Mateos Norian

Assignment 2

Project overview

All data used in this project was taken from Venmo, using the Venmo API Python wrapper. I analyzed the Venmo transaction notes between users. In case you're unfamiliar with Venmo, when you pay someone on Venmo users leave a note describing the payment -- the text within these notes is what I analyzed. I focused on extracting the most common words that appeared in notes, and the most common emojis that appeared in notes. I hoped to create a "Venmo dashboard," where a user could login to a web app and view some interesting statistics about their transaction notes and potentially some of their spending habits.

Implementation

The project had a few major components. The first major component was the venmo_functions.py file. This file contained all of the data processing functions that handled the Venmo API responses and broke them down. For the most part, I used a combination of dictionaries and lists to sort through the Venmo API responses, and to calculate the frequency of words and emojis. The file also included functions that read user data from a file, and used this data to generate a Venmo access token to make API calls on the user's behalf.

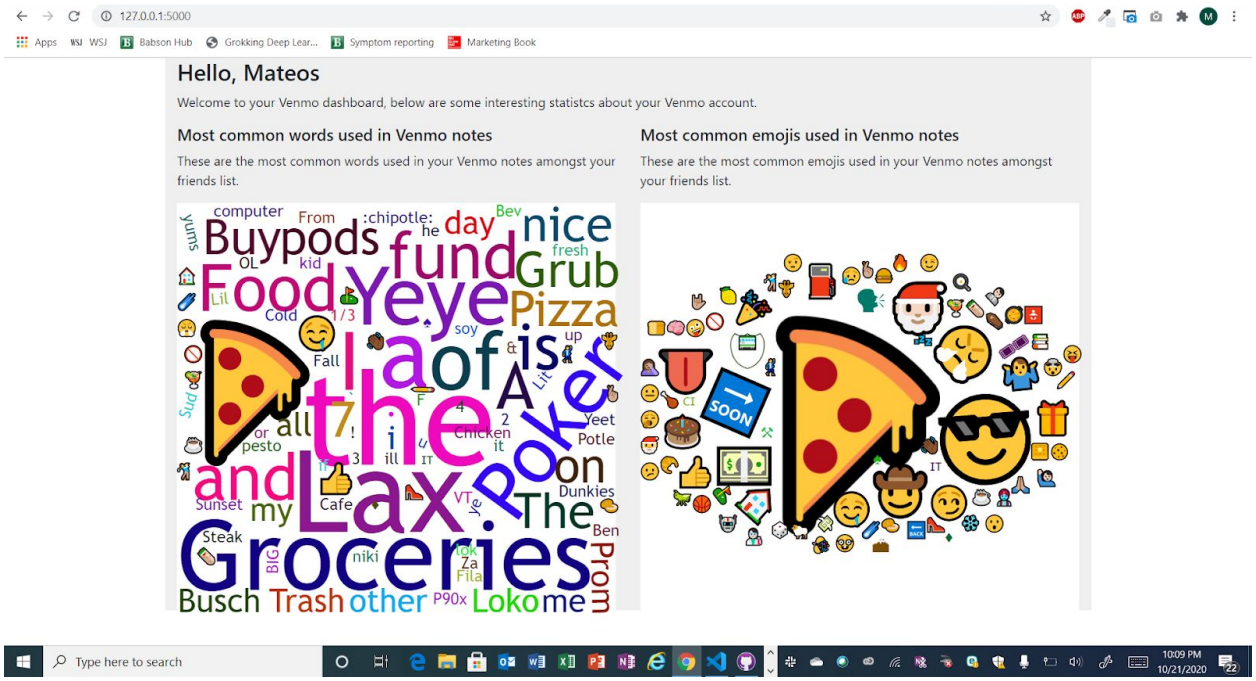
The second major component of the project was the "Flask" section. The Flask section consisted of a template, for the "/" route, and was used to display word clouds that were generated based on the information output from the venmo_functions.py file.

