Justin Hibbard
CS Senior Project
Final Documentation

Twitter Data Miner

1. Introduction

Goal:

- The overall goal of my senior project was to create a Twitter Data Mining application where the user was given the options to search tweets of any Twitter handle given and filter those tweets by filter words given by the user. You could select the number of tweets and really be specific on what you are trying to find on Twitter whether that be on your own account and other people's accounts. I really wanted more than just a general user to be able to use this so a business side was a big aspect in the overall future of the project. This is where businesses could log into their Twitter accounts and could post tweets directly from the application. By allowing people to log in they could also delete tweets as well. With the appropriate filters given you could also compare tweets for positive versus negative polarity. Comparing popularity with retweets between certain filter words and looking at the overall trend of retweets and favorites became important as well. After all, the general features that I wanted to complete are all capable on my application now.

Uses of the Application:

- There is several uses of my Twitter Data Mining application which I think you can separate in a business aspect and a user aspect.
  - **For the user:** A user may log in to their own Twitter account, post tweets, and remove tweets as well. But where this is useful is looking at an example like your job searching. You don't want companies to find your old tweets where there may be profanity which can really ruin your chances if a company was to look through your Twitter account. This negative publicity is not good for anyone and if you gave this application to professional athletes like for example entering the NFL draft then they would be able to stop this by cleaning up their Twitter using my application.

  - **For the business:** A business may log in to their own Twitter account and post tweets, remove tweets just like the user. This can be used by the person in the company who runs the Twitter account for their company to do everything from one interface. But where this really comes helpful to businesses is using the data interface tab where we can search a company as the filter word with the number of tweets you want to look like. You will get the output of the tweets, look at them as you want, but also graph the data. The sentiment analysis will give you a basis of the tweets on how positive or negative they are related. The compare two filter words can be useful to compare the popularity between two businesses on recent tweets. Also, just looking at your search results by graphing the recent retweets or favorites.

What was used:

- **Language:** The primary language that the application was written in was Python, I used the PyCharm CE IDE in order to make it easier to write in.

- **Interface:** The GUI was created in QtDesigner in which if you import PyQt5 into Python then you can import that GUI file into a Python file and directly grabbed from my main application for use.

- **APIs:** Twitter APIs was used along with the Tweepy library to connect to Twitter with my developer account in order to gain access to tweets. Using the APIs, you can log in to Twitter through my application to gain additional access to features.

2. Project Design

♦ **TwitterDataMiner Project Files:**

a. progButton.py

- Summary:
  This is the main file that is used to run the application, this imports the GUI file and I will break down every function for each use. To start the application, this is the essential part that is needed by the user in order for it to begin. If you look at outside of the main class you will see the authentication variables that is done through the Tweepy Module which calls the Twitter API Services. Every time someone logs in while using the application, the AUTH tokens will be updated so that they are using their own account now. Below are the libraries that were used in the making of the application along with all of the functions in depth of what they essentially do.

- Libraries:
  1. **Tweepy:**
     - Tweepy is a library that is used to grab data easier from Twitter to your Python project. In order to begin this, you need to make sure to register for a Twitter Developer Account. Once approved you take the consumer key and access token that is given through Twitter to set up the authentication that Tweepy offers. Tweepy has all type of methods built in that allow to grab people's tweets with some options to call on specific users or some filter words. The most difficult part is implementing this Tweepy library functionality into a working GUI with customizable options for the user.

  2. **PyQt5:**

  - PyQt5 is the library that is supported by the Qt Designer application that I used to make the interface. Once you import this library you can call on the GUI that you made in Qt Designer and once you change it into a Python file you have the customizability to change all of the functionality of the buttons, labels, and input boxes you created.

3. **MatPlotLib:**
   - MatPlotLib is the library that I used to create the graphing features of the application. When the buttons are pressed it calls on a function that calls on the MatPlotLib library that has pre-built methods to creating graphs easier. It basically allows for a separate interface to be displayed with some graphing customizability.

4. **Textblob:**
   - Textblob is the library that I used for the sentiment analysis function that looks at the positive versus negatives tweets. It basically is a built-in library that allows for automated polarity data graphing. Once Tweepy grabs the data from Twitter, Textblob does the graphing that is required in order to give the polarity percentage.

