OMIS 107 Project: Documentation

Kai Hirota and Marc Gehrig

About

Components:

- Data - All of the html, csv, and image files exported/rendered will be saved in this folder.
- Documentation.pdf - Design specifications and how the program works.
- commands.txt - All of the available commands for twitter.py
- config.txt - API keys, misc.
- sentimentDemo.py - Demo of VADER, which enables the sentiment analysis. Will enter infinite loop upon execution. On each loop, it will wait for string input, and then print the sentiment scores of the given input. Enter "quit" to exit loop.
  tweets.db - Where all the collected tweets are stored, along with each Tweet's sentiment scores.
- twitter.py - Main program.
- ShellScript - Shell script that makes the program a bit more user-friendly.

Available commands

- python twitter.py collect Trump 500
  - collect the latest 500 tweets containing the keyword "Trump", and save to tweets.db
- python twitter.py analyze Trump stats
  - create an overview stats of all numeric fields, including sentiment scores (Mean, Standard Deviation, Min, Max, Quartiles) saved as a csv file inside Data folder
- python twitter.py export Trump

    - export all tweets with the keyword "Trump" as a csv file, saved inside Data folder

Modes of Visualization

- python twitter.py analyze Trump interval
- python twitter.py analyze Trump line
- python twitter.py analyze Trump dist
- python twitter.py analyze Trump scatter
- python twitter.py analyze Trump pie
- python twitter.py analyze Trump map

SQL

- sqlite3 tweets.db .tables
    - display list of stored tables (each table is named after the keyword. e.g. "Trump"
    - "#" will be renamed to HASH
        - e.g. if you collect tweets with the search term "#Trump," the tweets will be stored in a table called "HASHTrump" inside tweets.db
    - Don't worry too much about this - I added functions that will automatically convert between "#" and "HASH" as needed, so there is no need to ever type "HASH"
- sqlite3 tweets.db .schema
    - display schema

Main functions

`collect` mode

1. Search Twitter for tweets containing the keyword or the hashtag specified by the user.
    - Filters out retweets
    - Requests for full_text, which includes longer tweets
    -

- Search result limited to English
- In the shell, there is a feature that allows users to collect tweets every hour for 12 hours to achieve a more accurate data set for analysis

2. Process & Analyze
    1. Tokenizes the tweet text
    2. filters out stop words
    3. performs sentiment analysis on the filtered text
    4. dekonizes the tweet text. Sentiment analysis was done using VADER (Valence Aware Dictionary and sEntiment Reasoner)
    - "VADER Sentiment Analysis. VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media, and works well on texts from other domains" https://github.com/cjhutto/vaderSentiment
    - Valence - "the number of grammatical elements with which a particular word, especially a verb, combines in a sentence."
    - VADER Sentiment analyzer recognizes emojis and informal speech. It's tuned for analyzing unstructured text collected from social media.
3. Load all the available attributes of a tweet + `tweetNoSW` which is the tweet text without stop words + Positive / Neutral / Negative / Compound sentiment scores into a Pandas DataFrame, then exports to `tweets.db` inside the same directory. If it doesn't exit, the program will create a new `tweets.db` file.

`analyze` mode

1. Perform visualization. Available methods: line (time-series analysis), scatter plot, pie chart, probability distribution using a module called plot.ly.
    - Output: Generate an html file inside `./Data` and open it automatically via browser.
2. Can also visualize and create line graph using matplotlib.
    - Output: a pop-up window of the graph. Will also save it as a png file in `./Data`
3. Can also export csv file of df.describe(). If csv file already exists, append

datetime in addition to the new output. This is for users who wants to quickly gain a quantitative glimpse into the dataset.

`export` mode

1. Export all of the data in `[keyword] table` inside `tweets.db` as a csv file in `./Data` If file exists, append.

Design specification

- The program does not require to be run on Jupyter Notebook, but it does require all the modules + others such as cufflinks, plotly, vaderSentiment.vaderSentiment, and nltk. My apologies for the heavy requirements.
- When running the program on terminal, the CWD has to be inside the directory that contains all the files.

Output / storage

- Upon successful execution, program will print `Task successfully completed`
- All output files will be named `"HASH" + keyword + "_[pie / scatter / line / etc]"` to make it easier for the user to distinguish between different types of visualizations and output files.
- Any keyword entered that begins with "#" will have "#" stripped off, and replaced with "HASH." This is to ensure that "#" doesn't get processed by any functions except for when searching for tweets.
- Opted to use sqlite database as storage to ensure fast retrieval and efficiency

Cancelled

- Tried to split userLocation into City and State in an attempt to visualize sentiment by US State. Was able to implement, but too much variance in format. Some write "USA," while others write "LA" or "CA."
    - Later we succeeded in implementing this.
- Was intending on adding linear regression and trend line in the scatter plot, but ran out of time.