Twitter Data Mining and Visualizations

Project Description

a) Overview

The Internet has allowed people to share and post information on social media platforms. Data is everywhere. But how to make the data useful? Are we able to find patterns and behavior? So many questions to be answered. We could start with Twitter, by analyzing the tweets, and see if we could understand what the people are talking about. We could also transform the meaningful data into a visualization to view!

In this assignment, I will be using Docker to house the project. I will need to set up Python with docker. I will use python as my backend. It will be a back-end rendered website. Before I can do anything, I must have a Twitter account and successfully obtain a twitter API token. I will learn to stream tweets and create plots using GGPLOT with filters to better understand the data. None of this is possible. To make this more fun, I will create a script which will conduct a sentimental analysis on the tweets. Each tweet will be interpreted and classified as positive, negative or neutral. I will be following the MVC architectural pattern. On top of that, I will create a Twitter bot which will interact with tweets. Test cases will be written to ensure the web app is running flawlessly.

Purpose

a. Learn how to use Docker
b. Work with APIs for web services
c. Understand Async vs Sync
d. Start a localhost
e. Understand how sentimental analysis work
f. Become more familiar with the MVC model
g. Represent data through visualization
h. Create a Twitter bot

b) Motivation

Docker is everywhere, just like open source. Docker applications don't require any specific operating system to work. It is an application that you

can download and start playing with it. I will get to explore various docker applications. The docker application will be the base for this project. I will use Twitter API to pull in tweets to verify that it works. Once there's data, I will need to conduct a sentimental analysis on the tweets, which is the process of computationally identifying and categorizing opinions expressed in a piece of text. I will learn how to create visualizations using GGPLOT.

c) Technical Specification

a. Tools: Docker
b. Architecture: MVC architecture
c. Database: SQLiteDB
d. Platform: Docker
e. Backend Server: Python (Flask/Django)
f. Visualization: GGPLOT
g. API: Twitter API
h. Web Framework: Flask
i. Libraries:
j. Programming Languages: Python, JavaScript
k. Frontend: HTML, CSS
l. IDE: Notepad++ or sublime

d) Architecture: MVC Model

a. Backend (Python [Flask/Django])
    i. Controller: Control the URL and map it
    ii. Model (create classes for Profile, tweets, etc)
        1. We call API (in JSON) to SQLiteDB
    iii. Frontend (html, CSS, JS)
        1. Views(html/CSS/JS)
        2. Display front end data with visualization (GGPLOT, etc)

e) High-level functionalities

a. Stream tweets
b. Interface:
    i. Command line interface that shares code with web application
c. Navigation Bar
    i. My profile

    ii. Home
    iii. Sign Out
    iv. Visualizations
d. Web app styling
    i. Hovering over a tweet to enlarge the text
    ii. Change of opacity
    iii. Current page/tab highlighted
e. DIY sentimental analysis
    i. Users can set their own parameters
    ii. Paste a string of text in text box, and run it to label their tweets as (positive, negative, or neutral)
f. DIY visualization
    i. User can upload JSON file or CSV file containing tweet data
    ii. Select one visualization (options for user to choose)
    iii. Generate a visualization based on data (only if it is clean data)
g. Write my own sentimental analysis for each tweet (positive, negative, neutral)
h. Write my own Twitter Bot which will interact with new tweets (like, comment, and/or retweet)
i. Create visualizations using GGPLOT

f) Score of the project

a. Product Scope: This will be a backend rendering website. I will use Twitter API to pull real time tweets and displaying it onto my website. A Twitter bot will be used to interact with tweets. Ability to understand and make sure of the MVC, and keep the Twitter feed updated. Data will then undergo a sentimental analysis which will rank the tweets with either (positive, neutral, or negative). Data will be analysis and visualizations will be created. To make this even more interactive and fun, users can upload their tweets onto the web app to create visualizations. They can also set their own parameters before conducting a sentimental analysis of a tweet.
b. Deliverables: Will be outline below
c. Additional Risks:
    i. Computer stops working

ii. Constraints and assumptions: Limitations of Twitter API, and learning new technologies