The third major component of the project were the libraries/wrappers used to make the API calls and word cloud generation possible. For the Venmo API calls, a GitHub user had created a Python wrapper for some of the most useful API calls, such as "get user info" and "get user transaction info." For the word clouds, wordcloud.js was used to generate word clouds on an HTML canvas.

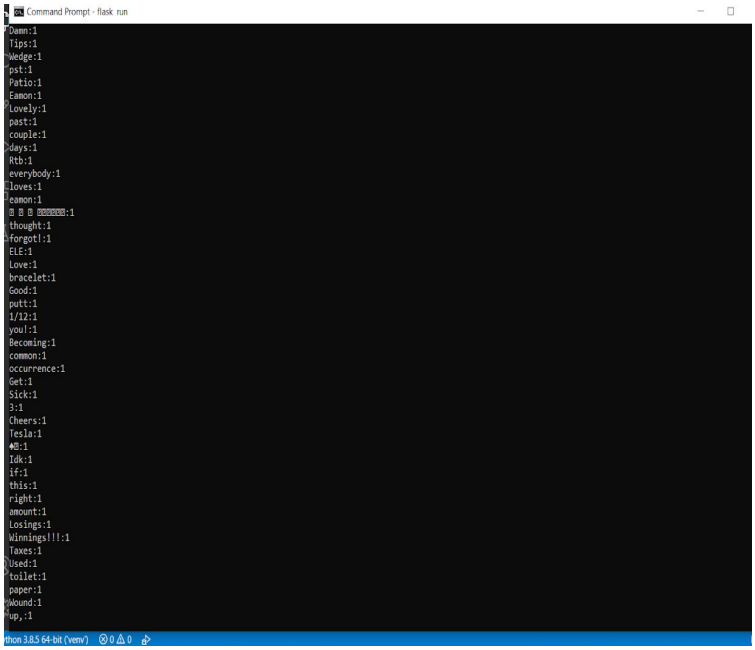
The largest design decision I made was deciding to stop trying to store API response data in an external file, and then retrieving it from that file. Instead, the API is called every time a request is made to the Flask server. I would've preferred to be able to store the API response data within a file instead of constantly making calls to the API to receive the same data in response, however, the serializing and deserializing of complex objects required to read/write the objects to the file set me back about six hours and created a significant amount of errors in my code.

Results

The text analysis was interesting because most of the user transactions analyzed were not full sentences. They were usually short phrases or one word, and a considerable amount of them were just one emoji. As you can see from the word cloud of transaction notes, there are many slang terms like “YeYe, Dunkies, Potle, Yeet, Lax.” The most common word was “the,” which was somewhat surprising to me because most of the transaction notes were not formatted as sentences, where the word “the” would commonly be found.



Screenshot of the web application displaying word clouds based on my transactions and my Venmo friends’ transactions



Screenshot from the terminal of the transaction notes being split into individual words and counted

Interestingly, the emojis were the most difficult text to analyze because of their encoding. Some emojis can be made up of multiple emojis, which makes them difficult to count. Characters that are made up of multiple characters are known as “graphemes,” to ensure that I wasn’t counting the emojis that combined to create the final emoji, I had to only count the graphemes.

Reflection

In terms of the process used to create this project, the most effective aspect of my approach was the amount of time I allocated myself to completing it. I spent roughly an entire week coding for 2-3 hours per day because I knew the size of the project would be difficult for me to handle if I left myself only a few days. The week-long “grace period” I gave myself eased my stress because I knew that I still had time to make the deadline, and I was not rushing or cutting corners in my code. I did not prepare adequately for how much reading would be required to understand the Venmo API, and its Python wrapper. In the future, I’ll know that integrating any third party library, especially one that is not documented for beginners, will take a significant amount of reading and tinkering to implement. Going forward, I’ll be sure to allocate more time to reading third party code, and giving myself time to understand it before getting impatient and doing “guess coding.”