



WINGSPAN

Final Presentation: Part C

MOTIVATION

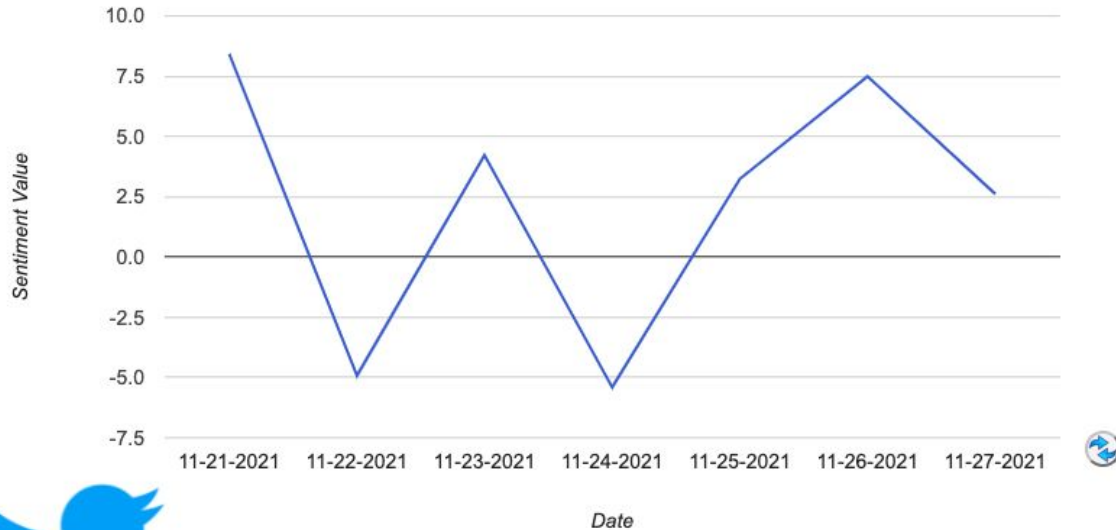
- Sentiment analysis has a wide range of applications from personal decisions, such as what movie to watch, to professional decisions, such as what stock to invest in.
- WingSpan gives users a dashboard of insights about the sentiment of Twitter users about a given topic.
- We were hoping to provide an application that allowed users to analyze changes in sentiment over any given timeframe.
- We focussed our application on providing sentiment analysis over a medium length timeframe of one week.
- This still provides most users with valuable information, but will not work for users wanting to do long term analyses or for users wanting to analyze real time sentiment changes.

USER BENEFITS

- The different graphical displays of the data allow the user to make predictions on future sentiment scores and make decisions involving the search topic based on the data.
- Shifts in sentiment on twitter may correlate with shifts in the stock market.
- It can also be used for market research and customer service approach since companies can see what the public thinks of their own products/services as well as those of their competitors.
- Users are able to filter their searches so they only see tweets from a certain Twitter user/users. With this feature, the user can gain knowledge about what the particular Twitter user/users thinks about the searched topic.