g) Sketch of GUI

Weekly goals

1) **Week 1:**
    a. Understand how docker works by visiting https://www.docker.com/
    b. Install Python on Docker
    c. Create a Twitter account if you do not have an account
    d. Obtain a Twitter API token
    e. Verify the Twitter API token is working by sending request to Twitter's API to request for get_followers()
    f. Create a web application that will display user info: username, name, followers count, following count, 10 recent tweets and/or 10 recent retweets
    g. Stream tweets by specifying parameters
    h. Save tweets in JSON format
    i. Create visualizations
        i. Number of tweets per 10 secs
        ii. Number of tweets per minute
    j. Create the nav bar
        i. Landing page
        ii. Profile
        iii. Visualization
        iv. DIY visualization
        v. DIY sentimental Analysis
    k. Web app styling:
        i. Hovering over a tweet to enlarge the text
        ii. Change of opacity
        iii. Current page/tab highlighted

2) **Week 2:**
    a. Create my own sentimental analysis is written in Python
    b. Training the sentimental analysis model by uploading my tweets data.
    c. After labeling tweets, display 10 tweets per category (Positive, Negative, Neutral) on my web app
    d. Create visualization using
        i. Create a visualization that displays my sentimental analysis result of the tweets
    e. Create the following views:
        i. Landing page
            1. Tweets (in real time)
        ii. Profile page
            1. Display Username
            2. Display Avatar
            3. When did you join Twitter?
            4. Followers count
            5. Following Count
            6. Your tweets
            7. Retweets
    f. Twitter bot (one or more of the following):
        i. Retweet
        ii. Like
        iii. Comment
3) **Week 3:**
    a. Create two more views
        i. DIY sentimental analysis
            1. Create a script that allows user to adjust the parameters before conducting a sentimental analysis on it
            2. Create a script that allows the user to paste or type in a string of text
            3. Create a button to proceed to run the sentimental script based on parameter
            4.
        ii. DIY visualization

1. Create a script that allows the user to upload JSON or CSV files
2. Create a Script that display which visualizations we can offer
3. User can select one visualization and the visualization will appear on the screen

b. Command line interface
   i. Build a command line interface for landing page and home page
   ii. Display same information that is on the screen

c. Ranking and statistic of sentimental analysis
   i. Display the statistic of the result (i.e. how positive it is, or how negative it is
   ii. Highlight the words in the tweet/statement that shows positive and/or negative and/or neutral

Rubrics:

Week 1: April 12

Category	Score	Detailed Rubrics
Successfully install Docker onto machine and make sure it is working using localhost	1	0: No Docker 1: Docker working with Python localhost
Display real time tweets using Twitter API properly	3	0: Not implement 1: Static tweets 3: Dynamic tweets (streaming tweets).
Create visualizations 1) Number of tweets per second 2) Number of tweets per minute	4	0: Not implement 2: Visualization created without cleaning the tweet data 4: Visualization created with cleaned tweet data from JSON.
Web App styling 1) Hovering 2) Change of opacity 3) Highlighting	3	0: Not implement 3: Web app with styling effect and code is structured

Nav Bar 1) Landing Page 2) Profile 3) Visualization 4) DIYs	4	0: Not implement 1: Nav bar created with button selection 1 button 4: Nav bar created with the selection buttons, code is structured.
Test 1: Python unittest case: Test authentication 1) Object is initialized 2) Class variables initialized	2	0: Didn't implement test 1: 50 % coverage implemented 2: Test implement with code coverage of at least 95% and code is structured
Test 2:Python unittest case: Root url: 1) Html file name is correct 2) Called index.html	2	0: Didn't implement test 1: 50 % coverage implemented 2: Test implement with code coverage of at least 95% and code is structured
Test 3: Python unittest case: App server and database	4	0: Didn't implement test 3: 60 % coverage implemented 4: Test implement with code coverage of at least 95% and code is structured
Test 4: Python unittest case: Test if API Token is working	2	0: Didn't implement test 1: 60 % coverage implemented 2:Test implement with code coverage of at least 95% and code is structured

