Nicolas Ortega

Colleen Van Lent

SI 206

15 December 2017

SI 206 Final Project Report

1, 2. Goals and how I achieved them

For this project, I decided to create a tool that would request for 100 tweets from an entered user and perform a text analysis of those 100 tweets based on syntax, vocabulary, and other factors determined by the text analysis API (Indico). I used the tweepy SDK to retrieve the tweets from the user, and passed the text of these tweets to another API, Indico, which helped me create an emotional and personality profile for the user based on the text of those 100 tweets. As per the Indico documentation page, the personality function used by the API “returns a dictionary that maps the following personality traits to their likelihood of describing the author: extraversion, openness, agreeableness, conscientiousness.” The emotion function “returns a dictionary that maps from 5 emotions (anger, fear, joy, sadness, surprise) to the probability that the author is expressing the respective emotion.” Using these scores generated by Indico, I then used the plot.ly offline graphing visualization tool to create grouped bar charts to show the average personality trait and emotion score for each day of the week, which I calculated using some basic list comprehension. The scores range from 0 to 1, representing percentage of the emotion or personality trait displayed. The plot.ly tool will create 2 HTML files, one for each function (personality and emotion), and store them in the local directory. Further, I wrote the information returned (user entered, text of the tweet, time of the tweet, dictionary with personality scores, dictionary with emotion scores) into a new database called twitter\_info. I ultimately decided to enter the information by creating a new table every time a user is entered by using the .format() function, making the database look cleaner and bypassing any unique error I may encounter. Finally, I included some final unit tests that I was using to write the code, mostly checking on length and type to not use so many API requests, as well as to check the length of the written database. The unit tests also helped me to test different scenarios for my code to make sure everything runs smoothly.

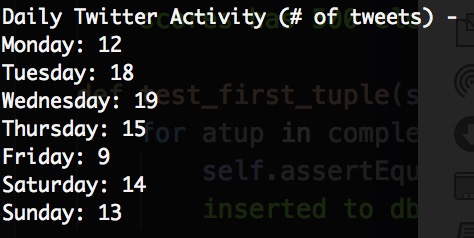
3. Problems Faced

The main problem I faced was when doing the average calculations for every category of both emotion and personality. Since we were required to get exactly 100 user interactions (tweets), sometimes the program couldn’t analyze text for a whole week since certain accounts like CNN or ESPN tweet much more often. I got an error because the length of the list in which a particular day’s scores turns out to be 0, and I was trying to get the average by computing sum(list\_scores)/len(list\_scores), and was therefore dividing by 0. The way I dealt with this was to use conditional statements, saying that if the list is longer than 0 elements, compute the average; else, just assign the score to 0. This will make sure that plot.ly still works (just that the scores for that day are empty). Other than that, another problem I didn’t necessarily encounter but could have easily is mistaken variable names and keeping track of which list is storing what. Sometimes the process got repetitive, and small mistakes were made in writing the code, but luckily I kept my code organized enough and didn’t encounter much of this problem. Constantly writing test code to check the type and length of pretty much every step also helped me stay focused and compartmentalize the different moving pieces.