Sentiment Analysis Averages



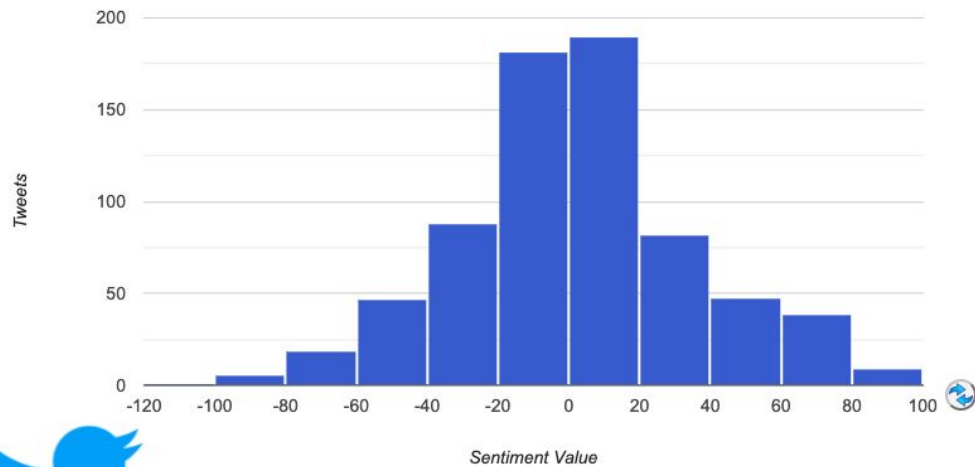
Dick Vitale ✓
@DickieV



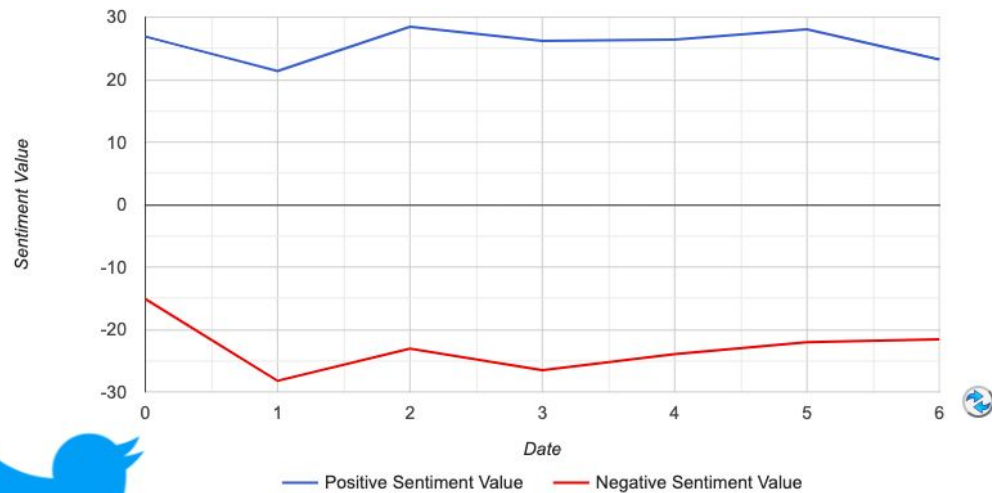
Just making final prep for game @espn 10 EST #1 GONZAGA - #2 UCLA - to be honest on Oct 12 when given the cancer diagnosis I never thought that I would be at courtside tonight / ur 🙏🙏🙏❤️❤️❤️ have helped so so much . To ALL of u in any battle "Don't ever believe in can't"!



Sentiment Analysis Histogram



Sentiment Positive and Negative Averages



FEATURES

Search Query

- The landing page has the WingSpan logo, a search bar, the twitter logo, and a welcome message with a short description of the different ways of making a search.
- This message is in conjunction with a new implementation from part b, which allows the user to make a search with a keyword, but also filter it to search only for tweets from a certain user or only from multiple users.

Graph Data

- Various forms of data regarding the searched topic. As in part b -> a line graph showing sentimental analysis averages over the past week.
- The new implementations involve
 - a histogram of the sentiment values
 - a line graph of the positive and negative sentiment value averages.
- There is a list of the top tweets that mention the requested search -> holds entire tweet objects so clicking on different parts of the tweets issues a response in the same way it would on Twitter's actual webpage.