Week 2: April 19

Category	Score	Detailed Rubrics
Sentimental analysis (create my script based on existing algorithms) to categorize each tweet (Positive, Negative, or neutral)	5	0: No code 2: Partially process of data (40%) 5: Completed function and code is structured

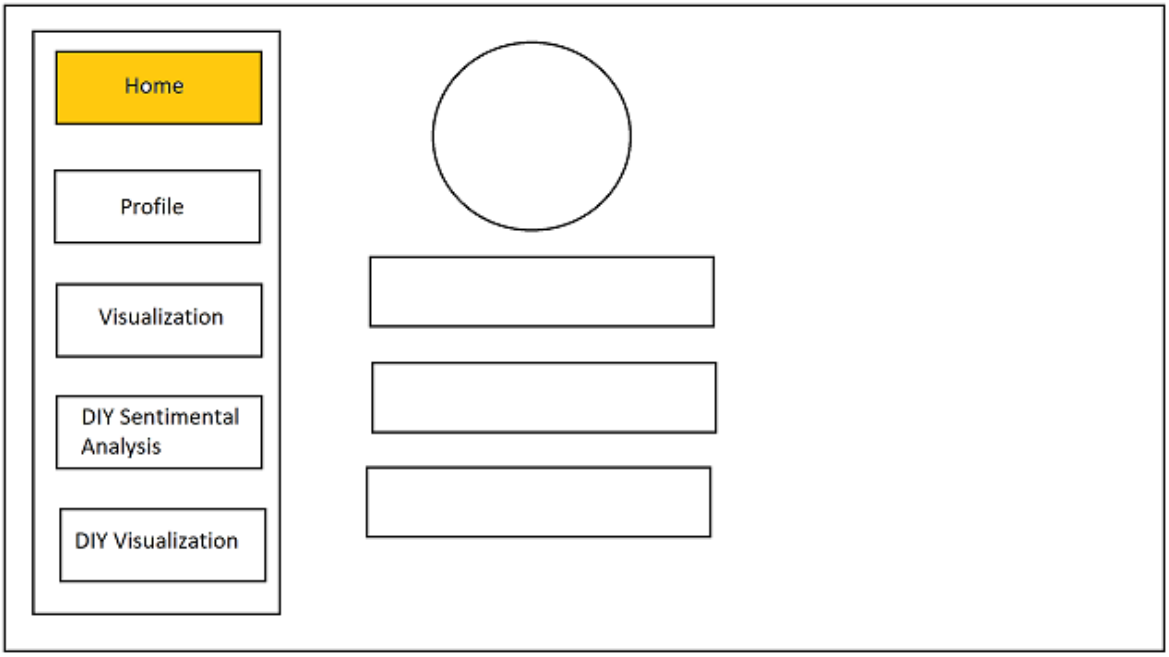
Create a visualization 1) displays my sentimental analysis result of the tweets	4	0: Not implement 2: Visualization created without cleaning the tweet data 4: Visualization created with cleaned tweet data from JSON.
Creating the following views 1) Landing page 2) Profile Page 3) Visualization Page	3	0: No code 1: Implementation of just one of the views and code is structured. 2: Program works, but does not have a clear functional organization of the MVC. 3: MVC organization implemented, and functionality works.
Twitter Bot: 1) Like 2) Retweet 3) Comment	3	0: No code 2: Partially process of data (66%) 3: Completed function and code is structured
Test 1: Python unittest case: API (used my twitter API rather than from a sample API)	2	0: Didn't implement test 1: 60 % coverage implemented 2: Test implement with code coverage of at least 95% and code is structured
Test 2: Python unittest case: Test case for each view to ensure they are displayed with the proper information	5	0: Didn't implement test 1: 30 % coverage implemented 3: 60 % coverage implemented 5: Test implement with code coverage of at least 95% and code is structured
Test 3: Python unittest case: Twitter bot. Make sure it can do the following: 1) Retweet 2) Comment 3) like	3	0: Didn't implement test 1: 1 feature covered 2: 2 features covered 3: 3 features covered