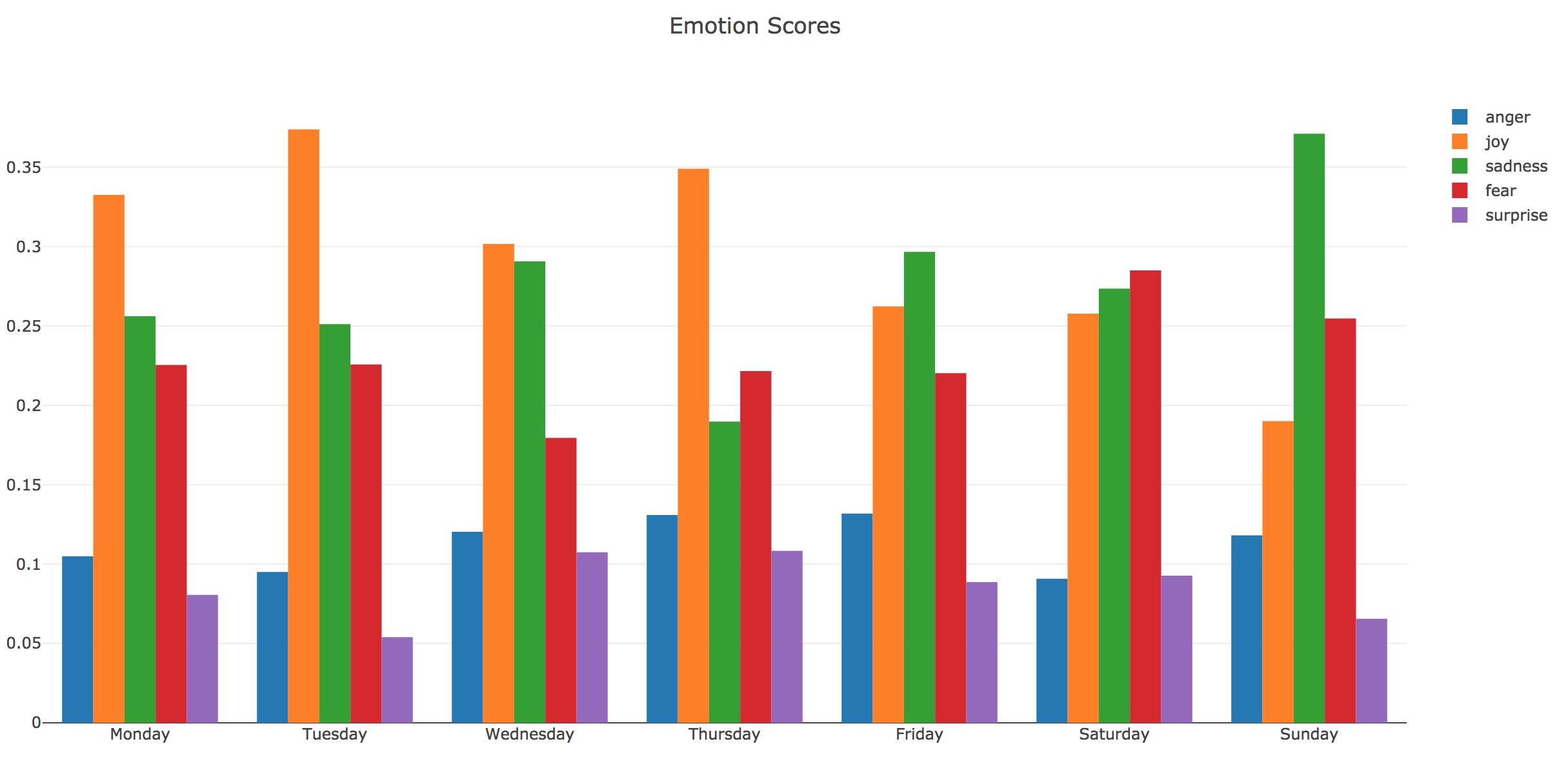
I also encountered an issue when writing the information into SQL. I want it to write the 100 tweets returned into the database only once, even if a username was entered already. I took some time figuring out the code and fixed this by doing some exception handling. I stored the cur.fetchall() for the database in a variable, and used that variable to say that if the tuple we’re about to insert is not there, then write the information to the database. Else, just continue the process. After that, there are only some extra unit tests I wrote to keep debugging. I’m also using the cache to write the information, so there won’t be updates to the database until the cache file is deleted (following the specific requirement of retrieving *exactly* 100 interactions).

