Lucas Hazan, Chris Chomicki , Justin Quinonez

**SportsTracker: The Writeup**

The pretense of SportsTracker is to see the correlation between how college teams performed in games and what people tweeted about before and after the games. Using this tool, college athletic departments could determine whether or not they are engaging their fans effectively on game days. It can help to make social media departments make data driven decisions about where they need to focus their efforts on fan engagement during game day. Our idea first sprouted from the code challenge we had when we did the twitter sentiment analysis, but we chose to expand even more on it. Using datasets from a website known as “sports-reference.com”, we were able to get the schedule and the results of the sporting events for each college we chose to use in our project. We chose to work with football and basketball for our sport categories because these are the two most trending sports on twitter. We first wanted to just do Syracuse University and see what people said about basketball and football games before they occur and what was the result afterwards, however, we chose to change it to include more schools, thus expanding the scope to an even larger level for the project.

How did we scrape all the results from these websites? Through using pandas we were able to scrape data from HTML tables, and acquire the data of wins and losses for football and basketball seasons all throughout the 20th and 21st centuries for over 400 universities. A user can specify in our application what year, sport, and school they want to see information from so that we aren’t querying for information about every single sport, school, and year each time a user enters a query. From the information input by the user, we would format the url to display the information that they wanted. We also included error handling just in case the website didn’t have information for that query or the user entered a year that does not make sense. This provided us the ability to get the dates of when each game occured for that season in order to search for tweets on those dates. Although, the dates that were on “sports-reference.com” were formatted differently than the needed input for the “GetOldTweets” tool to filter by the date for the game day. Utilizing tools from the datetime module, we were able to format the dates from an example like “Wed, Dec 6, 2017” to the needed format of “2017-12-06”. The data was stored into a dataframe in order to be utilized within SportTracker. We were not able to use the Twitter API to gather data because it restricts users to searching within a 7 day time frame. In order to search for tweets older than 7 days, we found a tool on Github called “GetOldTweets” created by a user named “Jefferson-Henrique”. He has this tool implemented in Java as well, but we clearly decided to go with the Python implementation. This tool allows a user to create search queries on Twitter by adding parameters onto the end of a twitter url based on keywords, locations, etc. He had the tool implemented in Python 2 so we had to change the tool around in order for it to work with Python 3 because the tool had not been fully implemented in Python 3. The Python 2 implementation included the ability to search by location, so we ported that over to our Python 3 implementation. This included updating the modules used to make web requests (urllib2 -> urllib.request). In the end, we created a new fork in his Github repository that included our changes that we made to his code. The user “Jefferson-Henrique” was actually grateful that we finished the Python 3 implementation because he had given up on the project long ago.

We use the school name as a keyword and location search within 15 miles of the school. The tool would then make a request to that url and begin scraping the HTML elements from that webpage using another tool called “PyQuery”. It’s very similar to “JQuery” in Javascript where it gives developers the ability to target DOM elements by their tag, id, or class in order to manipulate them. Since each tweet was an object in that list, it was pretty simple to calculate the length of that list using the “len(list)” function and returning it to specify the amount of tweets on that game day using the keyword of the school name itself. Using the amount of tweets per game day in that area, we then created a new dataframe listing each game of that season along with whether the school won or lost and the number of tweets for that game. By doing so we were able to sort the data and than take the data of results of the colleges and placed it in csv files to present by using data visualization.

Instead of using plotly, we decided to go with an easy to use Javascript library called ‘Chart.js’ that worked seamlessly with Flask. Chart.js allowed for us to do all of the graph creation directly in the browser while still utilizing the data from our Python program. All we had to do was format the data into lists that could then be interpreted by Chart.js to make our bar graphs. We also display whether the game was a win or loss by the border of the bar. (Green == ‘win’, Red == ‘loss’) Chart.js allowed us to easily customize and render html charts right in the UI and was perfect to use with Flask.

As far as who did what for the project, the group was able to make changes and contribute to the project altogether. Chris was the mastermind behind scraping all the wins/losses off of sports-reference.com and getting the tweets, he also was able to make the website UI with Flask and Javascript/JQuery. Lucas was able to make all the data capable of being visualized through bar graphs and also made the spectacular logo for our posterboard and application. Justin finalized and made sure all code worked and wrote all the exception errors throughout the python files to ensure everything is functioning accordingly. He was the creator of the flowchart that shows how our program should work and is also the one that made the poster board.

On the website, you are given a UI that is a drag down menu that gives you the choice to either choose football or basketball as a sport. You would then choose which college you want to see stats about. A user would next enter the year of the season they want to analyze. The statistics for the wins and losses for that season are then presented. A user would then begin to scrape the tweets for those days. Afterwards, a CSV file would then be created with all of the data about the wins/losses, games, and number of tweets. Then the user can go to our data visuaulization to view the data they just accumulated in a bar chart format thanks to Chart.js. The data is presented on the UI in visual manner by presenting the differences between the number of tweets on game days for both wins and losses. Our project first started as just a small idea to just analyze tweets for Syracuse men’s basketball but expanded to become much more because with software development, your imagination is not limited. We decided to go above and beyond with SportsTracker. We hope you enjoyed our project and understand the sweat and work that went into making this program and project come alive.