?, ?, ?, ?);",
                                (celeb.get("Name"),
                                 celeb.get("Photo"),
                                 celeb.get("Profession"),
                                 celeb.get("Best Work"),))
                con.commit()
        except Error as err:
            print("!!! SQLITE3 ERROR: %s !!!" % err)
            exit(1)

    def scrape_imdb(self):
        """
        Scrapes the IMDB born_on_date page to collect the top 10
        celebrity data.After collecting the data stores it in a sqlite3 db.
        :return: None
        """
        try:
            # Initialize the firefox driver.
            driver = Firefox(executable_path="./utils/geckodriver.exe")

            # Run the dynamic content on the webpage.
            driver.get(self.__url)
            driver.implicitly_wait(30)

            # Get the loaded webpage source.
            page_source = driver.page_source

            # Close the driver.
            driver.close()

            # Create a beautifulsoup crawler and load the page data.
            try:
                crawler = BeautifulSoup(page_source, "lxml")
            except Exception:
                crawler = BeautifulSoup(page_source, "html.parser")

            # Get the required details from the page
            page = crawler.find("section", "posters list")
            born_on_date = page.findChild("h1").text
            print("Getting Data for celebrities born on %s.." % born_on_date)

            # Get the celeb details from HTML and put it in the celeb list
            for link in crawler.find_all("a", class_="poster "):
```

```
                celeb = dict()
                # Parse  celeb name
                name = link.find("span", "title").text

                # parse celeb pic
                img = link.img["src"]

                # get profession and best_work
                profession, best_work = link.find("div",
                                                  "detail").text.split(",")

                # Form a dict for the celeb and push it into celeb list.
                celeb["Name"] = name
                celeb["Photo"] = img
                celeb["Profession"] = profession
                celeb["Best Work"] = best_work
                self.__celeb_list.append(celeb)

            # Dump the data into sqlite3 db
            self.__dump_into_db()

            print("Data Successfully Scraped and dumped into db...")
        except Exception as exp:
            print("Exception: %s" % exp)
            exit(1)
```

3. TwitterSentiment.py

```
"""
@author: Maneesh D
@date: 05-Jun-17
"""
from re import sub
from sqlite3 import connect
from textblob import TextBlob
from tweepy import API, OAuthHandler


class TwitterSentiment:
    """
    Twitter Sentiment Analyzer
    """
    def __init__(self):
        """
        Constructor
        """
        self.__consumer_key = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
        self.__consumer_secret = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
        self.__access_token = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
        self.__access_token_secret = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"

    @staticmethod
    def __get_data():
        """
        Get the Celeb names from db.
        :return: List of celeb data.
        """
        try:
            with connect("./data/celebData.db") as con:
```

```
            cur = con.cursor()
            # Return all the celeb names in db.
            cur.execute("SELECT NAME FROM CELEB_DATA;")
            return cur.fetchall()
    except Exception:
        print("!!! An Exception Occurred: "
              "Could not get celeb data from db !!!")
        return -1

@staticmethod
def __dump_data(data):
    """
    Update the sentiment data in db.
    :param data: list with dict of (celeb_name: sentiment)
    :return: None
    """
    try:
        with connect("./data/celebData.db") as con:
            cur = con.cursor()
            for d in data:
                # Update table with sentiment for the celeb.
                for key in d.keys():
                    cur.execute("UPDATE CELEB_DATA SET "
                                "SENTIMENT=? WHERE NAME=?;",
                                (d.get(key, "NA"), key,))
                    con.commit()
    except Exception:
        print("!!! An Exception Occurred: "
              "Could not update sentiment data in db !!!")
        return -1

@staticmethod
def __normalize_tweet(tweet):
    """
    Utility function to clean tweet text by removing links,
    special characters using simple regex statements.
    """
    return ' '.join(sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])"
                        "|(\w+://\S+)", "", tweet).split())

def __get_tweet_polarity(self, tweet):
    """
    Get the tweets sentiment polarity.
    :param tweet: The tweet text.
    :return: Polarity of the sentiment of tweet.
    """
    analysis = TextBlob(self.__normalize_tweet(tweet))
    return analysis.sentiment.polarity

def get_twitter_sentiment(self):
    """
    Analyze the tweet sentiment.
    :return: 1 if success else -1
    """
    try:
        print("Getting celeb data from db...")
        res = celeb_data = self.__get_data()
        if res == -1:
            return -1
```

```
        print("Authorizing app...")
        # Create OAuthHandler object
        auth = OAuthHandler(self.__consumer_key,
                            self.__consumer_secret)

        # Set access token and secret
        auth.set_access_token(self.__access_token,
                              self.__access_token_secret)

        # Create tweepy API object to fetch tweets
        api = API(auth)

        # For each celeb in celeb list perform setiment analysis
        sentiment_list = list()
        print("\nPerforming Twitter Sentiment Analysis...Please Wait...")
        for celeb in list(celeb_data):
            """
            For each celeb, get the tweets, calculate sentiment and
            form a dict(celeb: sentiment)
            """
            celeb_sentiment = {}
            negative = 0
            positive = 0
            neutral = 0
            celeb_name = celeb[0]

            # Get the last 100 tweets from twitter for the celeb.
            tweets = api.search(q=celeb_name, count=100)
            if len(tweets) == 0:
                celeb_sentiment[celeb_name] = "Does not have a " \
                                              "Twitter Account"
                continue
            for tweet in tweets:
                # get the tweets sentiment polarity
                tweet_polarity = self.__get_tweet_polarity(tweet.text)

                # decide positive negative or neutral based on polarity
                if tweet_polarity > 0:
                    positive += 1
                elif tweet_polarity == 0:
                    neutral += 1
                else:
                    negative += 1
            # Decide the overall sentiment for the celeb
            if positive >= neutral and positive >= negative:
                celeb_sentiment[celeb_name] = "POSITIVE"
            elif neutral >= positive and neutral >= negative:
                celeb_sentiment[celeb_name] = "NEUTRAL"
            else:
                celeb_sentiment[celeb_name] = "NEGATIVE"

            # Append the dict to sentiment list for later use.
            sentiment_list.append(celeb_sentiment)
        print("Twitter sentimental analysis completed...")

        print("Updated db with sentiment data...")
        res = self.__dump_data(sentiment_list)
        if res == -1:
```

```
                return -1
            print("Sentiment data successfully updated in db...")
            return 1
        except Exception as exp:
            print("!!! Exception: %s !!!" % exp)
            return -1
```

THANK YOU