5. **Webbrowser:**
   - Webbrowser is a very helpful library for this application because it allows me to launch a website link for whatever I needed. When you attempt to link your account to Twitter, web browser allows me to give the link needed for it to launch the link process. If you look at in the settings tab, this library was used several times to link the user to different websites.

- Functions:
1. **def __init__(self, dialog):**
   - This function is what is called whenever a button is pressed on the interface. Here you will find *button.clicked.connect* commands which will call on the correct function that is required to complete the task it is supposed to do. Since it is initially called when the interface is started, I repainted labels to be blank or buttons to be enabled/disabled based on what the user is supposed to see.

2. **def checkRetweetGraphFunction(self):**
   - The function here makes a list of the retweet count of the tweets that you searched in the Data Interface tab. When the Graph Retweets button is pressed, it takes this list that was saved and uses the MatPlotLib to display a graph. This graph is the retweet count of the number of tweets selected in the Most Recent Order.

3. **def checkFavoriteGraphFunction(self):**

- This function is similar to the Retweets Graph Function although instead it allows to look at the favorite count of the tweets in the data interface tab. The Graph Favorites button is what calls on this function and performs the same process as the Retweets Function. Important to note that since Retweets are present in the search, favorites are a lot of times 0 because Twitter doesn’t count Retweeted Tweets Favorites. This may skew the data for the user based on the filter word.

4. **def followAppAccFunction(self):**
   - This function calls on the WebBrowser library to link you to the DataMineTweets twitter account. It is the application twitter account; this function is called on by the Follow App on Twitter button in the Settings tab.

5. **def followDevAccFunction(self):**
   - This function is similar to the AppAccFunction although it links you to the Developer Test account Twitter account, DataMineDev. This function is called by the Follow Dev on Twitter button in the settings tab.

6. **def appWebsiteFunction(self):**
   - This function is called by the App Website button on the settings tab and brings you to a general website designed for this application. This website will be further updated in the future.

7. **def contactDevFunction(self):**
   - This function is called by the Contact Dev button on the settings tab and brings you to the contact website page where it displays my information for contact info.

8. **def downloadSrcFunction(self):**
   - This function is called by the Download source button on the settings tab and basically is where you will be able to download the source files of the application once it is uploaded to GitHub.

9. **def downloadDocFunction(self):**
   - The function here is called by the Download documentation button on the setting tab and is where this full documentation will be available for download once it is approved and put on GitHub.

10. **def reportIssueFunction(self):**
    - The function here is called by the Report Issue button on the settings tab. This brings you to a direct link to direct message the application Twitter account which will allow you to report an issue directly.

11. **def checkAccountFunction(self):**
    - This function is called on by the Check Account button which looks up on the authentication what account is currently linked up to the Tweepy & Twitter API's. When clicked it'll call the function and update the label on the username that is attached currently.

12. **def unlinkAppFunction(self):**
    - This function is called on the settings tab by the Unlink account button and when clicked will bring you to the Twitter settings link that is needed to get to unlink your account. Once you are on the Twitter settings link that it brings you, you press unlink app and it will be completed.

13. **def grabAccessTokenFunction(self):**
    - This function is on the settings tab when you click the grab access token button it'll put the result grabbed from the authentication api and put it in the box leaving the user able to take it if they would like to.

14. **def checkAuthFunction(self):**
    - The function here is called by the check authentication button on the settings tab and it'll check if the Tweepy and Twitter API are successfully linked which is what you need to use all the essentials of the application.

15. **def printTweetInfoFunction(self):**
    - This function is what is called when you select one of the tweets that you searched in the user interface tab. By selecting one of the tweets you searched, it'll grab that tweet's date, time, retweets, and favorites and update the labels in the scan-info section of the tab.

16. **def printDataTweetInfoFunction(self):**
    - This function performs exactly the same as the printTweetInfoFunction but instead it is used in the Data interface tab. Selecting a tweet will update the scan-info section for that tab.

17. **def testingRowResponseFunction(self):**
    - The function here is used to delete your own tweets, if you attempt to delete someone else's tweets it won't allow you and the label will be updated. If you link your account, once you search the tweet that you want, click the tweet and press the delete tweet button for it to be removed from Twitter.