REQUIREMENTS DESCRIPTION

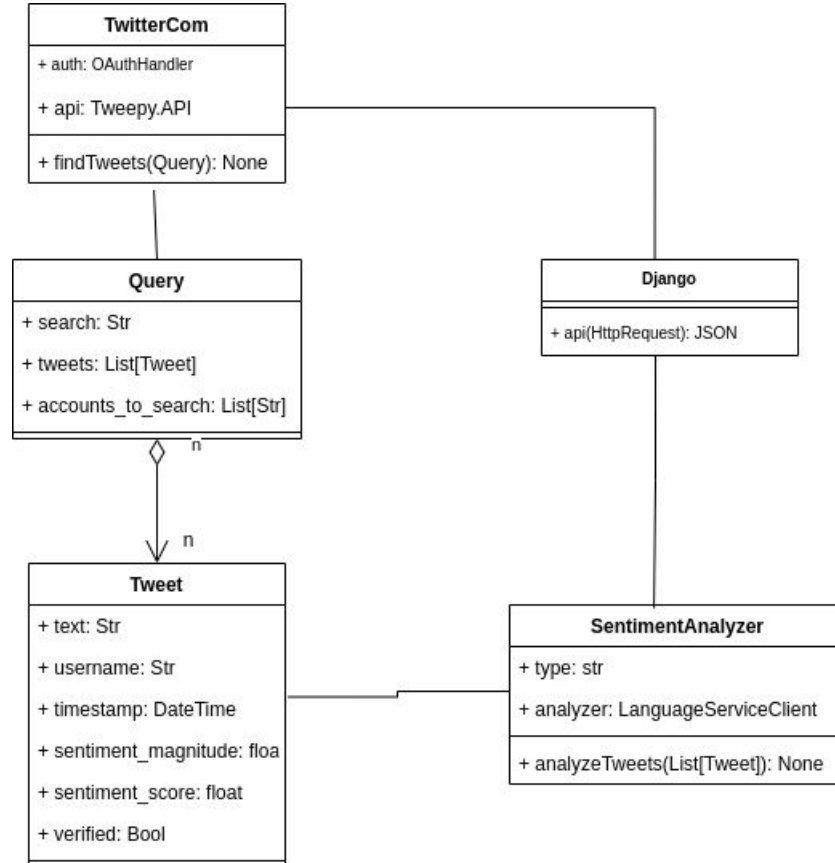
Functional Requirements

- The search boxes on the front page/data page send the search query to the Express server which then requests the data in our backend.
- The line graph shows the sentimental analysis averages over the past week.
- The histogram shows the frequency distribution of the sentiment values.
- The positive/negative averages line graph collects all sentiment values below and above zero, respectively and displays their averages.
- The list of all the top tweets associated with the searched topic holds entire tweet objects.

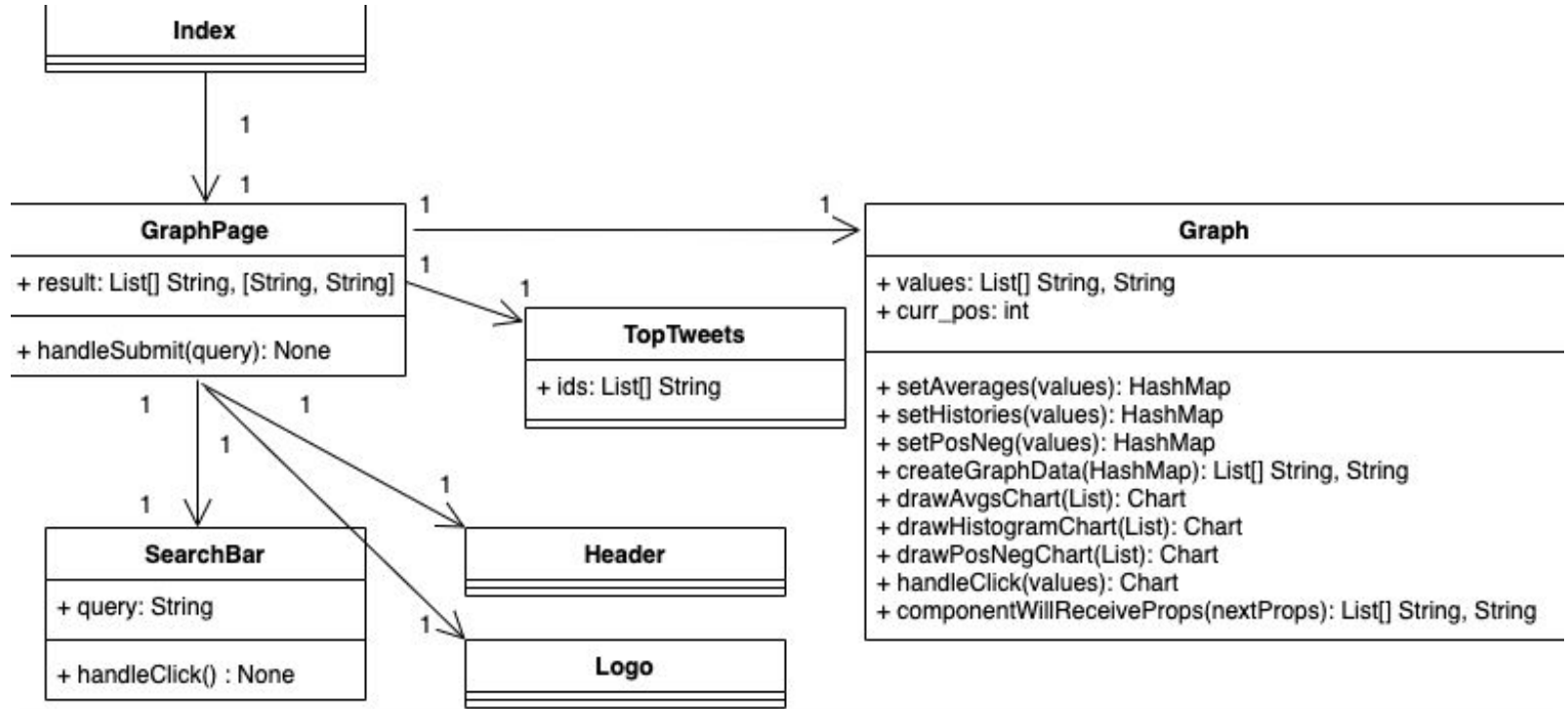
Non-Functional Requirements

- To protect the privacy of the twitter users, the top tweets displayed are from only verified Twitter accounts.
- The generation of the graphs takes a fair amount of time. In terms of performance, speed is not our first priority for this application.