4. Social Media Report

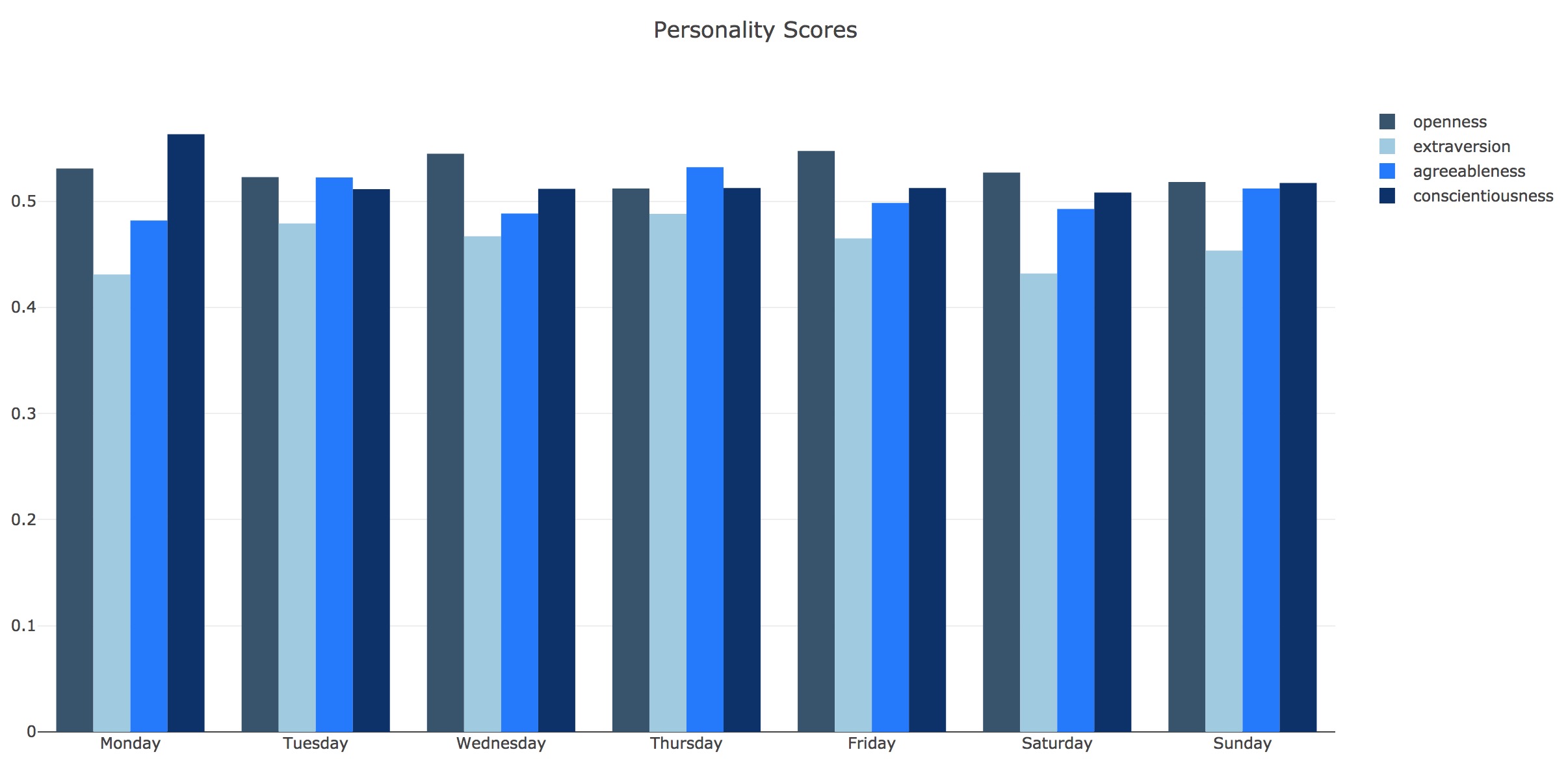
The program generates an output within terminal and two HTML files that open in your browser, apart from writing the information to the database.

This first output seen in the terminal uses the tuple variables used before to show the activity on a social media site. This sample output is from my own twitter account, and I learned that I’m more active in the middle of the week than at the end, which could be a powerful tool to gain a deeper understanding of user activity. The breakdown will help further to interpret the data shown in the grouped bar charts generated by plot.ly that follows.

Below is a sample HTML output from the code, taken from Donald Trump’s twitter account:



* The graph’s y-axis shows the average percentage of each trait portrayed, and the x-axis contains a label for each day of the week. Each has a legend to check which trait has what percentage on which day.



The output shows grouped bar charts, separated into days of the week, with each bar representing either a personality trait or an emotion. When we start cross-analyzing the bar charts with the text of the information written into the database we see certain patterns. For example, looking at Sunday’s emotion score, we see that sadness is the prevalent one. This is because that Sunday Trump was tweeting about the China shoplifting accident, the decision of big game hunting, and other issues that are somewhat appropriate for a Sunday. It seems that Trump tweets about certain harsher things on a Sunday, maybe because there won’t be as many replies or because there’s no obligation to explain himself formally. Although this might be considered extrapolation, with enough data points we can gain a deeper understanding of user behavior through tweeting patterns. His famous “short and fat” tweet is in the database, and was tweeted on a Sunday. Even just at looking the graph you can see how joy dwindles as the week progresses, and every end of the week seems to be a battle for Donald Trump. The mix of prevalent emotions (joy, fear, sadness) also reflects on the subject analyzed.

This analysis, although rather basic, could be extremely important in managing outside appearances and public relations, especially in the age of social media. If you know that sentiment and emotion can be at least loosely quantified, then you can use it on your favor and employ key words that will convey your message without losing its integrity while reaching a more specific but passionate audience. If we use more data points in the code, we could segment times of day where people are sad and angry, and group them with appropriate subjects that will respond to the message by exhibiting their prevalent emotions. Large data analysis firms like Cambridge Analytica have used audience segmentation to make their clients aware of how you are analyzed and interpreted by the digital world, which is a powerful step to use social media efficiently and to your benefit.

5. Instruction for Running Code

First, we need to install the API packages we need for the program to run.