18. **def linkToConnectTwitterFunction(self):**
    - This function is called by the Log in to Twitter button on the User Interface tab and is the first step on connecting your account. When clicked it'll bring you to my Twitter application authentication page which

you need to allow. It'll then return the pin that is needed for the next step.

19. def submitPinToConnectTwitterFunction(self):

- The function here is called by the Submit Pin button which is enabled after you get the pin from the linkToConnectTwitterFunction. When the pin would be entered into the input box, the button will update the API authentication with your account access token and will allow access to run under your account.

20. def grabDataSet1Function(self):

- This function is step one of the compare data in the Data interface and is what is called when you press Save Data Set 1. It'll set the variables needed in order to do the graphing in compareDataSetsFunction.

21. def grabDataSet2Function(self):

- This function is step two of the compare data in the Data interface and is what is called when you press Save Data Set 2. It sets the variables needed for the other half of the graphing in compareDataSetsFunction.

22. def compareDataSetsFunction(self):

- The function here is called by the Compare data button on the Data interface and takes the information that was stored from the grabDataSet1Function and grabDataSet2Function. It then calls on the matplotlib library in order to create the graphs.

23. def clearDataSetsFunction(self):

- This function is called by the Clear data button on the Data interface tab and clears all the variables that were stored by the grabDataSet1Function and grabDataSet2Function allowing the user to get rid of the current data stored so they can store more.

24. def graphData1Function(self):

- This function is called by the Graph Current Data button in the data interface tab for the sentiment analysis feature. It takes the current filter word and number of tweets put in by the user and calls the method in order to look at the polarity values. With this data creates a line graph and gives the average polarity of the tweets selected which helps compare positive versus negative.

25. def postTweetFunction(self):

- This function is called by the post tweet button on the user interface tab and is enabled when the user authenticates their account into the application. Once enabled you can put your tweet you would like to post in the input box and press Post Tweet.

26. def searchStreamFunction(self):

- This is the main function that is used by the Data interface tab because once you enter all the information necessary to create a search, this function will be called by the Search button. It uses the api.user_timeline function in order to grab the tweets that fit the filter data given. This function is essential for the Data Interface because it is also called when you want to graph the data.

27. def addTwitterHandleTextToFunction(self):

- This is the main function of the User Interface tab because similar to the searchStreamFunction it'll look at the user handle, number of tweets, and filter word and try to grab the data from Twitter. It'll be presented in the list widget to the right of the Search button. It updates multiple labels and other widgets in order for the other buttons to be able to work.

b. *firstgui2.py*

- <u>Summary:</u>
  This is the main file for the GUI interface that is displayed by the application when progButton.py is ran. This file is actually generated by the Qt Designer as a Qt Designer designated file. It is important to name all the buttons, widgets, labels appropriately because you need to call on them in order for them to be edited, enabled, disabled, call a function or method. If you open this file you can see all of the widgets, buttons, labels, input boxes and more that were used to fully create the interface. If you wanted to edit this interface you need to download the Qt Designer application and open the *firstgui.ui* file with it.

- <u>Tutorial:</u>
  In order to make it into a python file you must locate the file where it was saved as in this case it was called:
  firstgui.ui,

  Then you call the command in your terminal to convert to a python file:

```
pyuic5 firstgui.ui –o firstgui2.py
```

  This will output the firstgui2.py file which can be dragged into the Project Files which is called on by the progButton.py to display the interface created.

c. *Icons-folder*

- <u>Summary:</u>
  All of these icons were used to add more color to the interface whether that of been in the button to show what it does or with the tab overviews. All of these icons were

inserted in the same folder to keep it more organized and keep it from crashing when added in through Qt Designer. If you want to download the icons, I found them all online through an essential-collection pack located on flaticon.com. This is the link needed to access this icon pack:
https://www.flaticon.com/packs/essential-collection

♦ **Qt Designer File:**

a. **firstgui.ui**

- Summary:
  This is the interface that was created directly in the Qt Designer before it was turned into a python file with the tutorial to do that located above. In this Qt Designer is where I created the whole interface and important to note that all the icons need to be added in using this application. By creating a designated icon folder, the file will know where it is located even when it is translated into a Python file. This Qt Designer is extremely easy to use and highly recommend if you wish to edit the interface or add any additional content with more control. The Qt Designer file will be inserted in the Project Files folder in case it is needed.