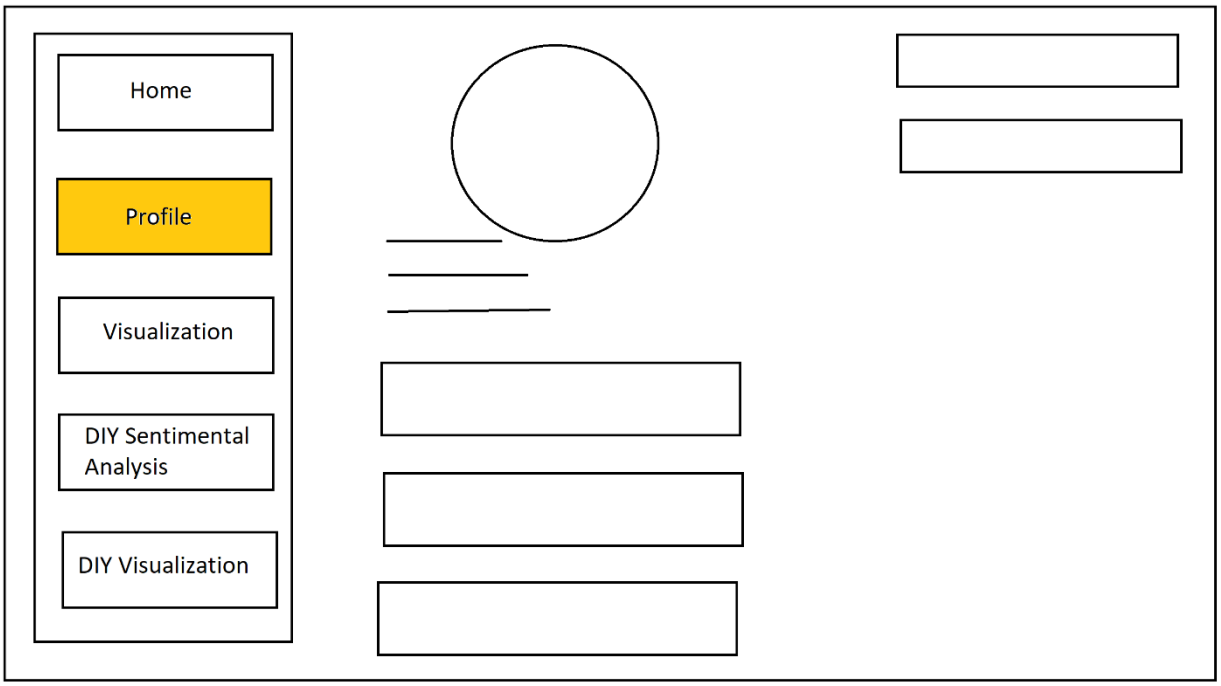
Week 3: April 26

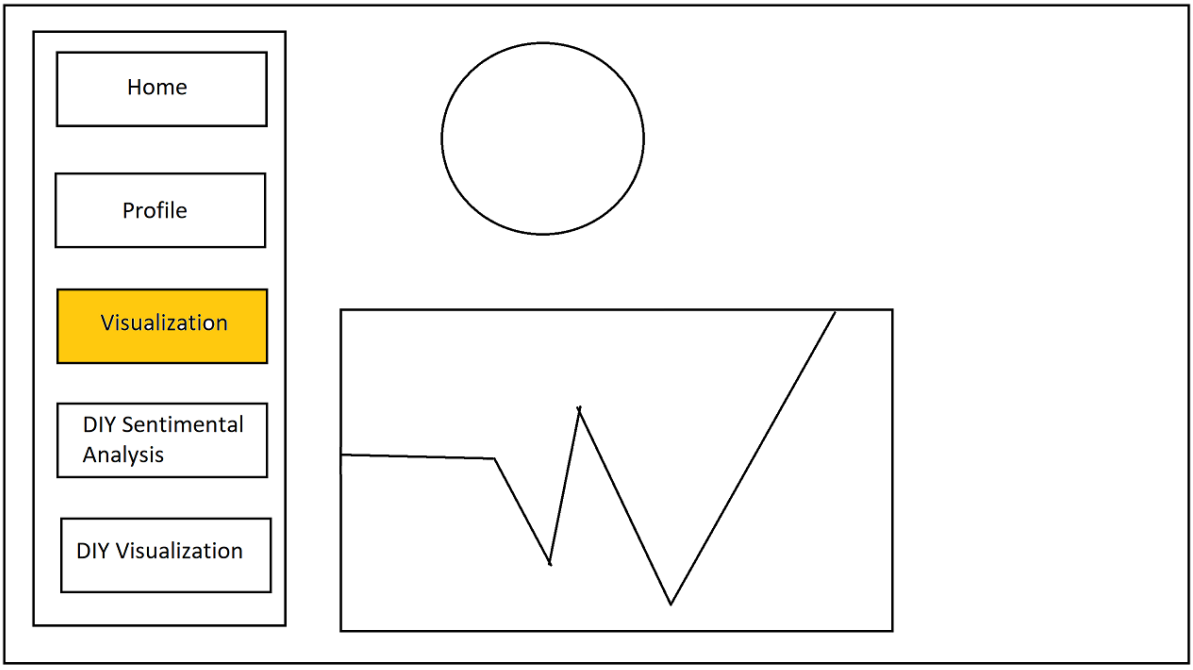
Category	Score	Detailed Rubrics
DIY sentimental analysis 1) Asks for user' string input 2) Allows user to adjust the parameter 3) Modify the sentimental analysis code with the user's input 4) Display the labeled tweet/string (positive, negative or neutral) 5) Organize the keywords by positive, negative, and/or neutral 6) Show how positive, how negative or how neutral it is	6	0: Didn't implement 3: Program partially works, but does not have a clear functional organization of the MVC. 6: MVC organization implemented, and functionality works.
DIY visualization 1) Ask for user to upload a file (JSON or CSV) 2) Ask user to select which type of visualization the use want 3) Display the visualization with the provided data	5	0: Didn't implement 2: Program partially