* 1. For **tweepy**, access your terminal and do pip install tweepy. If you’re protected by a password, use sudo pip install tweepy.
  2. For **Indico**, there are two methods to install the package:
     + (sudo) pip install indicoio *OR*
     + git clone <https://github.com/IndicoDataSolutions/IndicoIo-Python.git>  
       python setup.py install

Once you installed both packages in your terminal, we need another step before we can run the code.

1. For tweepy, you need to get your twitter access keys to authenticate and make the code run.
   1. Go to <https://apps.twitter.com/app/new> and register your app to get your access tokens. Alternatively, if you have access tokens already, you can copy and paste your keys into the variables within the code itself.
   2. As an alternative to copying the keys to the code, save your tokens in a python file called ‘twitter\_info.py’ within the same directory (file included, just open it and fill the empty variables), and use the format already in the code to access those tokens and keys.
2. For Indico, you need to register for an account.
   1. Go to <https://indico.io/pay-per-call> and register by following the instructions on the screen.
   2. Once you successfully registered, Indico will ask for your credit card information in case you go over the 10,000 requests limit per month. This is **not** necessary, you can just close that window and use the 10,000 free requests for this project.
   3. Once you complete the registration process, locate your API key in your Indico dashboard, and save the key into the variable in the code called indicoio.config.api\_key (variable has already been defined).

Now we’re ready to successfully run the code. Go to your terminal, go to the project directory and run the code: *python 206\_Ortega\_Final\_Project.py*. It may take a few seconds to initialize, and when it does it prompts you to enter a twitter handle. Try your own or someone else’s! You still can’t get information from private accounts.

6. Function Documentation

Throughout the project, I wrote a total of 6 functions to gather the data needed and visualize it appropriately.

1. get\_tweets –
   1. This function uses an if/else statement to either return the tweets from the cache file, or fetch the tweets using the tweepy API and write them to the cache file for future use.
   2. It’ll return a list with all 100 tweets of the specified user in JSON format, all information included.
2. get\_tweets\_text –
   1. This function accepts last function’s output (list of tweets) and returns a list with only the text of the 100 tweets to be analyzed later.
3. get\_tweets\_time –
   1. Like the last function, this function accepts the raw data generated by the first function and returns a list with all the dates of the tweets (accessing the “created at” key inside the tweets list).
4. analyze\_text –
   1. This function accepts the output of the second function (the list with only the text of the tweets).
   2. It also initializes two empty lists to store the scores for emotion and personality returned by the Indico API.
   3. The function returns two lists of dictionaries (since Indico returns the scores in dictionary form), one with all the personality scores and one with the emotions scores.
5. graph\_personality –
   1. This function first defines the user as a global variable to use for our HTML output files.
   2. The function uses the plot.ly offline service to generate two grouped bar charts and store them in the repository where the project python file in the format [entered\_user] \_personality\_summary.html
   3. The function uses plot.ly built-in functions to create bar charts for each category, and it uses the data gathered in the earlier variables to plot the average scores per day.
6. graph\_emotion –
   1. The function uses the user as a global variable once again.
   2. This function is similar to the previous, but instead of entering the average personality scores to the function, I use the average personality scores. The graph created will be saved as [entered\_user]\_emotion\_summary.html
   3. Immediately after the two previous functions, I executed them: graph\_personality(), graph\_emotion().

7. Resource Documentation

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Issue Description** | **Location of Resource** | **Result (solved?)** |
| 12/5/2017 | Couldn’t write a table by using a variable name. | <https://stackoverflow.com/questions/34392011/create-table-using-a-variable-in-python> | Solved, using .format(user) |
| 12/6/2017 | Was trying to use plot.ly API and store the HTML file online. | <https://stackoverflow.com/questions/37745917/using-plotly-without-online-plotly-account> | Solved, had to use offline graphing and store locally. |
| 11/29/2017 | Wanted to create a giant list of tuples with the information to be entered to database | <https://stackoverflow.com/questions/13704860/zip-lists-in-python> | Solved, zipped all necessary variables except for the entered user (included later) |
| 11/23/2017 | No issue. This is the tweepy documentation. | <http://docs.tweepy.org/en/v3.5.0/api.html#tweepy-api-twitter-api-wrapper> | Used user\_timeline method |
| 11/25/2017 | No issue. This is the Indico documentation. | [https://indico.io/docs#](https://indico.io/docs) | Used the emotion and personality analysis methods. |
| 11/28/2017 | No issue. This is the grouped bar chart documentation for plot.ly, followed by the entire plot.ly documentation link. | <https://plot.ly/python/bar-charts/#grouped-bar-chart> | Used the offline plot.ly graphin power, and got help from the code snippets. |