3. Major Issues & Solutions

- Filtering
  - Filtering was the major issue while working throughout the report because I wanted to just have it so the user only needed to use one tab in order to search a user or just all of Twitter in general for specific filter words. The problem I ran into was the limit of api methods that they offer because api.user_timeline is specific to a user but cannot be used just for filter words. Then with api.search you can search key words but that is general to Twitter not users.

  - **Solution:** The solution here was to make the user interface which uses the api.user_timeline specific to the user and then I was able to filter the words myself in a loop where it goes through all the tweets and checks for the filter word that you give. Then with the creation of data interface this one uses api.search which allows the filter word but just searches all of Twitter not specific to a user. This was helpful because api.search was really useful for allowing graphing more capabilities.

- Graphing
  - Graphing was a major issue since the beginning because I wanted to use QtDesigner and make a custom widget in order for me to offer all the graphing issues I wanted. Although it is quite complicated to actually implement a custom widget so I had to look elsewhere on a more realistic way of figuring out a way to allow users to graph in multiple different ways.

    - **Solution:** The solution I picked was using the matploblib library which allowed me to have a custom interface pop up when the graphing buttons were pressed appropriately by the user. Putting the tweets into lists allowed them to be put into a graph form.

- Icons and Graphics
    - Icons and graphics were quite difficult to implement in the beginning because when I added them into Qt Designer they would be lose on the transfer to a Python file. Also, to the fact that the sizing would always be wrong when it was actually implemented.

    - **Solution:** By creating a separate folder when implementing into Qt Designer that was actually pre-loaded into my project files in Python, it would never be lost on the transfer to a Python file. The reason for this is that the pathing problem was solved so it was looking for it in the same spot now.

- Token Authentication
    - It took a while to figure out a way for the users to log into their own Twitter account so that they could post, remove tweets, and look at private accounts if they wanted to. The reason being is that Twitter is very strict on their privacy settings so it is important to them they don't share access of people's username and password.

    - *Also, there is an issue with authentication where Twitter API services always will randomize my access token every once in a while, which if this occurs I need to update this in the project for authentication to work again which is out of my control.*

    - **Solution:** The way I was able to allow people to log into their own Twitter account is by setting up a callback_url in which it will bring the user to a separate page which the user must accept in order to give my application access to their account. This was a problem because the user has to manually give back the PIN in order for the application to complete verification so my only option was to have a box where the user enters the pin that is given at authentication.

4. User Tutorial

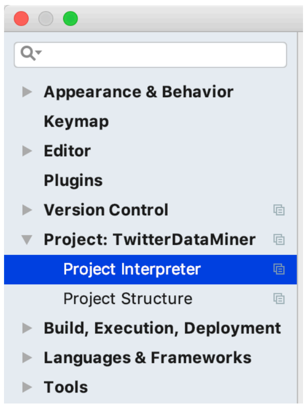
- How to Start the Main Application:
    - After downloading the project files, you can open it using an IDE like PyCharm CE, here is a download link to it:
      https://www.jetbrains.com/pycharm/download/#section=mac

- Once it is open, in order to have it run in order for the interface to show is running the Python file:

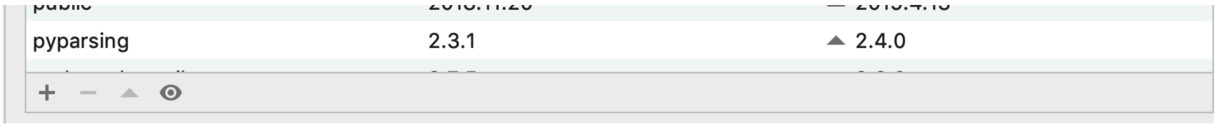
  progButton.py

- If you need to download the libraries that are used if you are getting an error, you can navigate through the PyCharm settings by doing the following:
  a. PyCharm -> Preferences

  b. It will present you with this interface:



  c. Clicking Project Interpreter will bring you to see this at the bottom:

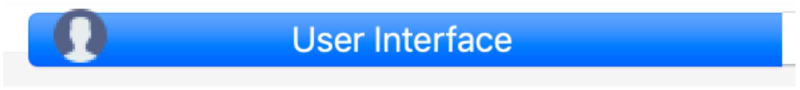


pyparsing	2.3.1	▲ 2.4.0

  d. Click the "+" button and will allow you to type in the Available Package that you need, you need to install:
     - PyQt5, Matplotlib, Tweepy, Textblob, Webbrowser

  e. Click Apply on the way out after you've done so and you are done.