CLASS DIAGRAM: BACKEND



CLASS DIAGRAM: FRONTEND



DESIGN: FRONTEND

The frontend part of the design utilizes React to create a single page web application that allows the user to send a search query to the backend.

Once the data is sent back to React from the backend, the three graphs are formed.

The user has the ability to cycle through a histogram, an average sentimental analysis graph, and a dual graph showing both positive and negative sentiments.

The graph page shows the top tweets from verified users alongside the graph information.

DESIGN: BACKEND

The backend receives the query from the frontend to retrieve information from both the Twitter API and the Google Sentiment API.

The data that is processed in the backend consists of the tweet, the time of the tweet, the sentimental value, and the tweet ID of only verified users.

The backend part uses the Django framework to prepare the twitter data and analysis to send to the frontend.

The TwitterCom class finds the tweets that correspond to the query string and user list, utilizing the Tweepy API, and filtering out duplicate tweets.

This returns a list of Tweet objects with necessary information needed to pass to the Sentiment API.

The SentimentAnalyzer takes the list of Tweet objects and finds the sentiment score -> it is calculated by multiplying the given score by the given magnitude.

API DESCRIPTION

Query

Fields: search, Tweets[]

Tweet

Fields: text, users, tid, username, timestamp, likes, retweets, verified, score, engagement

SentimentAnalyzer

```
analyzer, def analyzeTweets(List[Tweet])
```

analyzeTweets -> stores a sentiment score and magnitude for each tweet

TwitterCom

```
auth, api, def findTweets(keyword, count), def getTopTweets(List[Tweet])
```

findTweets -> returns a list of size count that contains Tweet objects with the keyword argument

getTopTweets -> returns a list of dictionaries with the “engagement” information about all of the inputted Tweets sent by verified users.

TEST SCENARIOS AND CASES: FRONTEND

Our frontend files were sufficiently modularized so they could be unit tested easily using the React extension of the Jest testing library

React allows components to be rendered into DOM format by themselves. DOM navigating functions target individual pieces and Jest provides “matchers” to test properties of a selected variable.







Tests are completely isolated; the success or failure of one component test does not rely on the success of any other component tests.

FRONTEND TEST COVERAGE

All files

98.24% Statements 112/114 92.3% Branches 24/26 96.87% Functions 31/32 98.11% Lines 104/106

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

File ▲		Statements ▾		Branches ▾		Functions ▾		Lines ▾	
graph.js		100%	83/83	90.9%	20/22	100%	15/15	100%	77/77
graphpage.js		100%	11/11	100%	2/2	100%	6/6	100%	11/11
header.js		100%	1/1	100%	0/0	100%	1/1	100%	1/1
index.js		0%	0/2	100%	0/0	0%	0/1	0%	0/2
logo.js		100%	1/1	100%	0/0	100%	1/1	100%	1/1
searchbar.js		100%	7/7	100%	0/0	100%	4/4	100%	7/7
toptweets.js		100%	9/9	100%	2/2	100%	4/4	100%	7/7

TEST SCENARIOS AND CASES: BACKEND

Used the unittest module to execute our tests, which is built-in to the Python standard library.

Used coverage.py which is a Python tool that measures code coverage and is easily integrated with unittest.

Three backend python testing files:

test_sentiment.py

test_twittercom.py

test_twittercom_bad_auth.py

BACKEND TEST COVERAGE

```
→ scripts git:(main) x ./django_tests.sh
nosetests --with-coverage --cover-package=tanalysis --verbosity=1
Creating test database for alias 'default'...
.....
Name                               Stmts   Miss Branch BrPart  Cover   Missing
-----
tanalysis/__init__.py              0        0      0      0    100%
tanalysis/admin.py                 1        0      0      0    100%
tanalysis/apps.py                  4        0      2      0    100%
tanalysis/migrations/0001_initial.py 5        0      2      0    100%
tanalysis/migrations/0002_auto_20211103_2349.py 4        0      2      0    100%
tanalysis/migrations/0003_auto_20211120_2237.py 5        0      2      0    100%
tanalysis/migrations/0004_auto_20211120_2309.py 4        0      2      0    100%
tanalysis/migrations/__init__.py     0        0      0      0    100%
tanalysis/models.py               15        0      6      0    100%
tanalysis/sentiment.py             38        6     18      2     82%   56->58, 64-69
tanalysis/twittercom.py            53        5     20      1     89%   72-73, 78-80
-----
TOTAL                             129     11     54      3     90%
-----
Ran 10 tests in 34.657s

OK
Destroying test database for alias 'default'...
```