works, but does not have a clear functional organization of the MVC. 5: MVC organization implemented, and functionality works.
Command line interface 1) Display information that is on Home page and Profile Page	4	0: Didn't implement 2: Program partially works, but does not have a clear functional organization of the MVC 4: MVC organization implemented, and functionality works
Test 1: Command line interface for Homepage and Profile page	4	0: Didn't implement test 2: 50 % coverage implemented

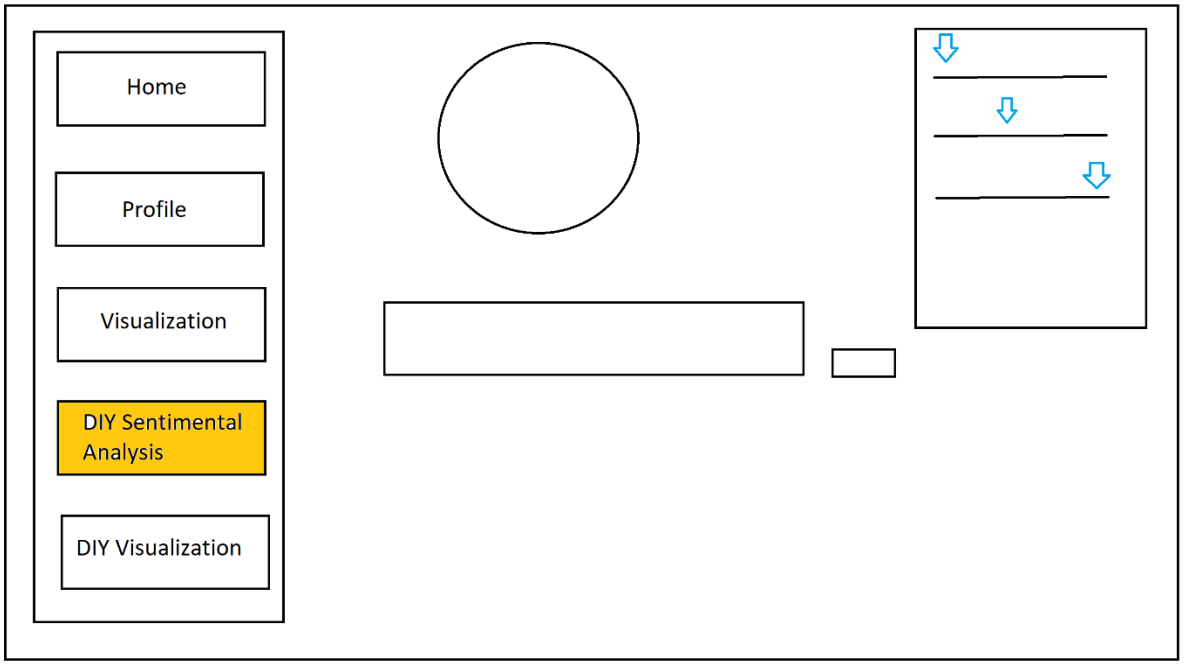
		4: Test implement with code coverage of at least 95% and code is structured
Test 2: Python unittest case: Test case for each view to ensure they are displayed with the proper information	6	0: Didn't implement test 3: 60 % coverage implemented 6: Test implement with code coverage of at least 95% and code is structured