- This will present the primary interface labeled at the top Twitter Data Miner and will be presented with User Interface:



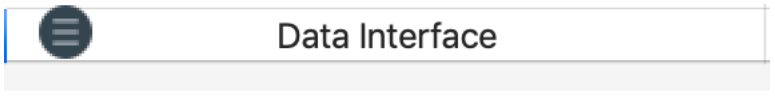
  a. Filling out the user handle, filter word, number of tweets options and pressing search will use the first feature in this tab by displaying the tweet results appropriately.

  b. Click the tweet to see how the date, retweets, favorites changes in the bottom left.

c. You can log in on this tab by following the Log In To Twitter option, once you press accept Twitter gives you a pin that you must submit in the pin text box.

d. Once authentication works you may post tweets using the Post Tweet button and text box, or select your own tweets and press Delete Selected Tweet.

- Switching to the data interface is the second tab and looks like so:



a. This is different as it is not specific to the user for this search will allows you to search out all of Twitter.

b. Filling out the filter word, number of tweets, and filter type will present you with the tweets that fit that search.

c. Click the tweet to see how the date, retweets, favorites changes in the bottom left.

d. Pressing Graph Current Data in the sentiment analysis will allow you to see the positive versus negative tweets of your search including the polarity average.

e. You can compare two filter words for the popularity based on retweets by searching one filter word pressing Save Data Set 1, then putting a new filter word and pressing Save Data Set 2. Press Compare or clear to graph or put in new data.

f. Lastly you can graph your current search based on the number of retweets or favorites if you would like by pressing the following to the right of the previous tab.

- Lastly there is the settings tab which is mostly links to helpful information to the user or look how it saves the user and data history on recent searches, the setting tab looks like so:

- <u>How to Edit the Qt Designer File:</u>
  1. In order to edit the Qt Designer file, you must download Qt Designer here: https://build-system.fman.io/qt-designer-download

2. Once it is installed, you can open the file:
   firstgui.ui

3. This will present you with an editable, draggable interface in which you can add widgets, buttons, labels, and much more. In order to turn this file into a Python file, there is a tutorial located above.

5. Conclusion & Future Work

- Future Updates:
    - During the summer and so on I would like to continue working on this application as I saw it as a really fun project to work on with a lot more potential. The most primary part that I would like to work on is more graphing features with it more cleaned up. Right now, it does what I initially had in mind but offering more additional features like graphing multiple filter words together would be ideal.

    - Cleaning up the interface would be one of my goals of my future updates because of developing it on a Mac actually limited how the customizability on what it looked like. Especially regarding color and design of the overall interface.

    - As the final product for now, it works well written in Python because of the libraries that are supported for the Twitter APIs. Although if I had the ability to, I would like to turn this application into a web-based application. This would allow for users to use this application easier and just expand the audience it is costumed to.

- Overall Conclusion
    - I am overall happy with the current state of the Twitter Data Miner but do believe with more time I will be able to improve some features. It was enjoyable learning how to program in Python and finding new ways to implement an interface using QtDesigner for the first time as well. The Twitter API was an interesting one to learn in my opinion because how big Twitter and social medias are. I am going to continue to work on the Twitter Data Miner in the future in my spare time and believe it has potential use from people.

    - In the end, my Twitter Data Miner allows users to be able to post, find, filter, and delete tweets. Allowing users to log in, allowing access and authentication from my application. Searching tweets is available to users or businesses while filtering specific words and types. Also, can graph to look at the sentiment analysis of average polarity of the searched tweets the user gives. Compare filter words with a number of tweets in all of

Twitter by looking at the graphing data of retweet counts. Also look at individual data searches and check how it fluctuates with the most recent tweets based on retweets and favorites. The settings are important for the user if you wish to access certain links or previous search history. Overall, my Twitter Data Miner is successfully doing what I set the goal to be able to in the beginning.