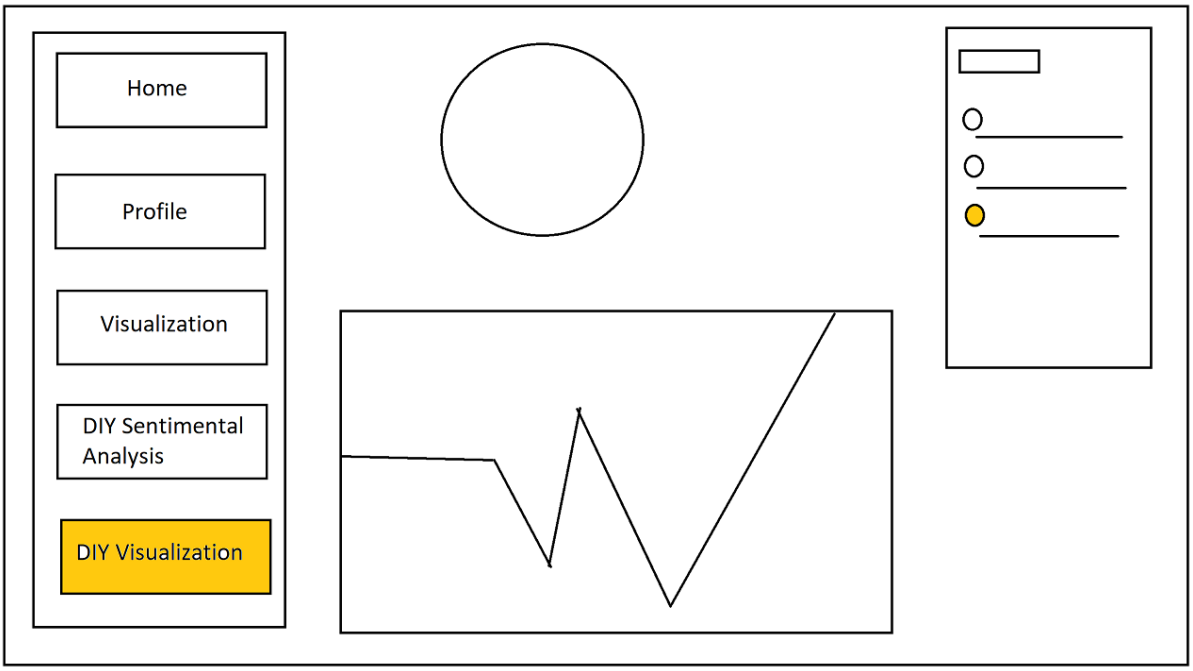
Sample GUI


Home
Profile
Visualization
DIY Sentimental Analysis
DIY Visualization


Home
Profile
Visualization
DIY Sentimental Analysis
DIY Visualization


Home
Profile
Visualization
DIY Sentimental Analysis
DIY Visualization


Home
Profile
Visualization
DIY Sentimental Analysis
DIY Visualization


Home
Profile
Visualization
DIY Sentimental Analysis
DIY Visualization